Table of Contents

Introduction

The Advanced Encryption Standard will be the new standard cryptographic algorithm for use by U.S.Goverment organizations to protect sensitive information. The algorithm Rijndael was chosen out of a group of contending algorithms. It is untended as the successor for DES (Data Encryption Standard).

Advanced Encryption Standard, a symmetric 128-bit block data encryption technique developed by Belgian cryptographers Joan Daemen and Vincent Rijmen. The U.S. Government adopted the algorithm as its encryption technique in October 2000, replacing the DES encryption it used. AES works at multiple network layers simultaneously. The National Institute of Standards and Technology of the U.S.Department of Commerce selected the algorithm, called Rijndael, Out of a group of five algorithms under consideration.

While the terms AES and Rijndael are used interchangeably, there are some differences between two. AES has fixed block size of 128-bit and a key size of 128, 192 or 256, whereas Rinjdael can be specified with any key and block size in multiple of 32-bits, with a minimum of 128-bits and a maximum of 256-bits.

Rijndael (pronounced rain-dahl) is the algorithm that has been selected by the U.S. National Institute of Standards and Technology (NIST) as the candidate for the Advance Encryption Standard. It was selected from a list of five finalists that were themselves selected from an original list of more than 15 submissions. Rijndael will begin to supplant the Data Encryption Standard (DES) - and later Triple DES - over the next few years in many cryptographic applications. The algorithm was designed by two Belgian cryptologists, Vincent Rijmen and Joan Daemen, whose surnames are reflected in the cipher's name. Rijndael has its origins in Square, an earlier collaboration between the two cryptologists.

Data Encryption Standard (DES)

The data encryption standard developed by IBM under the auspices of the United States Government. It was criticized because the research that went into the development of the standard remained classified. Concerns were raised that there might be hidden trap doors in the logic that would allow the government to break anyone's code if they wanted to listen in. DES uses a 56 bit key to perform a series of nonlinear transformation on a 64-bit data block. Even when it was first introduced a number of years ago, it was criticized for not having a long enough key. 56 bits just didn't put it far enough out of reach of a brute force attack. Today, with the increasing speed of hardware and its falling cost, it is feasible to build a machine that could crack a 56 bit key in under a day's time as demonstrated by EEF's DES Cracker Project (Electronic Frontier Foundation).

The Data Encryption Standard (DES) is a block cipher was selected by the National Bureau of Standards as an official Federal Information Processing Standard (FIPS) for the United States in 1976 and which has subsequently enjoyed widespread use internationally. It is based on a symmetric-key algorithm that uses a 56-bit key. The algorithm was initially controversial with classified design elements, a relatively short key length, and suspicions about a National Security Agency (NSA) backdoor. DES consequently came under intense academic scrutiny which motivated the modern understanding of block ciphers and their cryptanalysis.

DES is now considered to be insecure for many applications. This is chiefly due to the 56-bit key size being too small; in January, 1999,distributed.net and the Electronic Frontier Foundation collaborated to publicly break a DES key in 22 hours and 15 minutes . There are also some analytical results which demonstrate theoretical weaknesses in the cipher, although they are infeasible to mount in practice. The algorithm is believed to be practically secure in the form of Triple DES, although there are theoretical attacks. In recent years, the cipher has been superseded by the Advanced Encryption Standard (AES). Furthermore, DES has been withdrawn as a standard by the National Institute of Standards and Technology (formerly the National Bureau of Standards).

Advanced Encryption Standard (AES)

AES is short for Advanced Encryption Standard and is a United States encryption standard defined in Federal Information Processing Standard. AES is a symmetric encryption algorithm processing data in block of 128 bits. A bit can take the values zero and one, in effect a binary digit with two possible values as opposed to decimal digits, which can take one of 10 values. Under the influence of a key, a 128-bit block is encrypted by transforming it in a unique way into a new block of the same size. AES is symmetric since the same key is used for encryption and the reverse transformation, decryption. The only secret necessary to keep for security is the key. AES may configured to use different key-lengths, the standard defines 3 lengths and the resulting algorithms are named *AES-128*, *AES-192* and *AES-256* respectively to indicate the length in bits of the key. Each additional bit in the key effectively doubles the strength of the algorithm, when defined as the time necessary for an attacker to stage a brute force attack, i.e. an exhaustive search of all possible key combinations in order to find the right one.

BACKGROUND OF AES

In 1997 the US National Institute of Standards and Technology put out a call for candidates for a replacement for the ageing Data Encryption Standard, DES. 15 candidates were accepted for further consideration, and after a fully public process and three open international conferences, the number of candidates was reduced to five. In February 2001, the final candidate was announced and comments were solicited. 21 organizations and individuals submitted comments. None had any reservations about the suggested algorithm. AES is founded on solid and well-published mathematical ground, and appears to resist all known attacks well. There's a strong indication that in fact no back-door or known weakness exists since it has been published for a long time, has been the subject of intense scrutiny by researchers all over the world, and such enormous amounts of economic value and information is already successfully protected by AES. There are no unknown factors in its design, and it was developed by Belgian researchers in Belgium therefore voiding the conspiracy theories sometimes voiced concerning an encryption standard developed by a United States government agency. A strong encryption algorithm need only meet only single main criteria:

There must be no way to find the unencrypted clear text if the key is unknown, except brute force, i.e. to try all possible keys until the right one is found.

A secondary criterion must also be met:

The number of possible keys must be so large that it is computationally infeasible to actually stage a successful brute force attack in short enough a time.

The older standard, DES or Data Encryption Standard, meets the first criterion, but no longer the secondary one – computer speeds have caught up with it, or soon will. AES meets both criteria in all of its variants: AES-128, AES-192 and AES-256.

