Generalised Rijndael

Sergi Blanch-Torné[1], Ramiro Moreno Chiral[2], Francesc Sebé Feixa[2]

[1] Escola Politècnica Superior, Universitat de Lleida. Spain.
sblanch@alumnes.udl.es
[2] Departament de Matemàtica. Universitat de Lleida. Spain.
{ramiro,fsebe}@matematica.udl.es

September 13, 2012
Versión 0.0.2

Abstract. [3] This is the abstract

Keywords: Cryptography, Symmetric, Rijndael

1 Introduction

[1] [2] [3] [4]

2 Approach to the Rijndael Schema

Definition 1. *A Pseudo-Random Permutation (PRP) is defined as a application from the message space $\mathcal{M}$ and the key space $\mathcal{K}$ to the cipher space $\mathcal{C}$:*

$$\text{PRP: } \mathcal{M} \times \mathcal{K} \to \mathcal{C}$$

such that:

1. $\exists$ *"efficient"* deterministic *algorithm* $c = E(k, m)$
2. *The functions E is bijective*
3. $\exists$ *"efficient" inversion algorithm such that* $m = D(k, c)$

A pseudo-random permutation is used as a symmetric cryptosystem like Shannon have defined in [5]. Also Shannon have defined the concept of the *perfect secrecy*

Definition 2. *A* cipher *has* perfect secrecy *if* $\forall m_1, m_2 \in \mathcal{M}$ *s.t.* $|m_1| = |m_2| \wedge \forall c \in \mathcal{C}$ *and* $k \in_R \mathcal{K}$ *(random and uniform distributed), the probability to that c comes from m_1 or m_2 are the same*

$$Pr[E(k, m_1) = c] = Pr[E(k, m_2) = c]$$

[3] Partially founded by the Spanish project MTM20__-_____-___-__

This means that c does not reveal *any* information about the original m. This can also by says like: The distribution of the cipher of a message is the same than the distribution from another message, or formally:

Definition 3. *For a perfect secrecy system, the distributions of the ciphers between messages in the cipher space is computationally indistinguishable:*

$$\{E(k, m_1)\} \approx_p \{E(k, m_2)\}$$

Consider an scenario where an adversary has access to a random oracle where the output of this oracle can be or the output of the PRP or a truly random output, the advantage of the adversary to distinguish between if the output is get from one or the other can be described as:

$$Adv_F^{prp}(A) = Pr[Exp_F^{prp-1}(A) = 1] - Pr[Exp_F^{prp-0}(A) = 1] \tag{1}$$

where Exp_F^{PRP-1} is the probability to the adversary to win the bet that the output comes from a the PRP and Exp_F^{PRP-0} when the output comes from a truly random.

Definition 4. *A PRP is secure if for all "efficient" adversary, the advantage to distinguish if the output is from the PRP or the truly random is "negligible"*

The most efficient attacks on Rijndael that means this algorithm is still secure.

2.1 Design

3 Generalising the schema

3.1 key expansion

3.2 Rounds

3.3 subBytes

This transformation is a non-linear substitution of each word in the *state* matrix. In the original Rijndael it is used a substitution table called *S-Box*. This S-Box is represented in the figure 3 and there is also an inverse of it.

From the programmatic point of view the use of those boxes is so simple. Because the wordsize is 8 bits, by splitting the data to transform in two parts of 4 bits you can get the row and the column, taking the value in the cell as the value of the substitution. In the decipher operation, is used the inverse of the box, and with the same procedure of split the word and find the coordinates, but now with the inverse S-Box, the value you get back is the original data.

As an example, to transform the data `0x39` localise the cell in row `0x3` column `0x9`, and change the state matrix value with `0x12`. In the decipher procedure the transformation will be from the value `0x12`, reading the row `0x1` column `0x2` the cell have the value `0x39`, the original of this example.

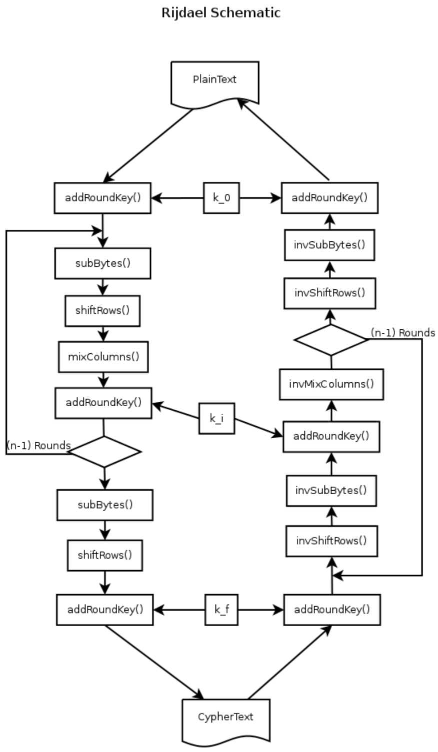


Fig. 1. rijndael diagram

But this tool of the *S-Box* is a faster way to compose two transformations in one and with not much computation.

