Cryptography

AES Block Cipher

Prof. Dr. Heiko Knospe

TH Köln – University of Applied Sciences

May 10, 2020

Block Ciphers

A block cipher is a *keyed family of permutations* which is designed to behave like a pseudorandom permutation (*prp*). Block ciphers operate on binary strings of fixed length. However, in combination with an operation mode (for example CBC or CTR mode), they define variable-length encryption schemes.

$$E: \{0,1\}^n \times \{0,1\}^l \to \{0,1\}^l$$

Currently, a block length of $l = 128$ bits and key lengths between $n = 128$ and $n = 256$ bits are widely used.

Design of Block Ciphers

The encryption function

$$E_k : \{0,1\}^l \rightarrow \{0,1\}^l$$

should be indistinguishable from a true random function if k is secret.

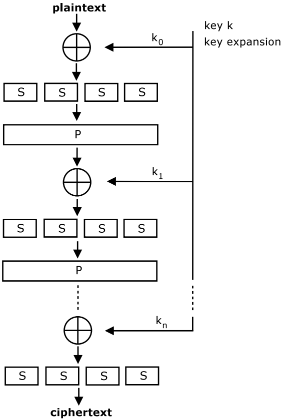
Diffusion and *confusion* are important design goals of block ciphers.

- Diffusion: Small input changes, e.g., only one bit, result in large output changes (avalanche effect).
- Confusion: Complex relationship between ciphertext and key, even if the plaintext is known.

Diffusion can be achieved by linear and affine maps, i.e., by XOR and matrix operations. Confusion requires *nonlinear* maps (S-Boxes). Since nonlinear maps are more difficult to describe, S-Boxes only operate on small segments of a block and are applied in parallel, e.g., to 8-bit segments of a 128-bit block.

Substitution-Permutation Networks

In practice, *substitution-permutation networks* (SPN) and *Feistel networks* are used to construct block ciphers.



SPN: Plaintext is transformed into ciphertext in several invertible rounds.

AES Encryption

The block cipher *Rijndael* has been adopted as *Advanced Encryption Standard (AES)* and the cipher is widely used today. The standardized AES cipher has a block length of 128 bits and a 128-, 192- or 256-bit key length. The Rijndael encryption function

$$E_k : \{0,1\}^{128} \to \{0,1\}^{128}$$

is given by a SPN. The 128-bit *state* is arranged in a 4×4 matrix over $GF(2^8)$ by writing the bytes $p_0, p_1, \ldots, p_{16}$ into the columns.

$$\begin{pmatrix} p_0 & p_4 & p_8 & p_{12} \\ p_1 & p_5 & p_9 & p_{13} \\ p_2 & p_6 & p_{10} & p_{14} \\ p_3 & p_7 & p_{11} & p_{15} \end{pmatrix}$$

Each byte is interpreted as an element of the field

$$GF(2^8) = GF(2)[x]/(x^8 + x^4 + x^3 + x + 1).$$

High-level Description of AES

```
Rijndael(State, CipherKey)
{
   KeyExpansion(CipherKey, ExpandedKey)
   AddRoundKey(State,ExpandedKey[0])
   for(i = 1; i < Nr ; i++)  {     // Nr is either 10, 12 or 14
      // Round i
      SubBytes(State)
      ShiftRows(State)
      MixColumns(State)
      AddRoundKey(State,ExpandedKey[i])
   }
   // Final Round
   SubBytes(State)
   ShiftRows(State)
   AddRoundKey(State,ExpandedKey[Nr])
}
```

AES S-Box

The AES S-Box `SubBytes` is the only non-affine component of AES.

The S-Box function $S_{RD} : GF(2^8) \to GF(2^8)$ is applied to each byte of the state individually.

$$\begin{pmatrix} p_0 & p_4 & p_8 & p_{12} \\ p_1 & p_5 & p_9 & p_{13} \\ p_2 & p_6 & p_{10} & p_{14} \\ p_3 & p_7 & p_{11} & p_{15} \end{pmatrix} \xrightarrow{\text{SubBytes}} \begin{pmatrix} S_{RD}(p_0) & S_{RD}(p_4) & S_{RD}(p_8) & S_{RD}(p_{12}) \\ S_{RD}(p_1) & S_{RD}(p_5) & S_{RD}(p_9) & S_{RD}(p_{13}) \\ S_{RD}(p_2) & S_{RD}(p_6) & S_{RD}(p_{10}) & S_{RD}(p_{14}) \\ S_{RD}(p_3) & S_{RD}(p_7) & S_{RD}(p_{11}) & S_{RD}(p_{15}) \end{pmatrix}$$

