SIV-Rijndael256 Authenticated Encryption and Hash Family

Designers/Submitters:

Zhenzhen Bao - Nanyang Technological University, Singapore
Jian Guo - Nanyang Technological University, Singapore
Tetsu Iwata - Nagoya University, Japan
Ling Song - Nanyang Technological University, Singapore and Institute of Information Engineering, CAS, China

zzbao@ntu.edu.sg, guojian@ntu.edu.sg, tetsu.iwata@nagoya-u.jp, songling@ntu.edu.sg

February 25, 2019

Chapter 1

Specification

In this document we propose the SIV-Rijndael256 family of AEAD and hash function, which utilizes Rijndael256 — the predecessor of AES — as the underlying primitive. On top of Rijndael256, SIV-Rijndael256-AEAD uses the SIV mode to enjoy the strong security against nonce-misuse and unverified plaintext release, and SIV-Rijndael256-Hash is based on the Sponge construction with Rijndael256 converted into a permutation by setting the master key to the constant 0. While the Rijndael256 primitive withstands long-term security analysis, the SIV AEAD mode and Sponge hash function construction come with well accepted security proofs. Due to the lightweightness of SIV and Sponge, the performances on both software and hardware enjoys that from Rijndael256 directly.

1.1 Notations and Preliminaries

1.1.1 Notations

Let $\{0,1\}^*$ be the set of all finite bit strings, including the empty string ε. For a bit string $X \in \{0,1\}^*$, $|X|$ is its length in bits, and we have $|\varepsilon| = 0$. For a bit string $X \in \{0,1\}^*$ and an integer $n \geq 1$, we define a parsing operation. For $X \neq \varepsilon$, it is defined as $(X[1], \ldots, X[x]) \xleftarrow{n} X$, where $|X[i]| = n$ for $1 \leq i \leq x-1$, $1 \leq |X[x]| \leq n$, and $X[1] \| \cdots \| X[x] = X$. Here $X \| Y$ is the concatenation of two bit strings X and Y. The number of blocks, x, is the block length of X. For $X = \varepsilon$, $X[1] \xleftarrow{n} X$, where $X[1] = \varepsilon$. Note that $x = 1$ and the block length of $X = \varepsilon$ is 1. For a bit string $X \in \{0,1\}^*$ and two positive integers n_1, n_2, we define a similar parsing operation. If $|X| > n_1$, it is defined as $(X[1], \ldots, X[x]) \xleftarrow{n_1,n_2} X$, where $|X[1]| = n_1$, $|X[2]| = \cdots = |X[x-1]| = n_2$, $1 \leq |X[x]| \leq n_2$, and $X[1] \| \cdots \| X[x] = X$. If $|X| \leq n_1$, including $X = \varepsilon$, $(X[1], \ldots, X[x]) \xleftarrow{n_1,n_2} X$ is equivalent to $X[1] \leftarrow X$ and $x = 1$. For a bit string $X \in \{0,1\}^*$ and an integer $\ell \leq |X|$, $\mathrm{msb}_\ell(X)$ denotes the first ℓ bits of X and $\mathrm{lsb}_\ell(X)$ denotes the last ℓ bits of X.

For $X \in \{0,1\}^*$ with $|X| \leq \ell$, we define a padding function as $\mathrm{pad}_\ell(X) = X$ if $|X| = \ell$, and $\mathrm{pad}_\ell(X) = X \| 10^{\ell-1-(|X| \bmod \ell)}$ if $0 \leq |X| < \ell$.

1.1.2 Synthetic Initialization Vector Scheme (SIV-scheme)

SIV scheme [35] combines an encryption scheme $\mathcal{E}$ and a pseudorandom function (PRF) $\mathcal{F}$ to obtain an AEAD scheme. We modify the original scheme in two ways.

- We modify the PRF so that it explicitly takes a nonce N as a part of the input.
- The encryption scheme and the PRF share the same key, and we maintain their independence with domain separation.

Fix the key length k and a block length n. The encryption scheme $\mathcal{E}$ takes a key $K \in \{0,1\}^k$, initial value (IV) $IV \in \{0,1\}^n$, and a plaintext $M \in \{0,1\}^*$ as input, and returns a ciphertext $C \in \{0,1\}^{|M|}$, and we write $C = \mathcal{E}_K^{IV}(M)$. The corresponding decryption scheme $\mathcal{D}$ takes (K, IV, C) and returns M, and we write $M = \mathcal{D}_K^{IV}(C)$. We require, for any K and IV, $M = \mathcal{D}_K^{IV}(\mathcal{E}_K^{IV}(M))$.

The PRF $\mathcal{F}$ takes a key $K \in \{0,1\}^k$, a nonce $N \in \{0,1\}^*$, associated data (AD) $A \in \{0,1\}^*$, and a plaintext $M \in \{0,1\}^*$ as input, and returns a fixed length output $T \in \{0,1\}^n$, and we write $T = \mathcal{F}_K(N, A, M)$.

Algorithm $\mathsf{SIV.Enc}_K(N, A, M)$

1. $T \leftarrow \mathcal{F}_K(N, A, M)$
2. $C \leftarrow \mathcal{E}_K^T(M)$
3. **return** (C, T)

Algorithm $\mathsf{SIV.Dec}_K(N, A, C, T)$

1. $M \leftarrow \mathcal{D}_K^T(C)$
2. $T^* \leftarrow \mathcal{F}_K(N, A, M)$
3. **if** $T^* = T$ **then return** M
4. **else return** $\perp$

Figure 1.1: The encryption and decryption algorithms of SIV scheme.

Algorithm $\mathcal{E}_K^{IV}(M)$

1. $(M[1], \ldots, M[m]) \xleftarrow{n} M$
2. $S \leftarrow IV$
3. **for** $i = 1$ **to** $m - 1$
4. $\quad S \leftarrow E_K^7(S)$
5. $\quad C[i] \leftarrow S \oplus M[i]$
6. $S \leftarrow E_K^7(S)$
7. $C[m] \leftarrow \mathrm{msb}_{|M[m]|}(S) \oplus M[m]$
8. $C \leftarrow (C[1], \ldots, C[m])$
9. **return** C

Algorithm $\mathcal{D}_K^{IV}(C)$

1. $(C[1], \ldots, C[m]) \xleftarrow{n} C$
2. $S \leftarrow IV$
3. **for** $i = 1$ **to** $m - 1$
4. $\quad S \leftarrow E_K^7(S)$
5. $\quad M[i] \leftarrow S \oplus C[i]$
6. $S \leftarrow E_K^7(S)$
7. $M[m] \leftarrow \mathrm{msb}_{|C[m]|}(S) \oplus C[m]$
8. $M \leftarrow (M[1], \ldots, M[m])$
9. **return** M

Figure 1.2: The definitions of $\mathcal{E}$ and $\mathcal{D}$.

With these components, the encryption algorithm of SIV scheme $\mathsf{SIV.Enc}$ takes a key K, a nonce N, AD A, and a plaintext M as input, and returns a pair of ciphertext and tag (C, T). We write $(C, T) = \mathsf{SIV.Enc}_K(N, A, M)$. The decryption algorithm of SIV scheme $\mathsf{SIV.Dec}$ takes (K, N, A, C, T) as input, and returns the corresponding plaintext M or the symbol $\perp$ indicating rejection. We write $M = \mathsf{SIV.Dec}_K(N, A, C, T)$ or $\perp = \mathsf{SIV.Dec}_K(N, A, C, T)$. They are define in Fig. 1.1.

Let $E : \{0,1\}^k \times \mathcal{I} \times \{0,1\}^n \to \{0,1\}^n$ be the underlying block cipher, where k is the key length, $\mathcal{I}$ is the domain separation space, and n is the block length. We instantiate $\mathcal{E}$ and $\mathcal{D}$ as in Fig. 1.2. This is an OFB mode of E. See the overall illustration in Fig. 1.4.

The definition of $\mathcal{F}$ is presented in Fig. 1.3. This is a variant of CBC-MAC, where N, A, and M are processed independently based on the domain separation.