The first transformation is to compute the multiplicative inverse in the field $\mathbb{F}_{2^w}$, where w is the wordsize ($w = 8$ in the original Rijndael). The second transformation is an affine transformation over the field $\mathbb{F}_{2^w}$. In the original Rijndael is:

$$b'_i = b_i \oplus b_{(i+4)mod8} \oplus b_{(i+5)mod8} \oplus b_{(i+6)mod8} \oplus b_{(i+7)mod8} \oplus c_i \tag{2}$$

Where b is the byte to be transformed and c is a fix value `0x63=0b01100011`. This transformation can be expressed as a matrix operation:

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1\,0\,0\,0\,1\,1\,1\,1 \\ 1\,1\,0\,0\,0\,1\,1\,1 \\ 1\,1\,1\,0\,0\,0\,1\,1 \\ 1\,1\,1\,1\,0\,0\,0\,1 \\ 1\,1\,1\,1\,1\,0\,0\,0 \\ 0\,1\,1\,1\,1\,1\,0\,0 \\ 0\,0\,1\,1\,1\,1\,1\,0 \\ 0\,0\,0\,1\,1\,1\,1\,1 \end{bmatrix} \cdot \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \tag{3}$$

Algorithm 1 KeyExpansion

INPUT: byte k[nRows*nColumns], nRounds, nRowns, nColumns, wSize
OUTPUT: word w[nRouns*(nRows+1)]

```
i := 0
while i¡nColumns do
    w[i] := word(k[nRows*(i+c) for c in range(nColumns)])
end while
i := nColumns
while i¡nRouns*(nRows+1) do
    temp := w[i-1]
    if i mod nColumns == 0 then
        temp := SubWord(RotWord(temp)) ⊕ Rcon[i/nColumns]
    else
        temp := SubWord(temp)
    end if
    w[i] := w[i-nColumns] ⊕ temp
    i++
end while
```

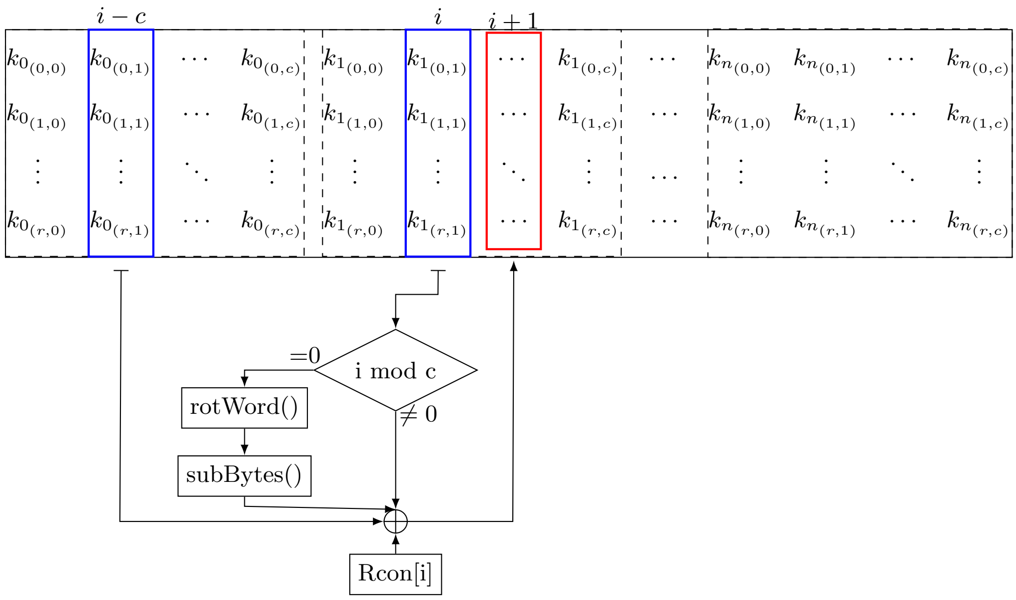


Fig. 2. Block diagram of the iterative construction of the *Rijndael Key Expansion* as a *PseudoRandomGenerator*, PRG

How to build different SBoxes Using the same *wordsize* there are two different things that can be changed: the `0x63` and the product over the field of equation 2. If the option is to use another wordsize this is the unique main parameter of the original Rijndael to set a different. With a wordsize of 4 the operations will be defined over $\mathbb{F}_{2^4}$, over 16 the field will be $\mathbb{F}_{2^{16}}$, and the subparameters of the affine transformation must also be set up.

