The Pelican MAC Function

Joan Daemen[1] and Vincent Rijmen[2,3]

[1] STMicroelectronics Belgium
joan.daemen@st.com
[2] IAIK, Graz University of Technology
vincent.rijmen@iaik.tugraz.at
[3] Cryptomathic A/S

Abstract. We present new a MAC function based on Rijndael that is a factor 2.5 more efficient than CBC-MAC with Rijndael, while providing a comparable claimed security level. This MAC function has the Alred construction [5] and can be seen as an optimized version of Alpha-MAC introduced in the same paper.

1 Introduction

Message Authentication Codes (MAC functions) are symmetric primitives, used to ensure authenticity of messages. They take as input a secret key and the message, and produce as output a short tag. The basic property of a MAC function is that it provides an unpredictable mapping between messages and the tag for someone who does not know, or only partially knows the key.

We propose here Pelican as a new MAC function. The design is based on the Alred construction, which was presented in [5]. Our reasoning about the security of Alpha-MAC, the concrete design presented in [5], resulted in several suggestions for modifications. The modifications result in a simpler design, because the *injection layout* map is removed. Secondly, the new design has a slightly better performance.

Pelican is based on Rijndael [4]. We have chosen for Rijndael mainly because we expect it to be widely available thanks to its status as the AES standard. Additionally, Rijndael is efficient in hardware and software and it has withstood intense public scrutiny very well since its publication [3].

The purpose of this paper is to provide to the cryptographic community a clear description of a MAC function that we announced at the Fast Software Encryption 2005 workshop under the (working) name Beta-MAC.

We specify Pelican in Section 2. In Section 3, we briefly repeat the security claims introduced in [5], and in Section 4 we recall the provable security properties of the Alred construction, that apply to Pelican. We discuss briefly internal collisions of Pelican in Section 5 and performance in Section 6. We conclude in Section 7.

2 Specification

PELICAN is an ALRED construction with Rijndael [4], restricted to a block length of 128 bits, as underlying block cipher. As Rijndael, PELICAN supports keys of 16, 20, 24, 28 and 32 bytes. For the key lengths 16, 24 and 32 bytes, Rijndael coincides with AES [1]. PELICAN can take a message m of any length and generates tags with length up to 128 bits.

In the PELICAN algorithm we can distinguish a number of steps. First the message is padded and split in message words. These message words are applied to a *state* that is initialized using the key and that afterwards undergoes a final step again using the key. More exactly:

Message padding and splitting: pad the message by appending a single 1 bit followed by the minimum number of 0 bits so that the resulting length is a multiple of 128 bits (padding method 2 in [2]). Split the result in 128-bit *message words* $x_1, x_2, \ldots x_q$.

State initialization: fill the 128-bit *state* with binary zeroes and subsequently apply Rijndael encryption to it using the key.

Chaining: XOR the first message word x_1 to the state. For each additional message word x_i, apply to the state the iteration function and then XOR the message word x_i to the state. The iteration function consists of four Rijndael rounds with round keys set to 0.

Finalization: Apply Rijndael encryption to the state using the key and form the tag by taking the first ℓ_m bits of the state.

PELICAN is illustrated in Figure 1.

3 Security claims

PELICAN is an iterative MAC function. Iterative MAC functions have the disadvantage that different messages may be found that lead to the same value of the state before the final transformation. This is called an *internal collision* [6]. We have proposed in [5] a set of three orthogonal security claims for iterative MAC functions that reflect this limitation. We repeat them here for clarity.

The claims are formulated in terms of three dimension parameters: the tag length ℓ_m, the key length ℓ_k and the *capacity* ℓ_c. The capacity is the size of the internal memory of a random map with the same probability for internal collisions as the MAC function and the designer must fix its value used in the security claim.

Claim 1 *The probability of success of any forgery attack not involving key recovery or internal collisions is* $2^{-\ell_m}$.

Claim 2 *There are no key recovery attacks faster than exhaustive key search, i.e. with an expected complexity less than* $2^{\ell_k - 1}$ *MAC function executions.*