Algorithm $\mathcal{F}_K(N, A, M)$

1. $S \leftarrow 0^n$
2. $(A[1], \ldots, A[a]) \xleftarrow{n} A$
3. **if** $|A[a]| < n$ **then** $d \leftarrow 1$ **else** $d \leftarrow 2$
4. $A[a] \leftarrow \mathrm{pad}_n(A[a])$
5. **for** $i = 1$ **to** a **do**
6. $\quad S \leftarrow S \oplus A[i]$
7. $\quad S \leftarrow E_K^0(S)$
8. $(M[1], \ldots, M[m]) \xleftarrow{n} M$
9. **for** $i = 1$ **to** $m - 1$ **do**
10. $\quad S \leftarrow S \oplus M[i]$
11. $\quad S \leftarrow E_K^d(S)$
12. **if** $|M[m]| < n/2$ **then**
13. $\quad S \leftarrow S \oplus (\mathrm{pad}_{n/2}(M[m]) \| N)$
14. $\quad T \leftarrow E_K^3(S)$
15. **if** $|M[m]| = n/2$ **then**
16. $\quad S \leftarrow S \oplus (M[m] \| N)$
17. $\quad T \leftarrow E_K^4(S)$
18. **if** $n/2 < |M[m]| < n$ **then**
19. $\quad S \leftarrow S \oplus (\mathrm{pad}_n(M[m]))$
20. $\quad S \leftarrow E_K^d(S)$
21. $\quad S \leftarrow S \oplus (0^{n/2} \| N)$
22. $\quad T \leftarrow E_K^5(S)$
23. **if** $|M[m]| = n$ **then**
24. $\quad S \leftarrow S \oplus M[m]$
25. $\quad S \leftarrow E_K^d(S)$
26. $\quad S \leftarrow S \oplus (0^{n/2} \| N)$
27. $\quad T \leftarrow E_K^6(S)$
28. **return** T

Figure 1.3: The definitions of $\mathcal{F}$.

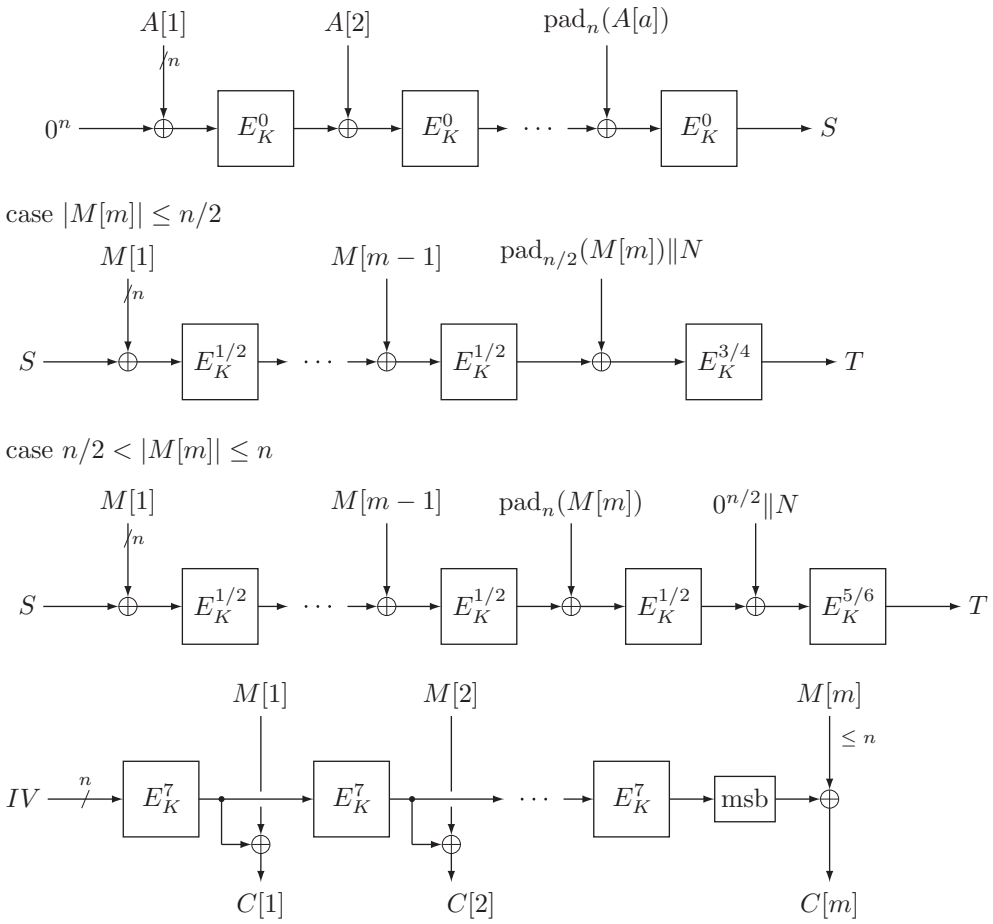


Figure 1.4: The overall structure of SIV scheme. Top: Process of AD A in $\mathcal{F}_K(N, A, M)$. 2nd: Process of a plaintext M in $\mathcal{F}_K(N, A, M)$ for the case $|M[m] \leq n/2$. 3rd: Process of M for the case $n/2 < |M[m] \leq n$. Bottom: $C = \mathcal{E}_K^{IV}(M)$. Note that $IV = T$.

1.1.3 Rijndael256

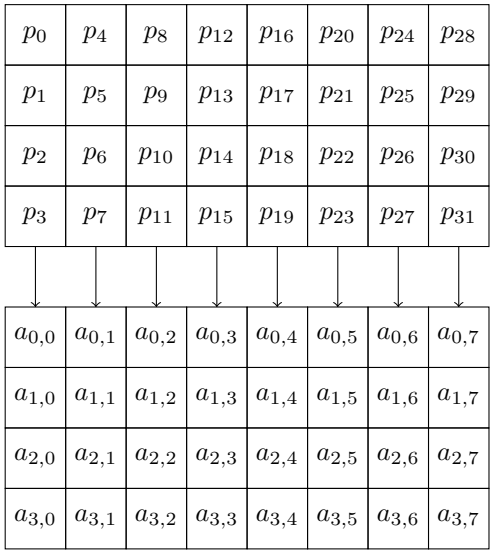
The Rijndael block cipher is a proposal for the Advanced Encryption Standard (AES). Rijndael was selected as the AES after narrowing the range of supported values for the block length and key length. As well-known, AES has a unique block length – 128 bits, and supports three key lengths – 128, 192, or 256 bits. Whereas, the original proposal – Rijndael, supports any independently specified block length b bits and key length k bits, such that b and k are multiple of 32, $128 \leq b \leq 256$, and $128 \leq k \leq 256$.

In the proposal of SIV-Rijndael256, we select Rijndael with fixed **block length 256** bits and fixed **key length 128** bits as our building block, and denote this primitive by Rijndael256. Next, we specify Rijndael256 in detail (for the full specification of Rijndael, please refer to [7]).

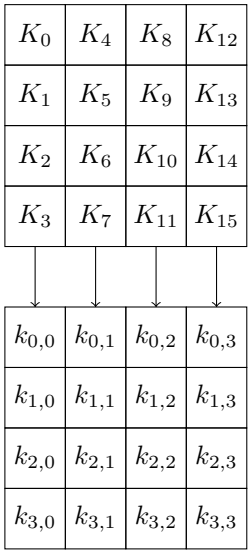
Rijndael256 is a key-alternating block cipher with block length 256 bits and key length 128 bits. It is composed of three algorithms – the encryption, the decryption, and the key schedule. In SIV-Rijndael256, only the encryption and the key schedule of Rijndael256 are used. Hence, we describe these two algorithms only. The encryption and the different transformations composing it operates on an intermediate result, called the *state*. The key schedule and its steps operate on an intermediate result, called the cipher key state. Both the state and the cipher key state can be pictured as a rectangular array of bytes, as illustrated in Figure 1.5. In which the number of columns in the state is denoted by Nb which equals 8, and the number of columns of the cipher key state is denoted by Nk which equals 4.

The encryption of Rijndael256 is of substitution-permutation-networks (SPN) structure. It consists of an initial key addition, denoted by AddRoundKey, followed by 13 applications of the transformation Round, and finally one application of FinalRound. Thus, the total number of rounds, denoted by Nr, is 14.

The round transformation Round is composed by the following four basic transformations, called *steps*.



(a) The mapping from plaintext to state



(b) The mapping from key to cipher key state

Figure 1.5: State and cipher key layout

- **SubBytes** (SB). The SubBytes step is the only non-linear transformation of the cipher. It consists of an substitution table (or S-box) applied to each byte of the state independently. The S-box, denoted by S_{RD}, is an 8-bit to 8-bit permutation and is obtained by taking the multiplicative inverse in GF(2^8) followed by applying an affine (over GF(2)) transformation. For the multiplicative inverse $x \mapsto x^{-1}$ in GF(2^8), multiplication is done modulo the irreducible binary polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$ (or '11B' in hexadecimal representation). The value 00 is mapped onto itself. The affine transformation and the tabular representation of $S_{RD}(xy)$ can be seen in Appendix A.1.1.

- **ShiftRows** (SR). The ShiftRows step is a byte transposition that cyclically shifts the rows of the state to the left. The shift offsets for row 0, 1, 2, 3 are denoted by C_0, C_1, C_2, C_3, which equal 0, 1, 3, 4 respectively. Then, the byte at position j in row i moves to position $(j - C_i) \mod 4$.