3.4 shiftColumns

3.5 mixColumns

3.6 Operate in a polynomial ring, with coeficients in a polynomial field

$$\frac{\mathbb{F}_{2^n}[y]}{m(y)}$$

where $m(y)$ is a composed polynomial of degree r columns. This gives a polynomial ring. The coeficients of this polynomial ring are elements of a polynomial field

$$\mathbb{F}_{2^n} = \frac{\mathbb{F}_{2^1}[x]}{m(x)}$$

where $m(x)$ is irreductible and gives a polynomial field. Standard rijndael (AES) uses a circulan invertible matrix for this to simplify and speed up the operations in the ring.

3.7 addRoundKey

4 Parameter combinations

5 New useful sizes for Rijndael

[6]

	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF
0x0	0x63	0x7C	0x77	0x7B	0xF2	0x6B	0x6F	0xC5	0x30	0x01	0x67	0x2B	0xFE	0xD7	0xAB	0x76
0x1	0xCA	0x82	0xC9	0x7D	0xFA	0x59	0x47	0xF0	0xAD	0xD4	0xA2	0xAF	0x9C	0xA4	0x72	0xC0
0x2	0xB7	0xFD	0x93	0x26	0x36	0x3F	0xF7	0xCC	0x34	0xA5	0xE5	0xF1	0x71	0xD8	0x31	0x15
0x3	0x04	0xC7	0x23	0xC3	0x18	0x96	0x05	0x9A	0x07	0x12	0x80	0xE2	0xEB	0x27	0xB2	0x75
0x4	0x09	0x83	0x2C	0x1A	0x1B	0x6E	0x5A	0xA0	0x52	0x3B	0xD6	0xB3	0x29	0xE3	0x2F	0x84
0x5	0x53	0xD1	0x00	0xED	0x20	0xFC	0xB1	0x5B	0x6A	0xCB	0xBE	0x39	0x4A	0x4C	0x58	0xCF
0x6	0xD0	0xEF	0xAA	0xFB	0x43	0x4D	0x33	0x85	0x45	0xF9	0x02	0x7F	0x50	0x3C	0x9F	0xA8
0x7	0x51	0xA3	0x40	0x8F	0x92	0x9D	0x38	0xF5	0xBC	0xB6	0xDA	0x21	0x10	0xFF	0xF3	0xD2
0x8	0xCD	0x0C	0x13	0xEC	0x5F	0x97	0x44	0x17	0xC4	0xA7	0x7E	0x3D	0x64	0x5D	0x19	0x73
0x9	0x60	0x81	0x4F	0xDC	0x22	0x2A	0x90	0x88	0x46	0xEE	0xB8	0x14	0xDE	0x5E	0x0B	0xDB
0xA	0xE0	0x32	0x3A	0x0A	0x49	0x06	0x24	0x5C	0xC2	0xD3	0xAC	0x62	0x91	0x95	0xE4	0x79
0xB	0xE7	0xC8	0x37	0x6D	0x8D	0xD5	0x4E	0xA9	0x6C	0x56	0xF4	0xEA	0x65	0x7A	0xAE	0x08
0xC	0xBA	0x78	0x25	0x2E	0x1C	0xA6	0xB4	0xC6	0xE8	0xDD	0x74	0x1F	0x4B	0xBD	0x8B	0x8A
0xD	0x70	0x3E	0xB5	0x66	0x48	0x03	0xF6	0x0E	0x61	0x35	0x57	0xB9	0x86	0xC1	0x1D	0x9E
0xE	0xE1	0xF8	0x98	0x11	0x69	0xD9	0x8E	0x94	0x9B	0x1E	0x87	0xE9	0xCE	0x55	0x28	0xDF
0xF	0x8C	0xA1	0x89	0x0D	0xBF	0xE6	0x42	0x68	0x41	0x99	0x2D	0x0F	0xB0	0x54	0xBB	0x16