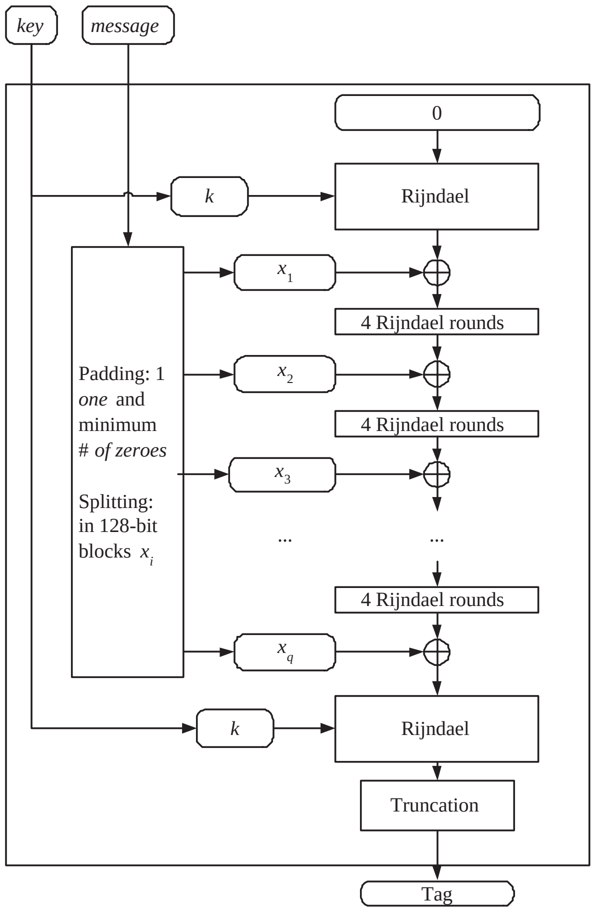

Fig. 1. Block scheme of Pelican.

Claim 3 *The probability that an internal collision occurs in a set of A ((adaptively) chosen message, tag) pairs, with $A < 2^{\ell_c/2}$, is not above* $1 - \exp(-A^2/2^{\ell_c+1})$.

Pelican should satisfy these three claims for the following range of values of the dimension parameters:

– Tag length ℓ_m: any value up to 128 bits.
– Key length ℓ_k: 128, 160, 192, 224 and 256 bits.
– Capacity ℓ_c: 120 bits

4 Provability

Pelican has the Alred structure [5]. For this structure, the security with respect to forgery in the absence of collisions and with respect to key recovery can be provably reduced to security properties of the used block cipher. This results in the following theorems for Pelican.

Theorem 1. *Every key recovery attack on* Pelican*, requiring t (adaptively) chosen messages, can be converted to a key recovery attack on Rijndael, requiring $t+1$ adaptively chosen plaintexts.*

Theorem 2. *Every forgery attack on* Pelican *not involving internal collisions, requiring t (adaptively) chosen messages, can be converted to a ciphertext guessing attack on Rijndael , requiring $t+1$ adaptively chosen plaintexts.*

For the proofs of these theorems we refer to [5].

5 On internal collisions in Pelican

We have based the value of the capacity used in our security claims on a preliminary analysis of generating internal collisions for Pelican. We have considered two types of internal collisions: extinguishing differentials and fixed points. We refer to [5] for more background. The value of the capacity is based on a preliminary analysis. We are continuing the research. New results may lead us to modify the capacity.

6 Performance

In this section we express the performance of Pelican in terms of Rijndael operations, more particularly, the Rijndael key schedule and the Rijndael encryption operation. This allows to use Rijndael (or AES) benchmarks for software implementations on any platform or even hardware implementations to get a pretty good idea on the performance of Pelican. We restrict ourselves to the case of 128-bit keys.

In a 32-bit implementation, one iteration of Pelican corresponds roughly to 4 rounds of Rijndael, hence roughly 4/10 of a Rijndael encryption. It is actually better because the XORs with 0 in the round key addition don't have to be implemented. Some implementations of Rijndael recompute the round keys for every encryption. This overhead is not present in Pelican. As the first message word is simply XORed without additional rounds, the message processing of the first message word must be subtracted from the message processing. Using these rough approximations, we can state that MACing a message requires:

setup: 1 Rijndael key schedule and 1 Rijndael encryption,
message processing: 0.4 Rijndael encryptions per 16-byte message word,
finalization: 0.6 Rijndael encryptions.

Hence, for long messages Pelican is more than a factor 2.5 faster than Rijndael encryption, for short messages the minimum cost is 1 Rijndael encryption for generating a tag and 1 additional Rijndael encryption and key schedule at key setup.

7 Conclusions

We introduced Pelican, a new MAC function based on the Alred construction. It is simpler and faster than its predecessor Alpha-MAC. The Alred construction comes with some security proofs, which are also applicable to this primitive. Further analysis of the security of this primitive is underway.

References

1. *Federal Information Processing Standard 197, Advanced Encryption Standard (AES)*, National Institute of Standards and Technology, U.S. Department of Commerce, November 2001.
2. *ISO/IEC 9797-1, Information technology - Security Techniques - Message Authentication Codes (MACs) - Part 1: Mechanisms using a block cipher,* ISO 1999.
3. Joan Daemen and Vincent Rijmen, "AES Proposal: Rijndael," *AES Round 1 Technical Evaluation CD-1: Documentation*, National Institute of Standards and Technology, Aug 1998.
4. Joan Daemen and Vincent Rijmen, *The design of Rijndael — AES, The Advanced Encryption Standard,* Springer-Verlag, 2002.
5. Joan Daemen and Vincent Rijmen, "A new MAC Construction Alred and a Specific Instance Alpha-MAC,", *Fast Software Encryption 2005, LNCS* H. Gilbert, H. Handschuh, Eds., Springer-Verlag, to appear.
6. Bart Preneel and Paul C. van Oorschot, "On the security of iterated Message Authentication Codes," *IEEE Trans. on Information Theory, Vol. IT45, No. 1,* 1999, pp. 188-199.