- **MixColumns** (MC). The MixColumns step is a linear (over GF(2)) permutation. It consists of a modular multiplication with a fixed polynomial $c(x)$ operating on each column of the state independently. The columns of the state are considered as polynomials over GF(2^8). They are multiplied modulo x^4+1 with the polynomial $c(x)$ which is given by $c(x) = \texttt{03}\cdot x^3 + \texttt{01}\cdot x^2 + \texttt{01}\cdot x + \texttt{02}$. This modular multiplication with $c(x)$ can be written as a matrix multiplication (denoted by $\mathbf{M} \times C$). Let $b(x) = c(x)\cdot a(x) \pmod{x^4+1}$. Then

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

- **AddRoundKey** (AK). The key addition, denoted by AddRoundKey, consists of bitwise XORing a round key to the state. A round key is denoted by ExpandedKey[i], $0 \le i \le$ Nr, with length equals the block length (256 bits). The array of 15 round keys ExpandedKey is derived from the cipher key by applying the key schedule.

The i-th ($1 \le i < 14$) round transformation Round(State, ExpandedKey[i]) can be written as (see Figure 1.6)

$$\mathsf{AddRoundKey} \circ \mathsf{MixColumns} \circ \mathsf{ShiftRows} \circ \mathsf{SubBytes}(\mathtt{State}, \mathsf{ExpandedKey}[i])$$

The last round transformation FinalRound(State, ExpandedKey[14]) can be written as

$$\mathsf{AddRoundKey} \circ \mathsf{ShiftRows} \circ \mathsf{SubBytes}(\mathtt{State}, \mathsf{ExpandedKey}[14]).$$

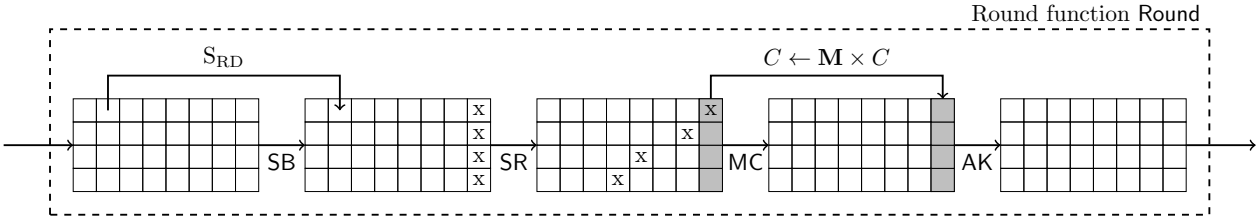


Figure 1.6: Graphical representation of the round transformation Round

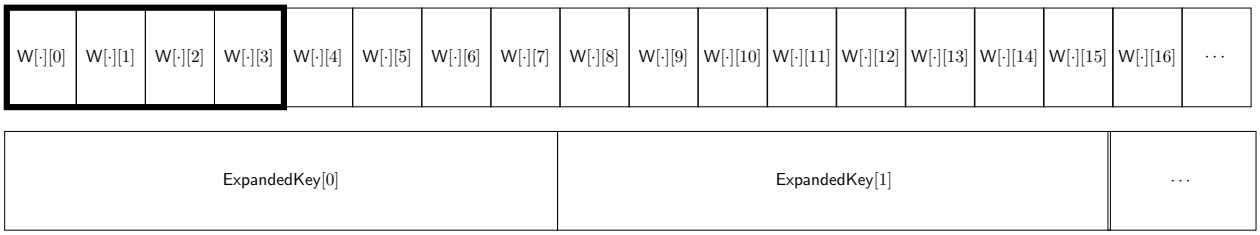


Figure 1.7: Key expansion and round key selection for Nb = 8 and Nk = 4.

The key schedule of Rijndael256 expands a cipher key (128 bits) into an expanded key array by operating the Key Expansion. From the expanded key array, the 15 round keys ExpandedKey (256 bits each) are selected by the Round Key Selection.

- **Key expansion**. The key expansion of Rijndael256 is the same as that in AES with 128 bits key, except for the total number of columns in the expanded key array. In Rijndael256, the expanded key array consists of 4 rows and $\mathrm{Nb} \cdot (\mathrm{Nr}+1)$ (*i.e.,* $8 \times 15 = 120$) columns, which is denoted by $\mathsf{W}[4][8 \times 15]$. The 128-bit cipher key is copied into the first four columns of the expanded key array, *i.e.,*

$$\mathsf{W}[i][j] = k_{i,j} \text{ for } 0 \le i \le 3 \text{ and } 0 \le j \le 3.$$

Then, $\mathsf{W}[i][j]$ for $4 \le j < 120$ and $0 \le i \le 3$ is computed as

$$\mathsf{W}[i][j] = \begin{cases} \mathsf{W}[i][j-4] \oplus \mathrm{S_{RD}}(\mathsf{W}[(i+1) \bmod 4][j-1]) \oplus \mathrm{RC}[j/4], & j \equiv 0 \bmod 4 \text{ and } i = 0 \\ \mathsf{W}[i][j-4] \oplus \mathrm{S_{RD}}(\mathsf{W}[(i+1) \bmod 4][j-1]), & j \equiv 0 \bmod 4 \\ \mathsf{W}[i][j] \oplus \mathsf{W}[i][j-1], & \text{otherwise.} \end{cases}$$

where, RC[·] is an array of bytes which are the round constants defined by the recursion rule in $\mathrm{GF}(2^8)$ with the irreducible polynomial $m(x)$ used to define $\mathrm{S_{RD}}$:

$$\mathrm{RC}[j] = \begin{cases} x^0 \text{ (\textit{i.e.,} 01)} & j = 0 \\ x^1 \text{ (\textit{i.e.,} 02)} & j = 1 \\ x \cdot \mathrm{RC}[j-1] = x^{j-1} & j \ge 2. \end{cases}$$

- **Round key selection**. The round key of the i-th round, denoted by $\mathsf{ExpandedKey}[i]$ is given by (and depicted by Figure 1.7)

$$\mathsf{ExpandedKey}[i] = \mathsf{W}[\cdot][8i] \| \mathsf{W}[\cdot][8i+1] \| \cdots \| \mathsf{W}[\cdot][8i+7],\ 0 \le i \le 14.$$

1.1.4 Add 3-bit Tweak into Rijndael256

In SIV-Rijndael256, multiple (explicitly, eight) independent instances of Rijndael256 are required. Thus, a 3-bit tweak is required to act as domain separator.

We propose to add a 3-bit tweak to the bytes at the second column in the state (*i.e.,* $a_{\cdot,1}$ shown in Figure 1.8) before the SubBytes step in each Round and FinalRound (when encoded by a byte, they are put

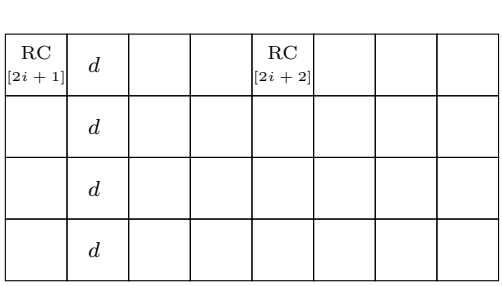


Figure 1.8: The positions of the bytes in the state to which the round constants and the 3-bit tweak d will be XOR-ed.

at the least significant bits). The position of the XOR-ed byte is chosen to avoid the interaction between the round constants which are equivalently XOR-ed to $a_{0,0}$ and $a_{0,4}$. The additional effect of adding these 3-bit tweak on implementation is negligible. The effect on security is also expected to be small. These can be viewed as following the Tweakey framework proposed in [15]. In [15], a family of tweakable block ciphers named Kiasu-BC was proposed. Kiasu-BC has a 128-bit internal state and 64-bit tweakey state. "It is exactly the AES cipher, except that the tweak value is XOR-ed to the two top rows of the internal state at every round after the addition of the subkeys (after the AddRoundKey operation)" [15]. In our case, considering that the number of controllable bit on tweak is quite small and it can be viewed as following the method of inserting tweak in Kiasu-BC, we believe the impact on security is limited. We will denote the instance of Rijndael256 inserted with 3-bit tweak d as Rijndael256^d, *i.e.,* Rijndael256^0, Rijndael256^1, $\cdots$, Rijndael256^7 respectively.

1.2 Specification of SIV-Rijndael256 Family

1.2.1 SIV-Rijndael256-AEAD Authenticated Encryption

The SIV-Rijndael256-AEAD family is the instance of the SIV scheme (specified in Sect. 1.1.2) with multiple instances of Rijndael256 (specified in Sect. 1.1.3 and Sect. 1.1.4) being the underlying block ciphers.

The SIV-Rijndael256-AEAD family consists of only one instance, with the parameter sizes:

- block size $n = 256$ bits,
- key size $k = 128$ bits,
- tag size $|T| = 256$ bits,
- nonce length $|N| = 128$ bits.

It supports the following:

- any bit length of associated data $|A| \geq 0$,
- any bit length of messages $|M| \geq 0$.