Encryption must be done properly

AES may, as all algorithms, be used in different ways to perform encryption. Different methods are suitable for different situations. It is vital that the correct method is applied in the correct manner for each and every situation, or the result may well be insecure even if AES as such is secure. It is very easy to implement a system using AES as its encryption algorithm, but much more skill and experience is required to do it in the right way for a given situation. No more than a hammer and a saw will make anyone a good carpenter, will AES make a system secure by itself. To describe exactly how to apply AES for varying purposes is very much out of scope for this short introduction.

Strong keys

Encryption with AES is based on a secret key with 128, 192 or 256 bits. But if the key is easy to guess it doesn't matter if AES is secure, so it is as critically vital to use good and strong keys as it is to apply AES properly. Creating good and strong keys is a surprisingly difficult problem and requires careful design when done with a computer. The challenge is that computers are notoriously deterministic, but what is required of a good and strong key is the opposite – unpredictability and randomness. Keys derived into a fixed length suitable for the encryption algorithm from passwords or pass phrases typed by a human will seldom correspond to 128 bits much less 256. To even approach 128--bit equivalence in a pass phrase, at least 10 typical passwords of the kind frequently used in day-to-day work are needed. Weak keys can be somewhat strengthened by special techniques by adding computationally intensive steps which increase the amount of computation necessary to break it. The risks of incorrect usage, implementation and weak keys are in no way unique for AES; these are shared by all encryption algorithms. Provided that the implementation is correct, the

security provided reduces to a relatively simple question about how many bits the chosen key, password or pass phrase really corresponds to. Unfortunately this estimate is somewhat difficult to calculate, when the key is not generated by a true random generator.

Security is relative

Security is not an absolute; it's a relation between time and cost. Any question about the security of encryption should be posed in terms of how long time, and how high cost will it take an attacker to find a key? Currently, there are speculations that military intelligence services possibly have the technical and economic means to attack keys equivalent to about 90 bits, although no civilian researcher has actually seen or reported of such a capability. Actual and demonstrated systems today, within the bounds of a commercial budget of about 1 million dollars can handle key lengths of about 70 bits. An aggressive estimate on the rate of technological progress is to assume that technology will double the speed of computing devices every year at an unchanged cost. If correct, 128-bit keys would be in theory be in range of a military budget within 30-40 years. An illustration of the current status for AES is given by the following example, where we assume an attacker with the capability to build or purchase a system that tries keys at the rate of one billion keys per second. This is at least 1 000 times faster than the fasted personal computer in 2004. Under this assumption, the attacker will need about 10 000 000 000 000 000 000 000 years to try all possible keys for the weakest version, AES-128. The key length should thus be chosen after deciding for how long security is required, and what the cost must be to brute force a secret key. In some military circumstances a few hours or days security is sufficient – after that the war or the mission is completed and the information uninteresting and without value. In other cases a lifetime may not be long enough.

Rinjdeal

Rijndael is the block cipher algorithm recently chosen by the National Institute of Science and Technology (NIST) as the Advanced Encryption Standard (AES). It supersedes the Data Encryption Standard (DES). NIST selected Rijndael as the standard symmetric key encryption algorithm to be used to encrypt sensitive (unclassified) American federal information. The choice was based on a careful and comprehensive analysis of the security and efficiency characteristics of Rijndael's algorithm.

Rijndael is an iterated block cipher. Therefore, the encryption or decryption of a block of data is accomplished by the iteration (a round) of a specific transformation (a round function). Section 3 provides the details of the Rijndael round function. Rijndael also defines a method to generate a series of sub keys from the original key. The generated sub keys are used as input with the round function.

Rijndael was developed by Belgian cryptographers Joan Daemen of Proton World International and Vincent Rijmen of Kathlieke Universities Leuven . The algorithm that they developed was designed as an easily understandable mathematical structure that can be broken down into simple components. Daemen and Rijmen write in their proposal to AES that Rijndael was designed based on the following three criteria:

1. Resistance against all known attacks;
2. Speed and code compactness on a wide range of platforms;
3. Design simplicity

Rijndael was evaluated based on its security, its cost and its algorithm and implementation characteristics. The primary focus of the analysis was on the cipher's security, but the choice of Rijndael was based on its simple algorithm and implementation characteristics. There were several candidate algorithms but Rijndael was selected because based on the analyses, it had the best combination of security, performance, efficiency, ease of implementation and flexibility.

Encryption

Since Rijndael is an iterated block cipher, the encryption or decryption of a block of data is accomplished by the iteration (a round) of a specific transformation (a round function). As input, Rijndael accepts one-dimensional 8-bit byte arrays that create data blocks. The plaintext is input and then mapped onto state bytes. The cipher key is also a one-dimensional 8-bit byte array.

With an iterated block cipher, the different transformations operate in sequence on intermediate cipher results (states). The design of Rijndael is based on easily understandable mathematical concepts including finite field mathematics and linear algebra for matrix manipulation.

Key and Block Size

A prime feature of Rijndael is its ability to operate on varying sizes of keys and data blocks. It provides extra flexibility in that both the key size and the block size may be 128, 192, or 256 bits. Since Rijndael specifies three key sizes, this means that there are approximately 3.4 x 1038 possible 128-bit keys, 6.2 x 1057 possible 192-bit keys and 1.1 x 1077 possible 256-bit keys. To compare, DES keys are only 56 bits long, which means there are approximately 7.2 x 1016 possible DES keys. Therefore, there are on the order of 1021 times more AES 128-bit keys than DES 56-bit keys.

