Serpent

Serpent is a modern block cipher published in 1998. It was one of the 5 finalists in the AES contest. It had the least number of negative votes and the second greatest positive vote count after Rijndael.

Construction

The cipher is a substitution-permutation network (SPN) consisting of 32 rounds (even though 16 rounds were deemed secure enough at the time of publication). Each round except the last consists of a *key mixing* operation, *substitution* (using 32 parallel 4-bit S-boxes) and a *linear transformation*. In the last round, the linear transformation is replaced by an additional key mixing step (this technique is known as *input and output whitening*).

There are 8 S-boxes used in Serpent; each one being used in 4 rounds (hence a total of 32 rounds). The S-boxes were generated deterministically from a *nothing-up-my-sleeve* seed and chosen based on their linear biases and differential characteristics, so that they would resist known cryptanalytical attacks.

Key expansion

As an AES candidate, Serpent offered 128, 192 and 256-bit keys. However, by design, it can accept any key between 0 and 256 bits inclusive. Any key that has less than the full 256 bits (a *short key*) is padded to 256 bits by appending a single 1 bit and then as many 0 bits as needed.

The full 256-bit key K is written as 8 32-bit words $w_{-8}, \ldots, w_{-1}$ and expanded into 132 32-bit words $w_0, \ldots, w_{131}$ (collectively called the *prekey*) using the recurrent expression

$$w_i = (w_{i-8} \oplus w_{i-5} \oplus w_{i-3} \oplus w_{i-1} \oplus \phi \oplus i) <<< 11,$$

where $\oplus$ denotes XOR and $<<<$ a left bit rotate.

A bitslice-mode S-box (see later) is applied to 4 words of the prekey at a time, producing 33 128-bit subkeys (round keys). The concrete S-box is changed for every round key, starting with S_3 and going forward modulo 8.

Implementation

The algorithm specification describes two possible approaches to implementing Serpent.

Formally, the Serpent round operates on state consisting of 32 4-bit chunks of data (the same S-box is applied 32 times in parallel). Similarly to DES, this is efficient in hardware, but not in software.

In the so-called *"bitslice mode"*, the state is rearranged into 4 32-bit words, where the first bits of each word correspond to the first 4-bit chunk in the formal description, etc.

A caveat not explicitly stated in the specification is that **Serpent treats input data (i.e. plaintexts and ciphertexts) as already converted to the bitslice representation**.

This means that when the "canonical" implementation is chosen, the input and output bits (as well as individual subkeys) **have to be permuted back from the default bitslice mode**. This makes sense given the fact that in a hardware implementation, permuting bits is basically free, while in software, bit permutations are slow and painful to implement.

Data representation

In Serpent, for some godforsaken reason, **everything is backwards**.

Externally, blocks are accepted and presented as *128-bit numbers*; the (user) key is interpreted as a *sequence of bits*. (An incredible formalization choice for a cipher that might commonly be implemented on a system where 128-bit ints aren't a thing.)

The user key is expanded to 256 bits like described before. On the other hand, the key expansion aceepts the key as an array of 8 32-bit words. *Naturally*, this is done by taking bits 0-31 of the input sequence and *setting bit 0 as the least significant bit of the first key word and bit 31 as the most significant* (and similarly for the other words). Note that not only are the key bytes interpreted as little-endian, the **bit order is reversed too**! So, for example, the correct way to pad a 128-bit key (in C) is this:

```
void serpent_pad_128bit_key(uint8_t key[32]){  key[16] = 0x01;  memset(&key[17], 32 - 17, 0x00);}
```

and **not** this:

```
void bad_pad_128bit_key(uint8_t key[32]){  key[16] = 0x80;  memset(&key[17], 32 - 17, 0x00);}
```

References

- Serpent algorithm specification paper

Revision #8
Created 19 March 2025 22:09:31 by Annatar
Updated 26 March 2025 19:03:42 by Annatar