Due to the mode, decryption algorithm of the cipher is not necessary.

1.2.2 SIV-Rijndael256 Hash Function

SIV-Rijndael256-Hash adopts the Sponge-like construction (as shown in Fig. 1.9 and 1.10). The difference with Sponge is that the initial absorbing rate and the squeezing rate are larger than the internal absorbing rate. Specifically, in SIV-Rijndael256-Hash, both of the initial absorbing bitrate and the squeezing bitrate are r_0, whereas, the internal absorbing bitrate is r_1 and output digest size is `ds`, *i.e.*,

$$r_0 = 128, \qquad r_1 = 32, \qquad c_0 = 128, \qquad c_1 = 224, \qquad \texttt{ds} = 256.$$

The underlying permutation in SIV-Rijndael256-Hash is the Rijndael256 with the master key set to the constant 0, and we keep the additional input d, and denote the resulted permutation as $f^{[d]}$. $d = 0$ is used for all

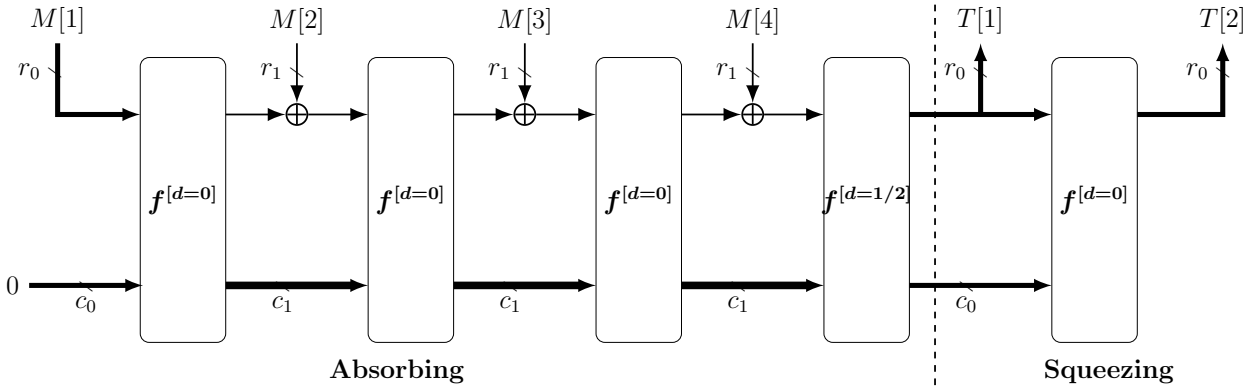


Figure 1.9: SIV-Rijndael256-Hash adopts Sponge-like construction, with initial absorbing bitrate r_0, internal absorbing bitrate r_1 and squeezing bitrate r_0.

Algorithm SpongeHash$[f^{[0/1/2]}, \text{pad}, r_0, r_1, \texttt{ds}](M)$

Absorption Phase:

1. $(M[1], \ldots, M[m]) \xleftarrow{r_0, r_1} M$
2. **if** $|M| \leq r_0$ **then**
3. $\quad d \leftarrow (|M[m]| = r_0)?1 : 2$
4. $\quad M[m] \leftarrow \text{pad}_{r_0}(M[m])$
5. **else**
6. $\quad d \leftarrow (|M[m]| = r_1)?1 : 2$
7. $\quad M[m] \leftarrow \text{pad}_{r_1}(M[m])$
8. $S \leftarrow 0$
9. **for** $i = 1$ **to** $m - 1$
10. $\quad S \leftarrow S \oplus (M[i] \mid\mid 0^{|S|-|M[i]|})$
11. $\quad S = f^{[0]}(S)$
12. $S \leftarrow f^{[d]}(S \oplus (M[m] \mid\mid 0^{|S|-|M[m]|}))$

Squeezing Phase:

13. $T = \text{msb}_{r_0}(S)$
14. **for** $i = 1$ **to** $\lceil \texttt{ds}/r_0 \rceil - 1$
15. $\quad S \leftarrow f^{[0]}(S)$
16. $\quad T = T \mid\mid \text{msb}_{r_0}(S)$
17. **return** $\text{msb}_{\texttt{ds}}(T)$

Figure 1.10: The definition of our modified Sponge construction.

other places, rather than the last call in the absorption phase. The padding rule follows the same as in the AEAD, i.e., when the last message is of full block (128 bits if $|M| \leq 128$, 32 bits otherwise), no padding is necessary otherwise a bit string of 10^* is padded, and the corresponding d is defined as 1 for full block, and 2 for non-full block.

Chapter 2

Security

2.1 Summary of Expected Security Strength

Attack Model	Time Complexity	Data Complexity
Key Recovery	128 bits	128 bits
Forgery	128 bits	128 bits

Table 2.1: The security claims of SIV-Rijndael256-AEAD.

collision	second-preimage	preimage
112 bits	112 bits	128 bits

Table 2.2: The security claims of SIV-Rijndael256-Hash.

2.2 Known Cryptanalytic Attacks on Rijndael with Block Length 256-bit

For SIV-Rijndael256-AEAD, we consider secrete-key attacks on Rijndael. According to the published results on cryptanalysis of large-block Rijndael, the impossible differential attacks and integral attacks are the most threatening attacks. This situation is similar with the situation on cryptanalysis of AES with 128-bit key. Table 2.3 lists current best attacks on Rijndael with 256-bit block and 256-bit key. These attacks shown in Table 2.3 are under the single-key model, which indicate that when using a 128-bit key the data complexity should be upper bounded by 2^{128}. For attacks on more than 7 rounds, the data complexity are all close to or larger than 2^{128}. For general estimation on resistance against the differential attack, we can refer to [38], in which authors provided updated bound on the number of active S-boxes for Rijndael with 256-bit block, which can be seen in Table 2.4.

According to the known best attacks, we estimate that the best attacks on Rijndael with 256-bit block and 128-bit key, *i.e.*, Rijndael256 used in SIV-Rijndael256-AEAD, cannot attack more than 10 rounds out of the 14 rounds with data complexity lower than 2^{128}.

In SIV-Rijndael256-Hash, we consider known-key attacks on Rijndael. In [39], known-key attacks are presented which work on 8-round Rijndael with 192-bit block and 9-round Rijndael with 256-bit block. In the attack on Rijndael with 256-bit block, one can find a pair of values which has 16-byte (128-bit) differences in both of input and output with 2^{48} computations and 2^{32} amount of memory, while an ideal case requires 2^{64}. No similar property is known on 10-round Rijndael with 256-bit block. Besides, for Rijndael with 192-bit block, better ShiftRows parameters are recommended for resisting truncated differential attack. However, for Rijndael with 256-bit block, no better ShiftRows parameters than the original one (*i.e.*, $(0, 1, 3, 4)$) was recommended. This conclusion on the optimality of the ShiftRows parameters in Rijndael256 is also supported

Cipher	NR	Data (CP)	Time (Enc)	Memory (Bytes)	Attack type	Source
Rijndael-256-256	7	$2^{130.5}$	2^{141}		Multiset	[16]
	7	2^{153}	2^{182}	2^{122}	IDA	[31]
	7	$2^{93.2}$	$2^{113.2}$	2^{61}	IDA	[50]
	7	6×2^{32}	2^{44}		Integral	[12]
	8	$\approx 2^{128}$	$\approx 2^{128}$		Integral	[12]
	9	$\approx 2^{128}$	2^{204}		Integral	[12]
	9	$2^{244.3}$	$2^{208.8}$	2^{189}	IDA	[50]
	9	$2^{132.5}$	$2^{174.5}$		Integral	[22]
	9	$2^{237.3}$	$2^{159.1}$	$2^{115.3}$	IDA	[45]
	9	$2^{245.3}$	$2^{127.1}$	$2^{90.9}$	IDA	[45]
	10	$2^{244.2}$	$2^{253.9}$	$2^{186.8}$	IDA	[45]
	10	$2^{244.4}$	$2^{240.1}$	$2^{186.4}$	IDA	[23]

IDA: impossible differential attack

Table 2.3: Summary of known attacks on Rijndael-256-256

r	3	4	5	6	7	8	9	10	12	14	20
MNDAS	9	25	41	50	58	65	74	85	105	120	175
MNDAS/r	3	6.25	8.2	8.33	8.28	8.12	8.22	8.5	8.75	8.57	8.75

MNDAS: minimum number of differentially active S-boxes

Table 2.4: Lower bound of the number of active S-box for r rounds of Rijndael-256 [38]

by authors of [38] after trying to find better options among ten possible parameters in terms of lower bound for the minimum number of active S-boxes.

There are many more other cryptanalysis results [23, 45, 39, 22, 46, 29, 12, 50, 31, 18, 47, 34, 41, 4, 3, 10, 32, 19, 6, 9] against classic security notations, and side-channels [42, 13, 43, 49].