The Sub key and the Key Schedule The sub keys are derived from the cipher key using the Rijndael key schedule. The cipher key is expanded to create an expanded key and the sub key is created by deriving a 'round key' by round key. The required round key length is equal to the data block length multiplied by the number of rounds plus 1. Therefore, the round keys are taken from the expanded key.

To maintain a secure system, the expanded key is always derived from the cipher key. This method ensures that the expanded key is never directly specified, which would open Rijndael up to several cryptanalytic attacks against its key generation methods. Recall that the security of this system depends entirely on the secrecy of the key, as the design of the algorithm itself is public and contains no secrecy.

Whole Byte operations, There are several mathematical preliminaries that define the addition and multiplication operations within a finite field and with matrices. When performing finite mathematics, the bytes are treated as polynomials rather than numbers, which can allow different and occasionally allows for more simple implementations.

Enciphering with Rijndael, The Rijndael cipher is an iterative block cipher. It therefore consists of a sequence of transformations to encipher or decipher the data. Rijndael encryption and decryption begin and end with a step to mix sub keys with the data block. This extra step is done as a protection against cryptanalysis. To encipher a block of data in Rijndael, you must first perform an Add Round Key step (XORing a sub key with the block) by itself, then the regular transformation rounds, and then a final round with the Mix Column step omitted. The cipher itself is defined by the following steps:

An initial Round Key addition (AddRoundKey);

Nr-1 Rounds ;(where r is 10, 12 or 14)

A final round.

The first Nr-1 rounds are similar consisting of

ByteSub

ShiftRow

MixColumn

AddRoundKey

The Last Round only perform the Transformation

ByteSub

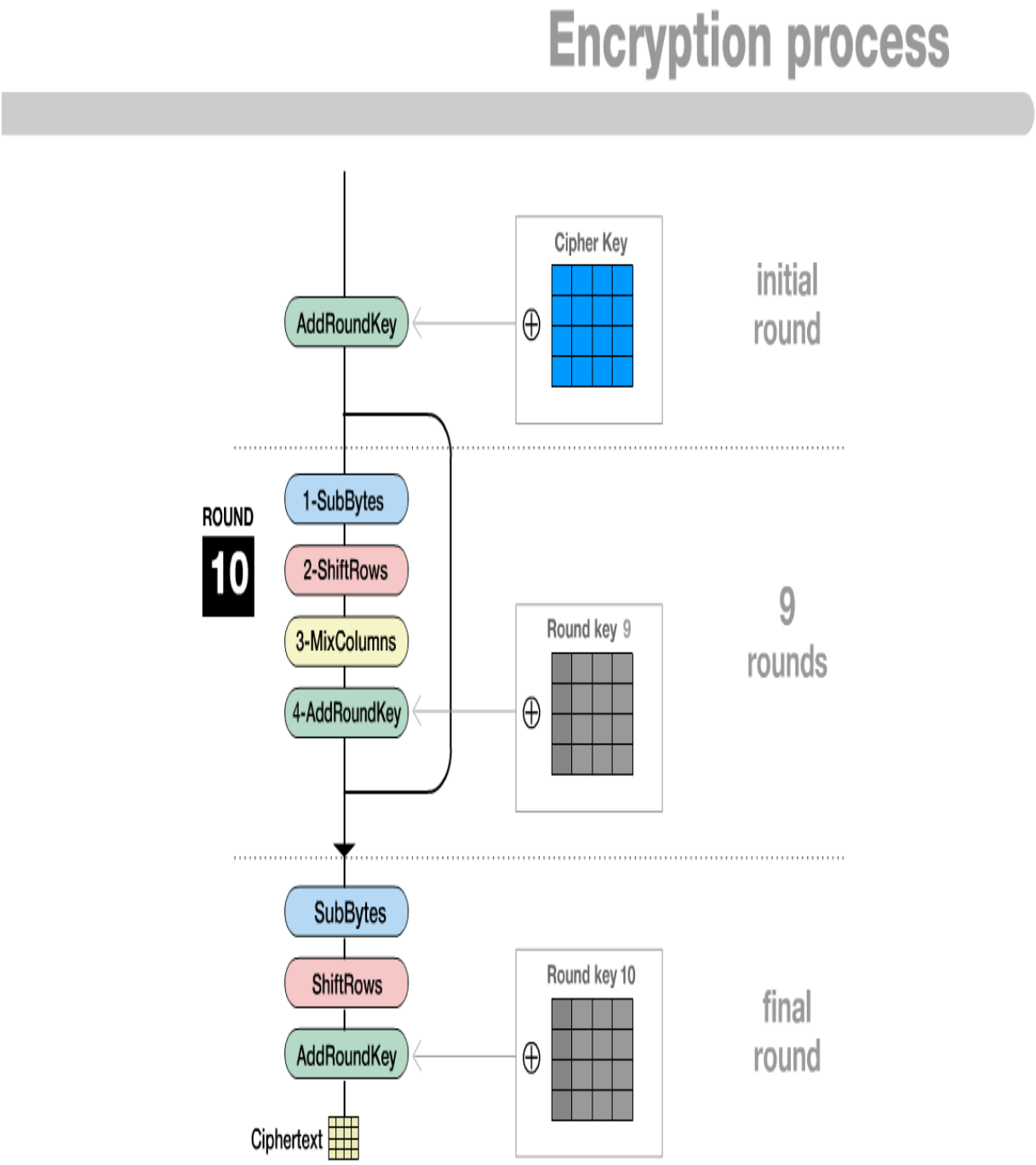
ShiftRow

AddRoundKey

Where Nr is the number of rounds that must be performed. Nr depends on the length of the data block (Nb) and the length of the key (Nk). Not counting an extra round performed at the end of decipherment, the number of rounds in Rijndael is: 9 if both the block and the key are 128 bits long, 11 if either the block or the key is 192 bits long, and neither of them is longer than that, and 13 if either the block or the key is 256 bits long.

