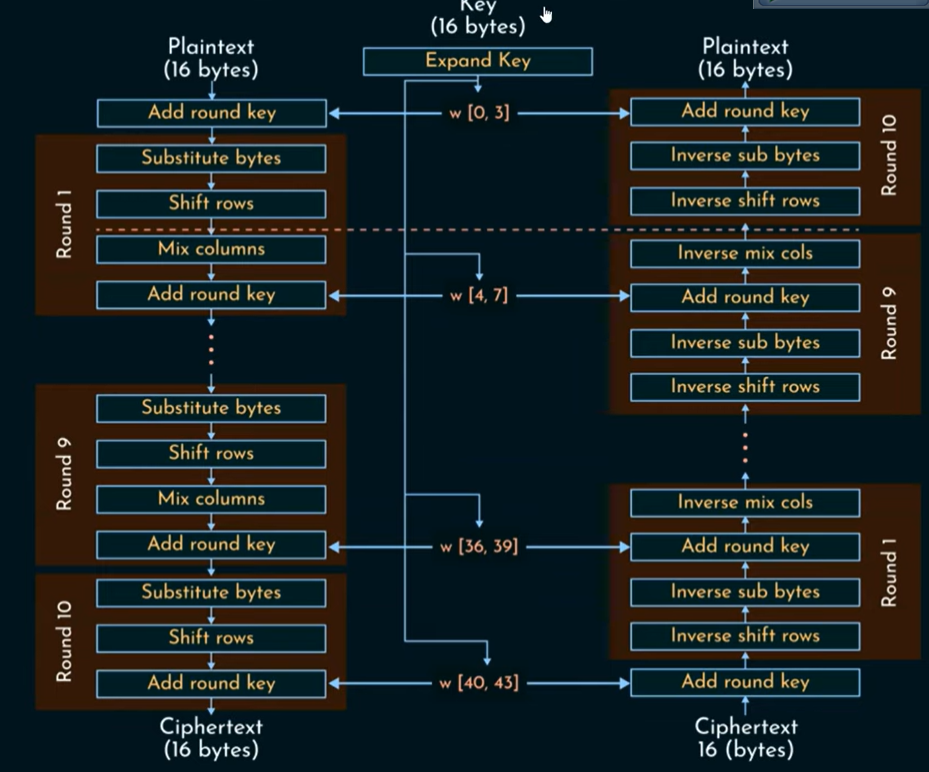


The 128 bits of the plaintext are arranged such that they form 4x4 matrix. The 4 transformations of the rounds are [substitute bytes – shift rows – mix columns – add round key]. Note that all rounds have same transformation steps except the initial transformation (a.k.a round 0) and the last round (Nth round). They number of rounds depends on the key size (illustrated in the following figure). However the **round keys** have constant sizes of 128 bits.

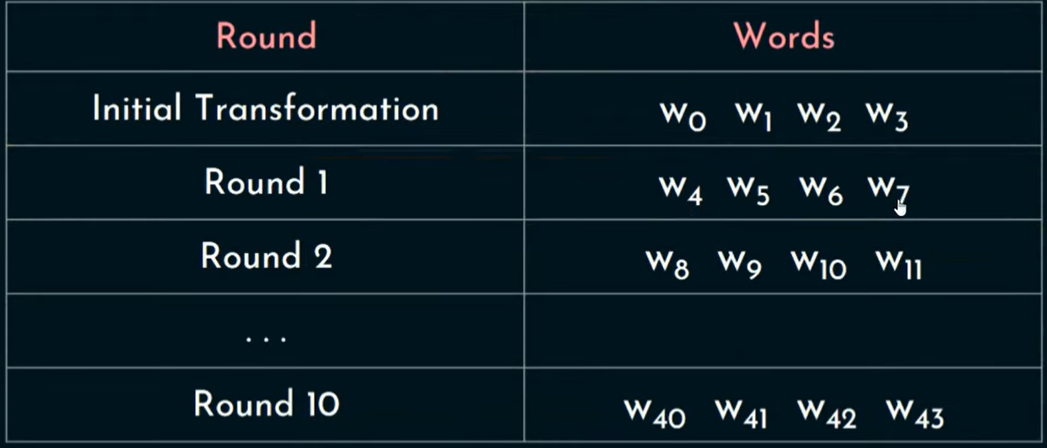
A black and white rectangular sign with numbers and hand pointing

Description automatically generated with medium confidence

The round keys are generated using the key schedule algorithm that is a function of the main key. The transformation of rounds is explained in the following figure. Note that the initial transformation **(having add round key only)** and the final transformation **(excluding the mix columns)** are unique to make the process invertible.



The word of the round key is 32 bits, which means that for 10 rounds, the key expansion unit exported 44 words with 4 words for each round. Now let’s talk about each transformation individually.



# Substitute Bytes

A screenshot of a computer

Description automatically generated

16x16 table is constructed where each element is represented by a byte that will be replaced by another value (ex. ) based on a look-up table shown in the following figure.



# Shift Rows

A screenshot of a computer

Description automatically generated

Based on the previous figure, the first row has 0 byte left circular shift, the second row has 1 byte left circular shift, the third row has 2 byte left circular shift, the fourth row has 3 byte left circular shift.

# Mix Columns

A computer screen shot of a computer

Description automatically generated

Matrix multiplication of the 4x4 data matrix with a pre-defined 4x4 matrix to have an output of 4x4 matrix.

# Add Round Key

A screenshot of a computer

Description automatically generated

As shown before, the round key has 4 words where each word is 32 bits. The word is stored in columns to add (simple XOR operation) it with each row corresponding to the 4x4 matrix to have new 4x4 matrix.

Each of the four previously mentioned operations is performed once in each round with the specified order. But the question now is **how to generate the round keys based on the main key**?

# AES Key Expansion

A screenshot of a computer

Description automatically generated

We need 11 round keys if the key size is 128 bits or 13 if key size is 192 bits or 15 if key size is 256 bits (due to the initial transformation in the beginning of the encryption). For the initial transformation, the first four words are generated directly from the summation (or concatenation, not sure yet) of the main key columns to get four words. However, to achieve the rest of the words, we need to perform the ‘G’ function for the last word in the round key (for the initial round it would be ) then add the output of the ‘G’ function to the first word to get the next first word of the round key.

## G Function

The word is divided into 4 bytes, where we perform 1 byte left circular shift at the beginning. Then we use the s-box (discussed earlier) to generate different bytes. Then we XOR the result with round constant (its generation is discussed briefly in the [**AES definition**](AES%20definition.docx)document) that have the following for 128-bit key size values.

A screenshot of a black and white screen

Description automatically generated

Final note, visit this [site](https://www.cryptool.org/en/cto/aes-step-by-step) to check the model correctness.