In summary, Rijndael-256-256 remains secure as a primitive after twenty years' cryptanalysis by the community, and Rijndael256 (Rijndael-256-128) as a special instance of Rijndael-256-256 is supposed to remain secure with high security margins.

Chapter 3

Design Rationale

Our design goals are summarized as follows.

- The AEAD scheme that is suitable for use lightweight applications.
- Strong security guarantee, based on well analyzed and trusted underlying primitive and well established mode of operation.
- Address misuse cases of nonce-repetition and release of unverified plaintexts.
- Avoid the use of decryption algorithm of the underlying block cipher, for AEAD decryption.
- The hash function can be easily defined by reusing the primitive from the AEAD scheme, under a well understood mode, slightly modified for efficiently processing short messages.
- All necessary modifications are kept to be minimum, without affecting the security and lightweightness in hardware/software.

3.1 Choice of SIV

Above goals in mind, we decided to use the SIV mode [36] as our mode of operation. The mode enjoys the provable security in the strong sense of nonce-misuse case, and it also has the provable security in terms of release of unverified plaintexts [1]. The combined OFB mode also enables decryption of the AEAD without the use of decryption algorithm of the underlying block cipher, which saves the gates required in the hardware implementations and reduces the code size or ROM in software implementations.

Our choice of CBC MAC is to build our scheme on an established standard scheme, which generates the tag from the associated data and message with well understood proven security. We use OFB mode for its solid provable security guarantee and its small footprint in implementations. Overall, the mode removes the use of decryption algorithm, and requires such a small amount of gates to implement that the overall amount of gates required by the AEAD design is almost the same as that by the underlying Rijndael256 block cipher.

The security bound is the standard birthday bound of the form $O(\sigma^2/2^n)$, where σ denotes the total number of blocks in the security game. With the application for hashing in mind, we adopt a block size of $n = 256$, which gives a solid security bound for any lightweight applications.

3.2 Choice of Rijndael

The Rijndael family is a long-standing and well-studied design. From this family, three members are selected as The Advanced Encryption Standard (AES) by the U.S. National Institute of Standards and Technology (NIST) in 2001. In the design of SIV-Rijndael256, another member Rijndael256 different from the three members in AES is selected, in the consideration of minimum dimensions to fit the minimum acceptability requirements on the AEAD algorithm and Hash function. The block length (256 bits) is selected in the consideration of the birthday bound (2^{128}) on the security of Hash function for collision-resistance. The key length (128 bits) is selected considering that any cryptanalytic attack on the AEAD algorithm shall require more than 2^{112} computations on a classical computer.

Rijndael256 should share both security-related and implementation-related properties with members of AES. The studies on both security and implementations aspects of AES have been going on for almost two decades. As a standard worldwide used, AES has become the most understood and deployed cryptographic scheme. It turns out that members of AES are strong enough to resist practical attacks. In addition, the performances both on hardware and software of AES are good. More than that, AES can be viewed as light-weight (1560 GEs) primitive using bit-slicing technique to implement [14]. Accordingly, other members of Rijndael, particularly Rijndael256, have the same advantages as that of members of AES.

Chapter 4

Performance

4.1 Hardware Performance

For hardware, we expect the area cost by SIV-Rijndael256 to be very small when using bit-slicing serial implementation methods. The mode SIV costs little on top of the cost of the underlying block cipher Rijndael256. Hence, to estimate the hardware implementation cost of SIV-Rijndael256, we focus on estimating the hardware implementation cost of Rijndael256.

We estimate the hardware performance of Rijndael256 with area minimization as the optimization target. The current record of minimized area of AES-128 is kept by the bit-serial implementations provided by Jean *et al.* [14]. Using the results in [14], we estimate area and latency of Rijndael256.

Compared with implementations of AES-128, the additional area cost for implementations of Rijndael256 comes from the cost for storing additional 128-bit state bits, the cost for storing the 3 bits (less than 8 3-bit values), and the cost for XOR-ing with 3-bit domain separators. Among the 128 additional state bits, 12 requires to be stored in scan flip-flops and 116 can be stored in regular flip-flops following methods in [14]. The domain separators can be stored using 8×3 regular flip-flops. Table 4.1 lists more detailed estimations on additional cost. Based on Table 4.1 and 4.2 and the results of AES-128 in [14], we get the estimation for Rijndael256 which are summarized in Table 4.3.

For latency of Rijndael256, the additional cycle-cost on top of that of AES-128 comes from the fact that Rijndael256 has more rounds and in each round double number of bits need to be updated. Note that, selecting and XOR-ing bits of domain separators can be implemented in the same clock cycles for AddRoundKey and SubBytes, thus cost no additional cycles. Thus, to estimate latency of Rijndael256, we use the following formula:

$$2 \times (13 \times Cycles_{\mathsf{Round}} + Cycles_{\mathsf{FinalRound}}),$$

where $Cycles_{\texttt{round}}$ is the clock cycles took by one complete round of AES, $Cycles_{\mathsf{FinalRound}}$ is that took by the last round and AddRoundKey, 13 is the number of complete round in Rijndael256, where $Cycles_{\texttt{round}}$ and $Cycles_{\mathsf{FinalRound}}$ for each implementation method is listed in Table 4.3 (column 8 for AES).

From Table 4.3, the hardware performance of Rijndael256 can be very small.

Besides, there are many previous studies on performance of Rijndael, *e.g.*, [40, 28, 25, 37, 11, 27, 21, 24, 44].

	UMC 180 GE	UMC 130 GE	UMC 90 GE	Ngate 45 GE	IBM 130 GE
1-bit DFF	4.67	5.00	4.25	5.67	4.25
1-bit Scan FF	6.00	6.25	5.75	7.67	5.50
1-bit XOR	2.67	2.75	2.50	2.00	2.00
2-to-1 MUX	2.33	2.25	2.25	2.33	2.25

Table 4.1: The (estimated) cost of regular flip-flops, 2-input XOR gates, and 2-to-1 Multiplexers in different libraries.

	UMC 180 GE	UMC 130 GE	UMC 90 GE	Ngate 45 GE	IBM 130 GE
12 bit of the additional 128 bit state	72	75	69	92.04	66
116 bit of the additional 128 bit state	541.72	580	493	657.72	493
Total cost of the additional 128 bit state	613.72	655	562	749.76	559
8 3-bit domain seperator	112.08	120	102	136.08	102
3-bit XOR and Multiplexer	15	15	14.25	12.99	12.75
Total cost of adding domain seperator	127.08	135	116.25	149.07	114.75

Table 4.2: The detail estimation on additional cost

Cipher	data path δ bits	UMC 180 GEs	UMC 130 GEs	UMC 90 GEs	Ngate 45 GEs	IBM 130 GEs	Latency Cycles	Ref.
AES-128	1	1727	1902	1596	1982	1560	$1776/\ (9 \times 168 + 264)$	[14]
	2	1796	1992	1667	2054	1625	$888/\ (9 \times 84 + 132)$	[14]
	4	1920	2168	1784	2146	1731	$520/\ (9 \times 50 + 70)$	[14]
	8	2112	2360	1968	2337	1912	$282/\ (9 \times 27 + 39)$	[14]
	8	2400	3574	2292	2768	2182	$226/\ (10 \times 21 + 16)$	[30]
Rijndael256	1	2468	2692	2274	2881	2234	$4896/\ 2 \times (13 \times 168 + 264)$	[14]
	2	2537	2782	2345	2953	2299	$2448/\ 2 \times (13 \times 84 + 132)$	[14]
	4	2661	2958	2462	3045	2405	$1440/\ 2 \times (13 \times 50 + 70)$	[14]
	8	2853	3150	2646	3236	2586	$780\ /\ 2 \times (13 \times 27 + 39)$	[14]
	8	3141	4364	2970	3667	2856	$620\ /\ 2 \times (14 \times 21 + 16)$	[30]

Table 4.3: Estimations on hardware implementation area and latency of Rijndael256 based on state-of-the-art results of AES-128

4.2 Software Performance

4.2.1 High-end CPU

We can also estimate the software performance on high-end CPU of Rijndael256 on the basis of best results of AES software, which can be found in [26, 48, 5, 20, 17, 33] etc. From those result, the software performance of AES-128 can be $8.5 \sim 15$ cycles per byte using table-based methods, $5 \sim 8$ cycles per byte using bitsliced methods, and $0.6 \sim 1.5$ cycles per byte using AES-NI [33]. We expect Rijndael256 can perform better than AES-128 when both implemented using table-based methods or bitsliced methods, because Rijndael256 encrypts messages of doubled length with less than double rounds. Thus, we can use $(14/(10 \times 2)) \times C_{\mathsf{AES}-128}$ to estimate Rijndael256 software performance, which can be $6 \sim 10$ cycles per byte using Table-based methods, $3.5 \sim 5.5$ cycles per byte using bitsliced methods.