AES/Rijndael Key-Block Round Combination

Nr	**Nb = 4**	**Nb = 6**	**Nb = 8**
Nk = 4	10	12	14
Nk = 6	12	12	14
Nk = 8	14	14	14



The Round Transformation

The round transformation is denoted Round, and is a sequence of four transformations,Called steps. This is shown in List. 3. 2. The final round of the cipheris slightly different. It is denoted Final Round and also shown in List. 3.2.

In the listings, the transformations (Round, SubBytes , ShiftRows , . . .) operation arrays to which pointers (State, ExpandedKeyr) are provided. It is easy to verify that the transformation Final Round is equal to the transformation Round, but with the Mix Columns step removed. The steps are specified in the following subsections, together with the design criteria we used for each step. Besides the step-specific criteria, we also applied the following two general design criteria:

1. Inevitability. The structure of the Rijndael round transformation requires that all steps be invertible.
2. Simplicity. As explained in Chap. 5, we prefer simple components over complex ones .

The pseudo C code is:

```
Rijndael(State,CipherKey)
{
        KeyExpansion(CipherKey,ExpandedKey);
        AddRoundKey(State,ExpandedKey);
        For( i=1 ; i<Nr-1 ; i++)
                Round(State, RoundKey);
         FinalRound(State,ExpandedKey + Nb*Nr);
}
```

And the round function is defined as:

```
Round(State,RoundKey)
 {
        ByteSub(State);
        ShiftRow(State);
        MixColumn(State);
        AddRoundKey(State,RoundKey);
}
```

The SubBytes Step

The SubBytes Step is the only non-liner transformation of the cipher. The SubBytes is a bricklayer permutation consisting of an S-Box applied to the. We denote the particular S-Box being used in Rijndael by S_{RD}. Figure illustrates the effect of the SubBytes steps on the state.

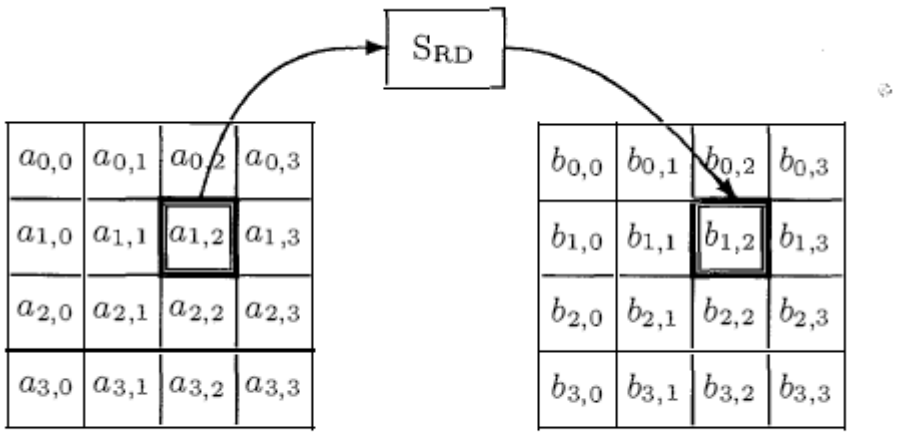


Fig. 3.2. SubBytes acts on the individual bytes of the state.

The ShiftRow Step

The ShiftRows step is a byte transposition that cyclically shifts the rows of the state over different offsets. Row 0 is shifted over C_0 rows of the state over different offsets. Row 0 is shifted over C_0 bytes, row 1 over C_1 bytes, row 2 over C_2 bytes and row 3 over C_3 bytes, so that the bytes at position j in row i moves to position $(j\text{-}C_{i)}$ mod N_b. The Shift offset $C_{0,}$ C_1, C_2 and C_3 depend on the value of N_b.

Table 3.1. `ShiftRows`: shift offsets for different block lengths.

N_b	C_0	C_1	C_2	C_3
4	0	1	2	3
5	0	1	2	3
6	0	1	2	3
7	0	1	2	4
8	0	1	3	4

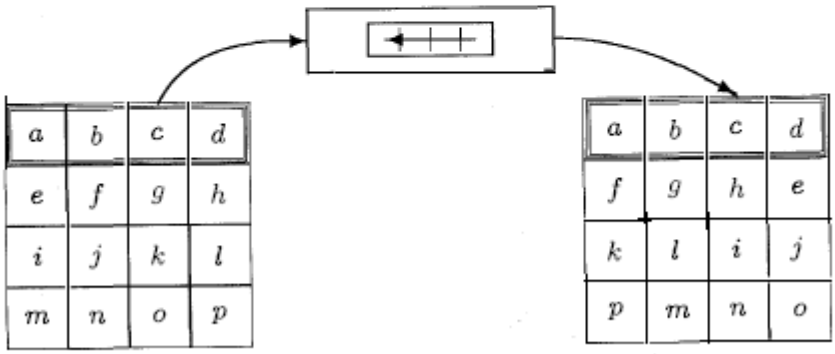


Fig. 3.4. `ShiftRows` operates on the rows of the state.

The MixColumn Step

The MixColumn step us bricklayer permutation operating on the state column by column.

The polynomial C(x) is given by

$C(x) = 03.x^3 + 01.x^2 + 01.x + 02$

This polynomial is comprised to x^4+1 and therefore invertible. As describe in following section.