Fig. 3. Sbox for 8 bits word size

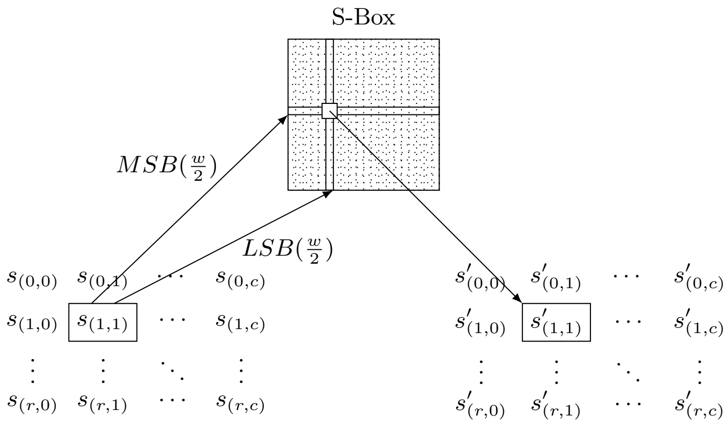


Fig. 4. Schematic diagram of the subBytes() transformation

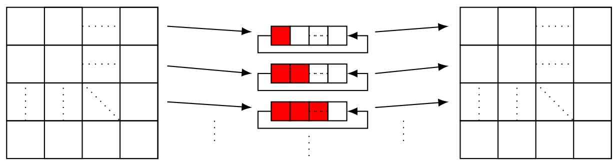
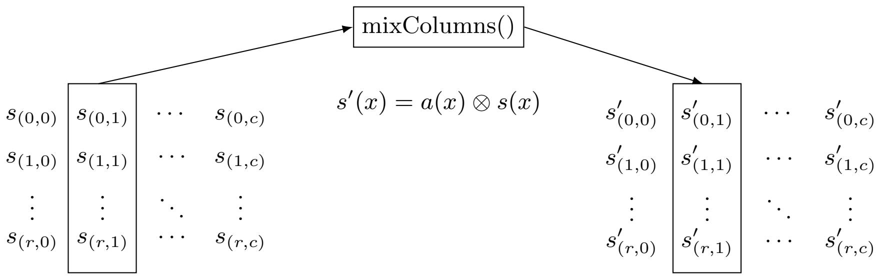


Fig. 5. Schematic diagram of the shiftColumns() transformation



$$s(x) = s_{(0,1)}x^{c-1} + s_{(1,1)}x^{c-2} + \cdots + s_{(r,1)}x^{c-c+1}$$
$$s(x), a(x), s'(x) \in \frac{\mathbb{F}_{2^w}[x]}{m(x)} \text{ with } m(x) \text{ reductible and order } c$$
$$s_{(i,j)} \in \frac{\mathbb{F}_{2^1}[z]}{m(z)} \text{ with } m(z) \text{ irreductible and order } w$$

Fig. 6. Diagram of the mixColumns() operation over the polynomial ring with coeficients in a polynomial field

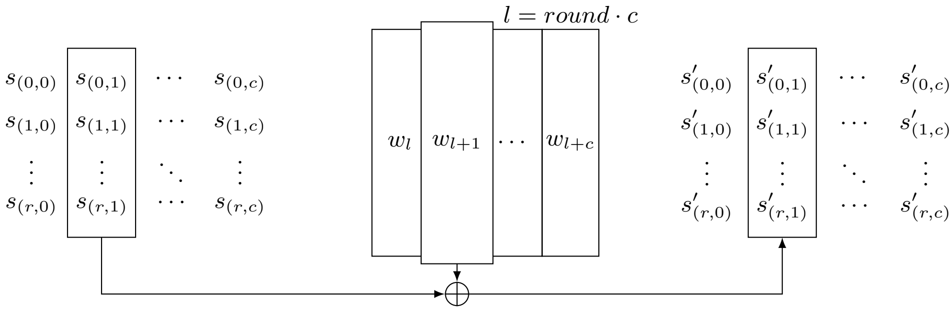


Fig. 7. Diagram of the addRoundKey()

References

1. J. Daemen and V. Rijmen, "The block cipher rijndael," in *Proceedings of the The International Conference on Smart Card Research and Applications*, CARDIS '98, (London, UK, UK), pp. 277–284, Springer-Verlag, 2000.
2. J. Daemen, J. Daemen, J. Daemen, V. Rijmen, and V. Rijmen, "Aes proposal: Rijndael," 1998.
3. J. Schaad and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm." RFC 3394 (Informational), Sept. 2002.
4. "Specification for the advanced encryption standard (aes)." Federal Information Processin Standards Publication 197, 2001.
5. C. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal, Vol 28, pp. 656–715*, 1949.
6. J. Daemen and V. Rijmen, "Efficient block ciphers for smartcards," in *Proceedings of the USENIX Workshop on Smartcard Technology on USENIX Workshop on Smartcard Technology*, WOST'99, (Berkeley, CA, USA), pp. 4–4, USENIX Association, 1999.