Crypto/Math Paper

SAND2020-7987R

AES Key Recovery from Round Keys

W.R. Cordwell

Sandia National Laboratories

SANDxxxxxx

03 November 2008

Summary

For AES-256, the entire key schedule, including the original secret key, can be recovered easily from a 32 consecutive byte portion of the key schedule.

Introduction

AES is a 128-bit block cipher that may use three possible lengths of the secret key: 128, 192, and 256 bits. For each size of key, there is a corresponding number of rounds, the 128-bit key requiring 10 rounds, the 192-bit key requiring 12 rounds, and the 256-bit key requiring 14 rounds. In each round, secret key material is mixed in, in 16-byte (128-bit) quantities. Additionally, there is an extra, initial XORing in of 16 bytes of key material. For 128-bit AES, this gives $11 \times 128 = 1408$ bits of key material, and for 256-bit AES, $15 \times 128 = 1920$ bits of key material. This "round key" material is generated by a *key expansion* of the original secret key.

128-bit and 256-bit AES are, by far, the most commonly used. Here, we shall focus on 256-bit AES. The results apply to the other cases, with some minor tweaks.

Description of Key Expansion

Using 32-bit words, 256-bit AES starts with eight words of secret key. The first round key is the first four words of the initial secret key; the second round key is the last four words of the initial secret key. Thereafter, each subsequent word, w_i, of the round keys is generated as follows:

Take the immediately previous word of the key expansion, w_{i-1}.

- If i is divisible by eight (numbering starts at zero, so this will happen for the first word after all of the initial secret key is used), cyclically shift the bytes of w_i, use the S-box lookup to substitute for all of the bytes, and XOR in a round constant. After this is done, XOR in the word of the key schedule that occurs eight words previously, viz., w_{i-8}.
- If i is not divisible by eight, but is divisible by four, take w_{i-1} and apply the S-box substitution for each byte, then XOR with w_{i-8}.
- If i is not divisible by four, just XOR w_{i-1} with w_{i-8}.

The important point is that, for each word of round key, $w_i = f(w_{i-1}) \oplus w_{i-8}$, where, most of the time, f does nothing, and the rest of the time it is easily computed.

Recovery of the Entire Key Expansion from Eight Consecutive Words

From knowing just eight consecutive words of the key expansion, one can recover the entire key expansion (including the original key). Recovering the later words of key is straightforward–just apply the key expansion algorithm and build the following words of the expansion. For the previous words of the key expansion, we simply peel back the process, illustrated by the following example.

Suppose that we know words 20 through 27 of the key expansion, $w_{20}, \ldots, w_{27}$. Word w_{27} was built from w_{26} and w_{19} by simply XORing, $w_{27} = w_{26} \oplus w_{19}$, so we recover $w_{19} = w_{27} \oplus w_{26}$. Similarly, $w_{18} = w_{26} \oplus w_{25}$, and we keep backing up. When we hit one of the words (with an index divisible by four) that required using the function f, we simply compute f of that word, and then back up, such as $w_{17} = w_{25} \oplus f(w_{24})$.

Comments

The key expansion process in AES was designed to help diffuse the bits of the secret key, and it uses a fairly simple process to do so. There was no attempt to use an irreversible process, such as a hash function, and the process is clearly reversible.

Note that, if one is performing decryption, which uses the key expansion in reverse, and one recovers the eight words of the key expansion used in the first part of the decryption, one can then recover the original key.

References

[1] Federal Information Processing Standard (FIPS) 197, Advanced Encryption Standard (AES), NIST, 2001