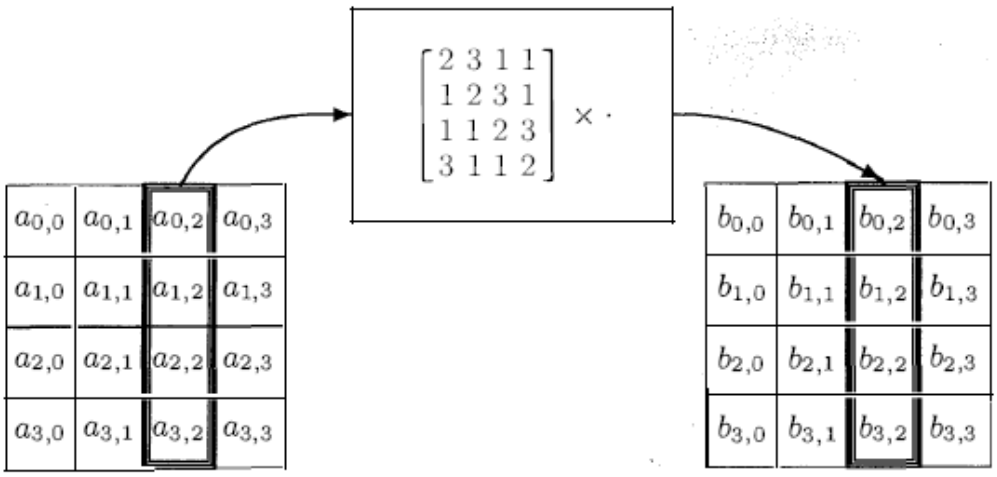


Fig. 3.6. MixColumns operates on the columns of the state.

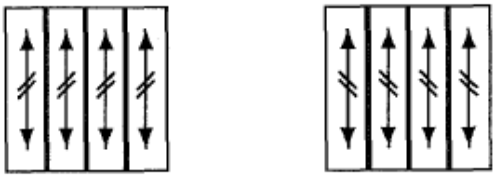


Fig. 3.7. Pictograms for MixColumns *(left)* and InvMixColumns *(right)*.

The AddRoundKey Step

The key addition is denoted AddRoundKey. In this transformation, the state is modified by combining it with a round key with the bitwise X-OR operation. A round key is denoted by ExpandedKey[i], $0 \leq i \leq N_r$. The array of round keys ExpandedKey is derived from the cipher key by means of the key schedule. The round key length is equal to the block legth.The AddRoundKey transformation is illustrated in fig. AddRoundKey is its own inverse.

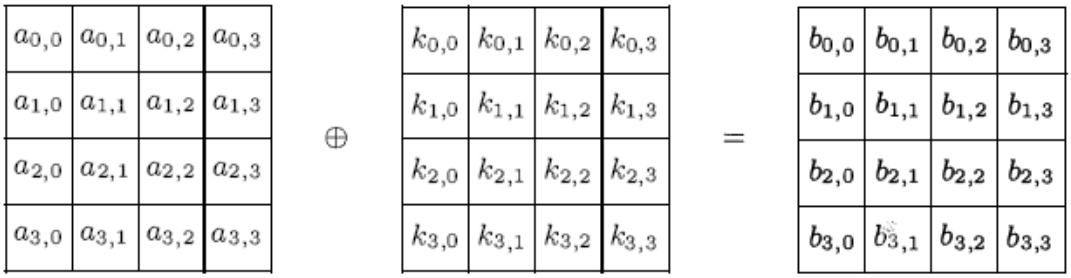


Fig. 3.8. In AddRoundKey, the round key is added to the state with a bitwise XOR.

Fig. 3.9. Pictogram for AddRoundKey.

The round transformation is broken into layers. These layers are the linear mixing layer, which provides high diffusion over multiple rounds. The non-linear layer which are basically applications of the Rijndael S-box. And the key addition layer which is simply an exclusive or of the round key and the intermediate state. Each layer is designed to have its own well-defined function which increases resistance to linear and differential cryptanalysis.

These layers are accomplished by four transformation steps. The Byte Sub step is a non-linear byte substitution. The Shift Row transformation is a cyclic shift. This is followed by the MixColumn step in which the columns of the State are considered as polynomials over a finite field and are modulo multiplied with a fixed polynomial. Finally, the Add Round Key step is performed.

The Byte Sub step is a non-linear byte substitution that operates on each of the 'state' bytes independently, where a state is an intermediate cipher result. The operation of this step is accomplished with a substitution table (S-box). ByteSub returns a word in which each byte from the incoming state are mapped by the S-box to corresponding bytes. The Rijndael S-box is simple, invertible and composed of two transformations. First, the multiplicative inverse of the data is taken in finite field space GF(28) and then an affine transformation is applied.

Diffusion is defined as the minimum number of active S-boxes in a linear or differential characteristic [10]. Diffusion is important because within block cipher cryptanalysis, almost all the attacks have a complexity that depends on the number of active S-boxes. The cryptanalytic complexity also is affected by the input/output correlation of the individual S-boxes.