S-Box Function S_{RD}

The definition

$$GF(2^8) = GF(2)[x]/(x^8 + x^4 + x^3 + x + 1)$$

gives a $GF(2)$-linear isomorphism between $GF(2^8)$ and $GF(2)^8$. The invertible S-Box function is defined by multiplicative inversion in $GF(2^8)^*$ (which is highly nonlinear) followed by an affine map:

$$S_{RD} : GF(2^8) \to GF(2^8),\ S_{RD}(a) = \begin{cases} Aa^{-1} + b & \text{for } a \neq 0 \\ b & \text{for } a = 0 \end{cases}$$

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad b = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

ShiftRows

ShiftRows is a bit-permutation and rotates the bytes in the second, third and fourth row to the left. The first row is left unchanged, the bytes in second row are rotated by one position, bytes in the third row are rotated by two positions and bytes in the fourth row are rotated by three positions. Obviously, *ShiftRows* is invertible.

$$\begin{pmatrix} p_0 & p_4 & p_8 & p_{12} \\ p_1 & p_5 & p_9 & p_{13} \\ p_2 & p_6 & p_{10} & p_{14} \\ p_3 & p_7 & p_{11} & p_{15} \end{pmatrix} \xrightarrow{\text{ShiftRows}} \begin{pmatrix} p_0 & p_4 & p_8 & p_{12} \\ p_5 & p_9 & p_{13} & p_1 \\ p_{10} & p_{14} & p_2 & p_6 \\ p_{15} & p_3 & p_7 & p_{11} \end{pmatrix}$$

MixColumns

MixColumns transforms the columns of the state by a $GF(2^8)$-linear map. The product of a constant 4×4 matrix M over $GF(2^8)$ and the state matrix defines the new state, i.e., each column of the state is multiplied with M. The matrix M is invertible.

$$\begin{pmatrix} p_0 & p_4 & p_8 & p_{12} \\ p_1 & p_5 & p_9 & p_{13} \\ p_2 & p_6 & p_{10} & p_{14} \\ p_3 & p_7 & p_{11} & p_{15} \end{pmatrix} \xrightarrow{\text{MixColumns}} \underbrace{\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}}_{M} \cdot \begin{pmatrix} p_0 & p_4 & p_8 & p_{12} \\ p_1 & p_5 & p_9 & p_{13} \\ p_2 & p_6 & p_{10} & p_{14} \\ p_3 & p_7 & p_{11} & p_{15} \end{pmatrix}$$

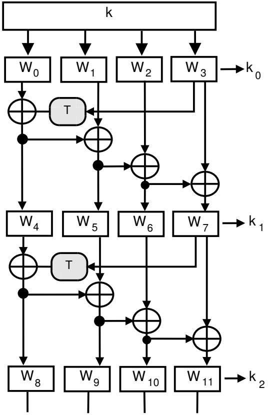
Key Scheduling

In the key expansion step, the 128-bit round keys $k_0, k_1, \ldots, k_r$ are derived from the AES key k. The key scheduling is nonlinear, which is intended protect the cipher against *related key attacks*.

Suppose k is a 128-bit AES key. Then AES has ten rounds and eleven 128-bit round keys $k_0, k_1, \ldots, k_{10}$ are required. During key expansion, 44 words $W_0, W_1, \ldots, W_{43} \in GF(2^8)^4$ of length 32 bits are computed. The round keys are given by

$$k_i = W_{4i} \| W_{4i+1} \| W_{4i+2} \| W_{4i+3} \text{ for } i = 0, 1, \ldots, 10.$$

Key Scheduling for 128-bit AES



The first two rounds of 128*-bit AES key scheduling (*$i = 1, 2$*). One defines* $T(W_{4i-1}) = \mathbf{S}(sh(W_{4i-1})) \oplus (RC_i, 0, 0, 0)$*, where sh is a left rotation,* $\mathbf{S} = S_{RD} \times S_{RD} \times S_{RD} \times S_{RD}$*,* RC_i *a round constant and* $\oplus$ *denotes XOR.*