4.2.2 Micro-controllers

For micro-controllers, we also expect the software implementation costs of SIV-Rijndael256 be small in terms of code size (ROM) and RAM. The base is on the available results both of AES-128 [8] and of primitives using Rijndael256 as the underlying component [2].

According to [8], for AES-128 encryption with key schedule included, in AVR devices, the code size and RAM requirement are 1026 bytes and 26 bytes respectively; in MSP devices they are 1022 bytes and 36 respectively; in ARM devices, they are 1208 bytes and 84 bytes respectively.

According to [2], in Atmel AVR devices, an implementation of the hash function – Shrimpton-Stam construction based on Rijndael-256/256 with 256-bit digest – has code size 734 bytes and RAM 168 bytes. An implementation of the Davies-Meyer construction based on Rijndael-256/256 with 256-bit digest, has code

size 696 bytes and RAM 136 bytes.

Considering the implementation of Rijndael256 on those devices can share the same implementation techniques and implementation merits of AES-128, and considering that both the SIV mode and the Sponge-like construction used in SIV-Rijndael256-AEAD and SIV-Rijndael256-Hash cost little on top of the underlying Rijndael256, we expect the code size (ROM) and RAM requirements of software implementation of SIV-Rijndael256 in micro-controllers to be small.

Chapter 5

References

5.1 Reference/Third-party Analysis on Rijndael

There are many studies on members of AES, which provide references for the security analysis of the member Rijndael256. Besides, there are also some studies focus on large-block Rijndael covering Rijndael-256-256 which has 256-bit block and 256-bit key. In the following, we listed published works on large-block Rijndael for further reference:

Classical Cryptanalysis:

1. Ya Liu, Yifan Shi, Dawu Gu, Bo Dai, Fengyu Zhao, Wei Li, Zhiqiang Liu, and Zhiqiang Zeng. Improved impossible differential cryptanalysis of large-block rijndael. *Science China Information Sciences*, 62(3):32101, 2019

2. Mahdi Sajadieh, Arash Mirzaei, Hamid Mala, and Vincent Rijmen. A new counting method to bound the number of active s-boxes in rijndael and 3d. *Des. Codes Cryptography*, 83(2):327–343, 2017

3. Qingju Wang, Dawu Gu, Vincent Rijmen, Ya Liu, Jiazhe Chen, and Andrey Bogdanov. Improved impossible differential attacks on large-block Rijndael. In Taekyoung Kwon, Mun-Kyu Lee, and Daesung Kwon, editors, *ICISC 12: 15th International Conference on Information Security and Cryptology*, volume 7839 of *LNCS*, pages 126–140, Seoul, Korea, November 28–30, 2013. Springer, Heidelberg, Germany

4. Yu Sasaki. Known-key attacks on Rijndael with large blocks and strengthening ShiftRow parameter. In Isao Echizen, Noboru Kunihiro, and Ryôichi Sasaki, editors, *IWSEC 10: 5th International Workshop on Security, Advances in Information and Computer Security*, volume 6434 of *LNCS*, pages 301–315, Kobe, Japan, November 22–24, 2010. Springer, Heidelberg, Germany

5. Yan-Jun Li and Wen-Ling Wu. Improved integral attacks on rijndael. *Journal of Information Science & Engineering*, 27(6), 2011

6. Yuechuan Wei, Bing Sun, and Chao Li. New integral distinguisher for Rijndael-256. Cryptology ePrint Archive, Report 2009/559, 2009. `http://eprint.iacr.org/2009/559`

7. Marine Minier, Raphael C.-W. Phan, and Benjamin Pousse. Distinguishers for ciphers and known key attack against Rijndael with large blocks. In Bart Preneel, editor, *AFRICACRYPT 09: 2nd International Conference on Cryptology in Africa*, volume 5580 of *LNCS*, pages 60–76, Gammarth, Tunisia, June 21–25, 2009. Springer, Heidelberg, Germany

8. Jorge Nakahara Jr., Daniel Santana de Freitas, and Raphael Chung-Wei Phan. New multiset attacks on rijndael with large blocks. In Ed Dawson and Serge Vaudenay, editors, *Progress in Cryptology - Mycrypt 2005, First International Conference on Cryptology in Malaysia, Kuala Lumpur, Malaysia, September 28-30, 2005, Proceedings*, volume 3715 of *Lecture Notes in Computer Science*, pages 277–295. Springer, 2005

9. Samuel Galice and Marine Minier. Improving integral attacks against Rijndael-256 up to 9 rounds. In Serge Vaudenay, editor, *AFRICACRYPT 08: 1st International Conference on Cryptology in Africa*, volume 5023 of *LNCS*, pages 1–15, Casablanca, Morocco, June 11–14, 2008. Springer, Heidelberg, Germany

10. Lei Zhang, Wenling Wu, Je Hong Park, Bon Wook Koo, and Yongjin Yeom. Improved impossible differential attacks on large-block Rijndael. In Tzong-Chen Wu, Chin-Laung Lei, Vincent Rijmen, and Der-Tsai Lee, editors, *ISC 2008: 11th International Conference on Information Security*, volume 5222 of *LNCS*, pages 298–315, Taipei, Taiwan, September 15–18, 2008. Springer, Heidelberg, Germany

11. Jorge Nakahara Jr. and Ivan Carlos Pavão. Impossible-differential attacks on large-block Rijndael. In Juan A. Garay, Arjen K. Lenstra, Masahiro Mambo, and René Peralta, editors, *ISC 2007: 10th International Conference on Information Security*, volume 4779 of *LNCS*, pages 104–117, Valparaíso, Chile, October 9–12, 2007. Springer, Heidelberg, Germany

12. Liam Keliher, Henk Meijer, and Stafford Tavares. Completion of computation of improved upper bound on the maximum average linear hull probability for Rijndael. Cryptology ePrint Archive, Report 2004/074, 2004. `http://eprint.iacr.org/2004/074`

13. Ralph Wernsdorf. The round functions of RIJNDAEL generate the alternating group. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encryption – FSE 2002*, volume 2365 of *LNCS*, pages 143–148, Leuven, Belgium, February 4–6, 2002. Springer, Heidelberg, Germany

14. Sangwoo Park, Soo Hak Sung, Seongtaek Chee, E-Joong Yoon, and Jongin Lim. On the security of Rijndael-like structures against differential and linear cryptanalysis. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 176–191, Queenstown, New Zealand, December 1–5, 2002. Springer, Heidelberg, Germany

15. Beomsik Song and Jennifer Seberry. Consistent differential patterns of Rijndael. In Pil Joong Lee and Chae Hoon Lim, editors, *ICISC 02: 5th International Conference on Information Security and Cryptology*, volume 2587 of *LNCS*, pages 149–163, Seoul, Korea, November 28–29, 2003. Springer, Heidelberg, Germany

16. Elad Barkan and Eli Biham. In how many ways can you write Rijndael? Cryptology ePrint Archive, Report 2002/157, 2002. `http://eprint.iacr.org/2002/157`

17. Elad Barkan and Eli Biham. The book of Rijndaels. Cryptology ePrint Archive, Report 2002/158, 2002. `http://eprint.iacr.org/2002/158`

18. Niels Ferguson, Richard Schroeppel, and Doug Whiting. A simple algebraic representation of Rijndael. In Serge Vaudenay and Amr M. Youssef, editors, *SAC 2001: 8th Annual International Workshop on Selected Areas in Cryptography*, volume 2259 of *LNCS*, pages 103–111, Toronto, Ontario, Canada, August 16–17, 2001. Springer, Heidelberg, Germany

19. Kenji Ohkuma, Hideo Shimizu, Fumihiko Sano, and Shinichi Kawamura. Security assessment of Hierocrypt and Rijndael against the differential and linear cryptanalysis (extended abstract). Cryptology ePrint Archive, Report 2001/070, 2001. `http://eprint.iacr.org/2001/070`

20. Liam Keliher, Henk Meijer, and Stafford E. Tavares. Improving the upper bound on the maximum average linear hull probability for Rijndael. In Serge Vaudenay and Amr M. Youssef, editors, *SAC 2001: 8th Annual International Workshop on Selected Areas in Cryptography*, volume 2259 of *LNCS*, pages 112–128, Toronto, Ontario, Canada, August 16–17, 2001. Springer, Heidelberg, Germany

21. Jung Hee Cheon, MunJu Kim, Kwangjo Kim, Jung-Yeun Lee, and SungWoo Kang. Improved impossible differential cryptanalysis of Rijndael and Crypton. In Kwangjo Kim, editor, *ICISC 01: 4th International Conference on Information Security and Cryptology*, volume 2288 of *LNCS*, pages 39–49, Seoul, Korea, December 6–7, 2002. Springer, Heidelberg, Germany

Side-Channel Attacks:

1. Merrielle Spain and Mayank Varia. Diversity within the rijndael design principles for resistance to differential power analysis. In Sara Foresti and Giuseppe Persiano, editors, *CANS 16: 15th International Conference on Cryptology and Network Security*, volume 10052 of *LNCS*, pages 71–87, Milan, Italy, November 14–16, 2016. Springer, Heidelberg, Germany

2. Catherine H. Gebotys, Simon Ho, and C. C. Tiu. EM analysis of Rijndael and ECC on a wireless Java-based PDA. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, volume 3659 of *LNCS*, pages 250–264, Edinburgh, UK, August 29 – September 1, 2005. Springer, Heidelberg, Germany

3. François-Xavier Standaert, Siddika Berna Örs, and Bart Preneel. Power analysis of an FPGA: Implementation of Rijndael: Is pipelining a DPA countermeasure? In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *LNCS*, pages 30–44, Cambridge, Massachusetts, USA, August 11–13, 2004. Springer, Heidelberg, Germany

4. Sung-Ming Yen. Amplified differential power cryptanalysis on Rijndael implementations with exponentially fewer power traces. In Reihaneh Safavi-Naini and Jennifer Seberry, editors, *ACISP 03: 8th Australasian Conference on Information Security and Privacy*, volume 2727 of *LNCS*, pages 106–117, Wollongong, NSW, Australia, July 9–11, 2003. Springer, Heidelberg, Germany

Acknowledgements

The submitters would like to thank Kazuhiko Minematsu of NEC Corporation for various discussions in the early design stage of our submission.

Bibliography

[1] Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. How to securely release unverified plaintext in authenticated encryption. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 105–125, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg, Germany.

[2] Josep Balasch, Baris Ege, Thomas Eisenbarth, Benoit Gérard, Zheng Gong, Tim Güneysu, Stefan Heyse, Stéphanie Kerckhof, François Koeune, Thomas Plos, Thomas Pöppelmann, Francesco Regazzoni, François-Xavier Standaert, Gilles Van Assche, Ronny Van Keer, Loïc van Oldeneel tot Oldenzeel, and Ingo von Maurich. Compact implementation and performance evaluation of hash functions in ATtiny devices. Cryptology ePrint Archive, Report 2012/507, 2012. `http://eprint.iacr.org/2012/507`.

[3] Elad Barkan and Eli Biham. The book of Rijndaels. Cryptology ePrint Archive, Report 2002/158, 2002. `http://eprint.iacr.org/2002/158`.

[4] Elad Barkan and Eli Biham. In how many ways can you write Rijndael? Cryptology ePrint Archive, Report 2002/157, 2002. `http://eprint.iacr.org/2002/157`.

[5] Daniel J. Bernstein and Peter Schwabe. New AES software speed records. In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, *Progress in Cryptology - INDOCRYPT 2008: 9th International Conference in Cryptology in India*, volume 5365 of *LNCS*, pages 322–336, Kharagpur, India, December 14–17, 2008. Springer, Heidelberg, Germany.

[6] Jung Hee Cheon, MunJu Kim, Kwangjo Kim, Jung-Yeun Lee, and SungWoo Kang. Improved impossible differential cryptanalysis of Rijndael and Crypton. In Kwangjo Kim, editor, *ICISC 01: 4th International Conference on Information Security and Cryptology*, volume 2288 of *LNCS*, pages 39–49, Seoul, Korea, December 6–7, 2002. Springer, Heidelberg, Germany.

[7] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[8] Daniel Dinu, Yann Le Corre, Dmitry Khovratovich, Léo Perrin, Johann Großschädl, and Alex Biryukov. Triathlon of lightweight block ciphers for the internet of things. Cryptology ePrint Archive, Report 2015/209, 2015. `http://eprint.iacr.org/2015/209`.

[9] Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Michael Stay, David Wagner, and Doug Whiting. Improved cryptanalysis of Rijndael. In Bruce Schneier, editor, *Fast Software Encryption – FSE 2000*, volume 1978 of *LNCS*, pages 213–230, New York, NY, USA, April 10–12, 2001. Springer, Heidelberg, Germany.

[10] Niels Ferguson, Richard Schroeppel, and Doug Whiting. A simple algebraic representation of Rijndael. In Serge Vaudenay and Amr M. Youssef, editors, *SAC 2001: 8th Annual International Workshop on Selected Areas in Cryptography*, volume 2259 of *LNCS*, pages 103–111, Toronto, Ontario, Canada, August 16–17, 2001. Springer, Heidelberg, Germany.

[11] Viktor Fischer and Milos Drutarovský. Two methods of Rijndael implementation in reconfigurable hardware. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *LNCS*, pages 77–92, Paris, France, May 14–16, 2001. Springer, Heidelberg, Germany.

[12] Samuel Galice and Marine Minier. Improving integral attacks against Rijndael-256 up to 9 rounds. In Serge Vaudenay, editor, *AFRICACRYPT 08: 1st International Conference on Cryptology in Africa*, volume 5023 of *LNCS*, pages 1–15, Casablanca, Morocco, June 11–14, 2008. Springer, Heidelberg, Germany.

[13] Catherine H. Gebotys, Simon Ho, and C. C. Tiu. EM analysis of Rijndael and ECC on a wireless Java-based PDA. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, volume 3659 of *LNCS*, pages 250–264, Edinburgh, UK, August 29 – September 1, 2005. Springer, Heidelberg, Germany.

[14] Jérémy Jean, Amir Moradi, Thomas Peyrin, and Pascal Sasdrich. Bit-sliding: A generic technique for bit-serial implementations of SPN-based primitives - applications to AES, PRESENT and SKINNY. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems – CHES 2017*, volume 10529 of *LNCS*, pages 687–707, Taipei, Taiwan, September 25–28, 2017. Springer, Heidelberg, Germany.

[15] Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Tweaks and keys for block ciphers: The TWEAKEY framework. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 274–288, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg, Germany.

[16] Jorge Nakahara Jr., Daniel Santana de Freitas, and Raphael Chung-Wei Phan. New multiset attacks on rijndael with large blocks. In Ed Dawson and Serge Vaudenay, editors, *Progress in Cryptology - Mycrypt 2005, First International Conference on Cryptology in Malaysia, Kuala Lumpur, Malaysia, September 28-30, 2005, Proceedings*, volume 3715 of *Lecture Notes in Computer Science*, pages 277–295. Springer, 2005.

[17] Emilia Kasper and Peter Schwabe. Faster and timing-attack resistant AES-GCM. Cryptology ePrint Archive, Report 2009/129, 2009. `http://eprint.iacr.org/2009/129`.

[18] Liam Keliher, Henk Meijer, and Stafford Tavares. Completion of computation of improved upper bound on the maximum average linear hull probability for Rijndael. Cryptology ePrint Archive, Report 2004/074, 2004. `http://eprint.iacr.org/2004/074`.

[19] Liam Keliher, Henk Meijer, and Stafford E. Tavares. Improving the upper bound on the maximum average linear hull probability for Rijndael. In Serge Vaudenay and Amr M. Youssef, editors, *SAC 2001: 8th Annual International Workshop on Selected Areas in Cryptography*, volume 2259 of *LNCS*, pages 112–128, Toronto, Ontario, Canada, August 16–17, 2001. Springer, Heidelberg, Germany.

[20] Robert Könighofer. A fast and cache-timing resistant implementation of the AES. In Tal Malkin, editor, *Topics in Cryptology – CT-RSA 2008*, volume 4964 of *LNCS*, pages 187–202, San Francisco, CA, USA, April 7–11, 2008. Springer, Heidelberg, Germany.

[21] Henry Kuo and Ingrid Verbauwhede. Architectural optimization for a 1.82Gbits/sec VLSI implementation of the AES Rijndael algorithm. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *LNCS*, pages 51–64, Paris, France, May 14–16, 2001. Springer, Heidelberg, Germany.

[22] Yan-Jun Li and Wen-Ling Wu. Improved integral attacks on rijndael. *Journal of Information Science & Engineering*, 27(6), 2011.

[23] Ya Liu, Yifan Shi, Dawu Gu, Bo Dai, Fengyu Zhao, Wei Li, Zhiqiang Liu, and Zhiqiang Zeng. Improved impossible differential cryptanalysis of large-block rijndael. *Science China Information Sciences*, 62(3):32101, 2019.

[24] A. K. Lutz, J. Treichler, Frank K. Gürkaynak, Hubert Kaeslin, G. Basler, Antonia Erni, S. Reichmuth, P. Rommens, Stephan Oetiker, and Wolfgang Fichtner. 2Gbit/s hardware realizations of RIJNDAEL and SERPENT: A comparative analysis. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *LNCS*, pages 144–158, Redwood Shores, CA, USA, August 13–15, 2003. Springer, Heidelberg, Germany.