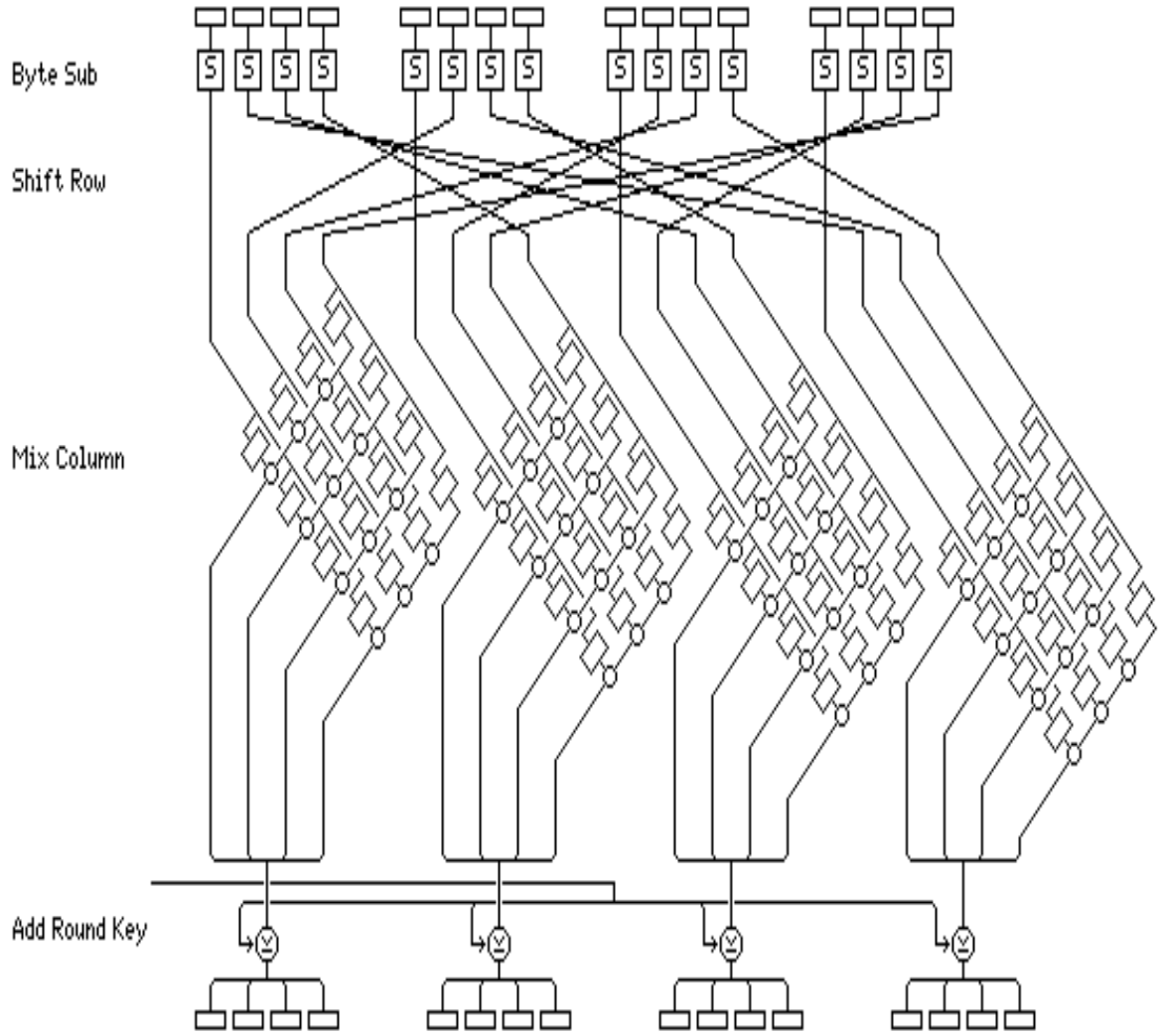
Linear diffusion layer results in provable lower bounds on the number of active S-boxes. They have provable lower bounds for the nonlinear order, the difference to linear functions and the resistance to linear and differential cryptanalysis.

The Shift Row step ensures that the different bytes of each row do not only interact with the corresponding byte in other rows. The rows of the State are cyclically shifted with different offsets. Row 0 is not shifted, Row 1 is shifted over C1 bytes, row 2 over C2 bytes and row 3 over C3 bytes. The shift offsets C1, C2 and C3 depend on the block length Nb .

The Mix Column step is where the different bytes interact with each other. It causes every byte in a column to affect every other byte. The state is viewed as polynomials and the transformation consists of matrix multiplication of the state with a multiplication polynomial over a finite field. The mix column transformation step is the only place in Rijndael's round transformation that the columns are mixed. This step works with the Shift Row step to ensure that all parts of the block impinge on each other.

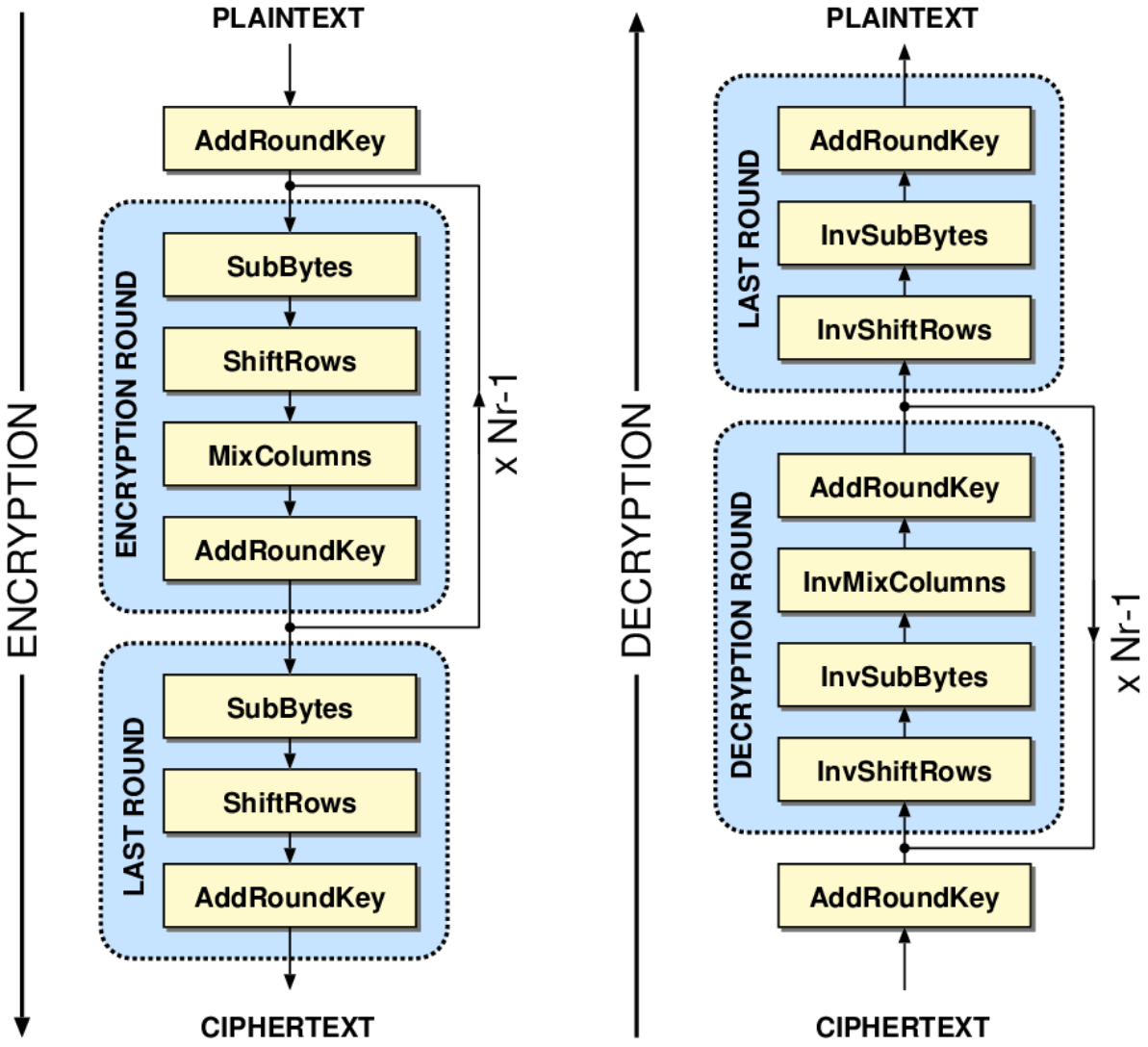
The Add Round Key step generates a new round key for the following round of transformations. The derivation of subkey material is defined in the section above entitled The Sub key and the Key Schedule.

The official Rijndael web site displays this image to promote understanding of the Rijndael round transformation.



Decryption

The algorithm for decryption can be found in a straightforward way be using the inverses of the steps InvSubBytes, InvShiftRows, InvMixColumns and AddRoundKey and reversing their order. In this algorithm, not only so the steps themselves differ from those used in encryption, but also the sequence in which the steps occur is different. The structure of Rijndael is such that it is possible to define an equivalent algorithm for decryption in which the sequence of steps is equal to that for encryption, with the steps replaced by their inverses and a change in the key schedule.



Algebraic Properties

In order to derive the equivalent decryption algorithm, we use two properties of the steps:

1. The order of InvShiftRows and InvSubBytes is indifferent
2. The order of the AddRoundKey and InvMixColumn can be iverted if the round key is adopted accordingly

The First property can be explained as follows. InvShiftRows simply transpose the bytes and has no effect on the bytes values. InvSubBytes operated on individual bytes, independent of their position. Therefore the two steps commute.

Round transformations of the straightforward decryption algorithm

InvRound (State , ExpandedKey[i])

{

AddRoundKey (State , ExpandedKey[i]) ;

InvMixColumns (Stat e) ;

InvShiftRows (State) ;

InvSubBytes (State) ;

}

InvFinalRound (State , ExpandedKey[Nr])

{

AddRoundKey(State , ExpandedKey[Nr]) ;

InvSh if tRows (State) ;

InvSubBytes (State) ;

}

Straightforward decryption algorithm for a two-round variant

AddRoundKey (State , ExpandedKey [2]) ;

InvShi ftRows (Stat e) ;

InvSubBytes(State) ;

AddRoundKey (State , ExpandedKey[l]) ;

InvMixColumns (Stat e) ;

InvShiftRows (State) ;

InvSubBytes (State) ;

AddRoundK ey (State , ExpandedKey[O]) ;

The explanation of the second property is somewhat more sophisticated.

For any linear transformation A : x --+ y = A (x), it holds by definition that

A(x E9 k) = A(x) E9 A(k) . (3.20)

Since AddRoundKey simply adds the const ant ExpandedKey[i] to its input,and InvMixColumns is a linear operation, the sequence of steps

AddRoundKey (State, ExpandedKey [i]) ;

InvMixColumns (State) ;

can be replaced by the following equivalent sequence of steps:

InvMixColumns (Stat e) ;

AddRoundKey (State, EqExpandedKey[i]) ;

where EqExpandedKey[i] is obtained by applying InvMixColumns to

ExpandedKey[i] .