[25] Massoud Masoumi, Farshid Raissi, and Mahmoud Ahmadian. NanoCMOS-molecular realization of Rijndael. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems – CHES 2006*, volume 4249 of *LNCS*, pages 285–297, Yokohama, Japan, October 10–13, 2006. Springer, Heidelberg, Germany.

[26] Mitsuru Matsui and Junko Nakajima. On the power of bitslice implementation on intel core2 processor. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES 2007*, volume 4727 of *LNCS*, pages 121–134, Vienna, Austria, September 10–13, 2007. Springer, Heidelberg, Germany.

[27] Máire McLoone and John V. McCanny. High performance single-chip FPGA Rijndael algorithm implementations. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *LNCS*, pages 65–76, Paris, France, May 14–16, 2001. Springer, Heidelberg, Germany.

[28] Nele Mentens, Lejla Batina, Bart Preneel, and Ingrid Verbauwhede. A systematic evaluation of compact hardware implementations for the Rijndael S-box. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *LNCS*, pages 323–333, San Francisco, CA, USA, February 14–18, 2005. Springer, Heidelberg, Germany.

[29] Marine Minier, Raphael C.-W. Phan, and Benjamin Pousse. Distinguishers for ciphers and known key attack against Rijndael with large blocks. In Bart Preneel, editor, *AFRICACRYPT 09: 2nd International Conference on Cryptology in Africa*, volume 5580 of *LNCS*, pages 60–76, Gammarth, Tunisia, June 21–25, 2009. Springer, Heidelberg, Germany.

[30] Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. Pushing the limits: A very compact and a threshold implementation of AES. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 69–88, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany.

[31] Jorge Nakahara Jr. and Ivan Carlos Pavão. Impossible-differential attacks on large-block Rijndael. In Juan A. Garay, Arjen K. Lenstra, Masahiro Mambo, and René Peralta, editors, *ISC 2007: 10th International Conference on Information Security*, volume 4779 of *LNCS*, pages 104–117, Valparaíso, Chile, October 9–12, 2007. Springer, Heidelberg, Germany.

[32] Kenji Ohkuma, Hideo Shimizu, Fumihiko Sano, and Shinichi Kawamura. Security assessment of Hierocrypt and Rijndael against the differential and linear cryptanalysis (extended abstract). Cryptology ePrint Archive, Report 2001/070, 2001. `http://eprint.iacr.org/2001/070`.

[33] Jin Hyung Park and Dong Hoon Lee. FACE: Fast AES CTR mode encryption techniques based on the reuse of repetitive data. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(3):469–499, 2018. `https://tches.iacr.org/index.php/TCHES/article/view/7283`.

[34] Sangwoo Park, Soo Hak Sung, Seongtaek Chee, E-Joong Yoon, and Jongin Lim. On the security of Rijndael-like structures against differential and linear cryptanalysis. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 176–191, Queenstown, New Zealand, December 1–5, 2002. Springer, Heidelberg, Germany.

[35] Phillip Rogaway and Thomas Shrimpton. A Provable-Security Treatment of the Key-Wrap Problem. In *EUROCRYPT*, pages 373–390, 2006.

[36] Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 373–390, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany.

[37] Atri Rudra, Pradeep K. Dubey, Charanjit S. Jutla, Vijay Kumar, Josyula R. Rao, and Pankaj Rohatgi. Efficient Rijndael encryption implementation with composite field arithmetic. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *LNCS*, pages 171–184, Paris, France, May 14–16, 2001. Springer, Heidelberg, Germany.

[38] Mahdi Sajadieh, Arash Mirzaei, Hamid Mala, and Vincent Rijmen. A new counting method to bound the number of active s-boxes in rijndael and 3d. *Des. Codes Cryptography*, 83(2):327–343, 2017.

[39] Yu Sasaki. Known-key attacks on Rijndael with large blocks and strengthening ShiftRow parameter. In Isao Echizen, Noboru Kunihiro, and Ryôichi Sasaki, editors, *IWSEC 10: 5th International Workshop on Security, Advances in Information and Computer Security*, volume 6434 of *LNCS*, pages 301–315, Kobe, Japan, November 22–24, 2010. Springer, Heidelberg, Germany.

[40] Akashi Satoh, Sumio Morioka, Kohji Takano, and Seiji Munetoh. A compact Rijndael hardware architecture with S-box optimization. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 239–254, Gold Coast, Australia, December 9–13, 2001. Springer, Heidelberg, Germany.

[41] Beomsik Song and Jennifer Seberry. Consistent differential patterns of Rijndael. In Pil Joong Lee and Chae Hoon Lim, editors, *ICISC 02: 5th International Conference on Information Security and Cryptology*, volume 2587 of *LNCS*, pages 149–163, Seoul, Korea, November 28–29, 2003. Springer, Heidelberg, Germany.

[42] Merrielle Spain and Mayank Varia. Diversity within the rijndael design principles for resistance to differential power analysis. In Sara Foresti and Giuseppe Persiano, editors, *CANS 16: 15th International Conference on Cryptology and Network Security*, volume 10052 of *LNCS*, pages 71–87, Milan, Italy, November 14–16, 2016. Springer, Heidelberg, Germany.

[43] François-Xavier Standaert, Siddika Berna Örs, and Bart Preneel. Power analysis of an FPGA: Implementation of Rijndael: Is pipelining a DPA countermeasure? In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *LNCS*, pages 30–44, Cambridge, Massachusetts, USA, August 11–13, 2004. Springer, Heidelberg, Germany.

[44] François-Xavier Standaert, Gaël Rouvroy, Jean-Jacques Quisquater, and Jean-Didier Legat. Efficient implementation of Rijndael encryption in reconfigurable hardware: Improvements and design tradeoffs. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2003*, volume 2779 of *LNCS*, pages 334–350, Cologne, Germany, September 8–10, 2003. Springer, Heidelberg, Germany.

[45] Qingju Wang, Dawu Gu, Vincent Rijmen, Ya Liu, Jiazhe Chen, and Andrey Bogdanov. Improved impossible differential attacks on large-block Rijndael. In Taekyoung Kwon, Mun-Kyu Lee, and Daesung Kwon, editors, *ICISC 12: 15th International Conference on Information Security and Cryptology*, volume 7839 of *LNCS*, pages 126–140, Seoul, Korea, November 28–30, 2013. Springer, Heidelberg, Germany.

[46] Yuechuan Wei, Bing Sun, and Chao Li. New integral distinguisher for Rijndael-256. Cryptology ePrint Archive, Report 2009/559, 2009. `http://eprint.iacr.org/2009/559`.

[47] Ralph Wernsdorf. The round functions of RIJNDAEL generate the alternating group. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encryption – FSE 2002*, volume 2365 of *LNCS*, pages 143–148, Leuven, Belgium, February 4–6, 2002. Springer, Heidelberg, Germany.

[48] HongJun Wu. Hongjun's optimized C-code for AES-128 and AES-256. eSTREAM Project, 2007.

[49] Sung-Ming Yen. Amplified differential power cryptanalysis on Rijndael implementations with exponentially fewer power traces. In Reihaneh Safavi-Naini and Jennifer Seberry, editors, *ACISP 03: 8th Australasian Conference on Information Security and Privacy*, volume 2727 of *LNCS*, pages 106–117, Wollongong, NSW, Australia, July 9–11, 2003. Springer, Heidelberg, Germany.

[50] Lei Zhang, Wenling Wu, Je Hong Park, Bon Wook Koo, and Yongjin Yeom. Improved impossible differential attacks on large-block Rijndael. In Tzong-Chen Wu, Chin-Laung Lei, Vincent Rijmen, and Der-Tsai Lee, editors, *ISC 2008: 11th International Conference on Information Security*, volume 5222 of *LNCS*, pages 298–315, Taipei, Taiwan, September 15–18, 2008. Springer, Heidelberg, Germany.

Appendix A

Appendix

A.1 Tabular Representation of Some Mappings Used in Rijndael256

A.1.1 The S-box S_{RD} of Rijndael256

The affine (over GF(2)) transformation f composing the S-box S_{RD} of Rijndael256 is defined by [7]:

$$
\begin{array}{rcl}
b & = & f(a) \\
 & \Updownarrow & \\
\begin{bmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{bmatrix} & = & \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} a_7 \\ a_6 \\ a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}
\end{array}
$$

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Table A.1: The tabular representation of $S_{RD}(xy)$ [7]

A.1.2 The Round Constants Used in **Rijndael256**

i	0	1	2	3	4	5	6	7
RC[i]	00	01	02	04	08	10	20	40
i	8	9	10	11	12	13	14	15
RC[i]	80	1B	36	6C	D8	AB	4D	9A
i	16	17	18	19	20	21	22	23
RC[i]	2F	5E	BC	63	C6	97	35	6A
i	24	25	26	27	28	29	30	31
RC[i]	D4	B3	7D	FA	EF	C5	91	39

Table A.2: The round constants RC[·] [7]