Algorithm for the inverse key expansion for Nk > 6

```
InvKeyExpansion(byte K_i[4][N_k], byte W_i[4][N_b(N_r + 1)])
                                                  /* for N_k ≤ 6 */
{
for(j = 0; j < N_k; j++)
  for(i = 0; i < 4; i++) W_i[i][j] = K_i[i][j];
for(j = N_k; j < N_b(N_r + 1); j++)
  {
  if (j mod N_k == 0)
    {
    W_i[0][j] = W_i[0][j − N_k] ⊕ S[W_i[1][j − 1] ⊕ W_i[1][j − 2]] ⊕ RC[N_r + 1 − j/N_k];
    for(i = 1; i < 4; i++)
      W_i[i][j] = W_i[i][j − N_k] ⊕ S[W_i[i + 1 mod 4][j − 1] ⊕ W_i[i + 1 mod 4][j − 2]];
    }
  else
    {
    for(i = 0; i < 4; i++)
      W_i[i][j] = W_i[i][j − N_k] ⊕ W_i[i][j − N_k − 1];
    }
  }
  }
```

Algorithm for InvKeyExpansion

```
InvKeyExpansion(byte K_i[4][N_k], byte W_i[4][N_b(N_r + 1)])
                                                    /* for N_k > 6 */
{
for(j = 0; j < N_k; j++)
  for(i = 0; i < 4; i++) W_i[i][j] = K_i[i][j];
for(j = N_k; j < N_b(N_r + 1); j++)
  {
  if (j mod N_k == 0)
    {
    W_i[0][j] = W_i[0][j − N_k] ⊕ S[W_i[1][j − 1] ⊕ W_i[1][j − 2]] ⊕ RC[N_r + 1 − j/N_k];
    for(i = 1; i < 4; i++)
      W_i[i][j] = W_i[i][j − N_k] ⊕ S[W_i[i + 1 mod 4][j − 1] ⊕ W_i[i + 1 mod 4][j − 2]];
    }
  else if (j mod N_k == 4)
    {
    for(i = 0; i < 4; i++)
      W_i[i][j] = W_i[i][j − N_k] ⊕ S[W_i[i][j − N_k − 1]];
    }
  else
    {
    for(i = 0; i < 4; i++)
      W_i[i][j] = W_i[i][j − N_k] ⊕ W_i[i][j − N_k − 1];
    }
  }
  }
```

Strengths of RIJNDAEL

Daemen and Rijmen have specified Rijndael's advantages based on implementation aspects, simplicity of design, variable block length and extensions. Rijndael's implementation is very flexible since it can be used with varying key sizes and block sizes. It is also possible to change the sequence of some steps in Rijndael without affecting the cipher. The cipher is has a simple and elegant structure. It does not hide its structure by using complex components. Instead, it benefits from the advantages gained by the use of simple components in a well-defined structure. Rijndael's security is based on the interaction of the cipher's individual components.

Rijndael is described as having a 'rich algebraic structure' which allows the cipher's security to be easily assessed in a limited time frame. This is an advantage over more complex designs that require extensive thinking, searching and 'bit tracing'. Rijndael is consistently a very good performer in both hardware and software across a wide range of computing environments. Its key setup time is excellent, and its key agility is good. Rijndael's very low memory requirements make it very well suited for restricted-space environments. There is additional security in that Rijndael's operations are among the easiest to defend against power and timing attacks.

The following scenario has been described to convey Rijndael's resistance to brute force attacks, "Assuming that one could build a machine that could recover a DES key in a second (i.e., try 255 keys per second), then it would take that machine approximately 149 thousand-billion (149 trillion) years to crack a 128-bit AES key. To put that into perspective, the universe is believed to be less than 20 billion years old. [5]" It also provides good security against linear cryptanalysis, differential cryptanalysis and opportunistic attacks.

Weakness of Rijndael

Rijndael can be subject to standard techniques of differential and linear cryptanalysis. It is also weak to an attack called the "Square attack" that is based on the way matrix multiplication works, and because in GF(2^8), all the coefficients of the Mix Column matrix have reciprocals. However, in practical terms, this attack is not sufficient to compromise the security of the Rijndael algorithm.

Rijndael is also limited by its inverse cipher. Though the encryption is suited for many applications and is quite fast, the inverse cipher takes more code and cycles is not as well suited for different implementations, such as a smart card.

Comparison of RIJNDAEL to other AESs and DESs

Rijndael was chosen over four other candidate algorithms; MARS, RC6, Serpent and Twofish. Its performance was superior to all of the candidates for the large majority of performance analyses. Its encryption and decryption performance placed in the highest level for 64-bit (C and assembler), 8-bit (C and assmbler), 32-bit smartcard and for Digital Signal Processors. Its performance placed it in the second level when implemented with 32-bit C or Java. Its key scheduling performance placed in the first level for each platform. Therefore, AES concluded that Rijndael placed in the highest level for overall performance [10].

The Rijndael cipher algorithm was not chosen over the other AES candidates because it provided a higher level of security. NIST commented that each of the five finalists provided satisfactory and relatively equivalent security. So far, none of the ciphers have fallen to a cryptanalytic attack, though many have been attempted. Therefore, each cipher algorithm maintains a high level of security.

Rijndael was chosen because of the simplicity of its design. The design uses simple components with easily proven and verified properties. The simplicity of Rijndael's design and components provided an advantage over the other design candidates. Daemen and Rijmen write that "From the beginning, our design strategy was to use as simple as possible components, to define clear evaluation criteria, and to use simple components with easily provable properties where possible [9]". The other teams had some difficulty proving and analysing their ciphers. Daemen and Rijmen state that [11] "both the Serpent design team and the MARS design team did not produce S-boxes that are in accordance with their own design criteria... and the key dependent S-boxes of Twofish have made it apparently very difficult to determine the security of the cipher.

Conclusion

We Conclude that the in present the rijndael is the most secure and practically non-practically in crack able block cipher algorithm. There is no method yet invented to crack the rijndael.It have excellent resistance to known attacks. It is very efficient compare to other algorithm and also we can implement it in any plat-form.