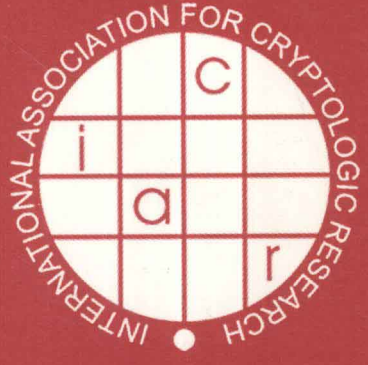
Henri Gilbert
Helena Handschuh (Eds.)

LNCS 3557

Fast Software Encryption

12th International Workshop, FSE 2005
Paris, France, February 2005
Revised Selected Papers



Springer

Henri Gilbert Helena Handschuh (Eds.)

Fast Software Encryption

12th International Workshop, FSE 2005
Paris, France, February 21-23, 2005
Revised Selected Papers

Springer

Volume Editors

Henri Gilbert
France Telecom, 92794 Issy les Moulineaux, France
E-mail: henri.gilbert@francetelecom.com

Helena Handschuh
Gemplus SA, Issy-les-Moulineaux, France
E-mail: Helena.Handschuh@gemplus.com

Library of Congress Control Number: 2005928340

CR Subject Classification (1998): E.3, F.2.1, E.4, G.2, G.4

ISSN	0302-9743
ISBN-10	3-540-26541-4 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-26541-2 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11502760 06/3142 5 4 3 2 1 0

Preface

The Fast Software Encryption 2005 Workshop was the twelfth in a series of annual workshops on symmetric cryptography, sponsored for the fourth year by the International Association for Cryptologic Research (IACR). The workshop concentrated on all aspects of fast primitives for symmetric cryptology, including the design, cryptanalysis and implementation of block and stream ciphers as well as hash functions and message authentication codes. The first FSE workshop was held in Cambridge in 1993, followed by Leuven in 1994, Cambridge in 1996, Haifa in 1997, Paris in 1998, Rome in 1999, New York in 2000, Yokohama in 2001, Leuven in 2002, Lund in 2003, and New Delhi in 2004.

This year, a total of 96 submissions were received. After an extensive review by the Program Committee, 30 submissions were accepted. Two of these submissions were merged into a single paper, yielding a total of 29 papers accepted for presentation at the workshop. Also, we were very fortunate to have in the program an invited talk by Xuejia Lai on "Attacks and Protection of Hash Functions" and a very entertaining rump session that Bart Preneel kindly accepted to chair. These proceedings contain the revised versions of the accepted papers; the revised versions were not subsequently checked for correctness.

We are very grateful to the Program Committee members and to the external reviewers for their hard work. Each paper was refereed by at least three reviewers, and at least five reviewers in the case of papers (co-)authored by Program Committee members; eventually, an impressive total of 334 reviews was produced. Special thanks are also due to the members of the Local Organizing Committee, Côme Berbain, Olivier Billet (who designed the FSE 2005 Web pages and assembled the preproceedings), Julien Brouchier (who managed the submission and Webreview servers), Stanislas Francfort, Aline Gouget, Françoise Levy, Pierre Loidreau, and Pascal Paillier (who managed on-site registration), for their generous efforts and strong support.

Many thanks to Kevin McCurley for handling the registration server, to Patrick Arditti, Virginie Berger and Claudine Campolunghi for providing assistance with the registration process, and to the research group COSIC of the K.U.Leuven for kindly providing their Webreview software.

Last but not least, we would like to thank the conference sponsors France Telecom, Gemplus, and Nokia for their financial support, DGA and ENSTA for hosting the conference on their premises, and all submitters and workshop participants who made this year's workshop such an enjoyable event.

April 2005 Henri Gilbert and Helena Handschuh

FSE 2005

February 21–23, 2005, Paris, France

Sponsored by
the International Association for Cryptologic Research (IACR)

Program and General Chairs

Henri Gilbert ..... France Telecom, France
Helena Handschuh ..... Gemplus, France

Program Committee

Kazumaro Aoki ..... NTT, Japan
Steve Babbage ..... Vodafone, UK
Eli Biham ..... Technion, Israel
Anne Canteaut ..... INRIA, France
Don Coppersmith ..... IBM Research, USA
Joan Daemen ..... STMicroelectronics, Belgium
Thomas Johansson ..... Lund University, Sweden
Antoine Joux ..... DGA and Université de Versailles, France
Xuejia Lai ..... Shanghai Jiaotong University, China
Stefan Lucks ..... Universität Mannheim, Germany
Mitsuru Matsui ..... Mitsubishi Electric, Japan
Willi Meier ..... FH Aargau, Switzerland
Kaisa Nyberg ..... Nokia, Finland
Bart Preneel ..... K.U.Leuven, Belgium
Matt Robshaw ..... Royal Holloway, University of London, UK
Palash Sarkar ..... Indian Statistical Institute, India
Serge Vaudenay ..... EPFL, Switzerland
Moti Yung ..... Columbia University, USA

Local Organizing Committee

Côme Berbain, Olivier Billet, Julien Brouchier, Stanislas Francfort, Henri Gilbert, Aline Gouget, Helena Handschuh, Françoise Levy, Pierre Loidreau, Pascal Paillier

Industry Sponsors

France Telecom
Gemplus SA
Nokia

External Referees

Frederik Armknecht
Daniel Augot
Gildas Avoine
Thomas Baignères
Elad Barkan
An Braeken
Claude Carlet
Pascale Charpin
Sanjit Chatterjee
Rafi Chen
Debra L. Cook
Christophe De Cannière
Orr Dunkelman
Matthieu Finiasz
Pierre-Alain Fouque
Soichi Furuya
Louis Granboulan
Tor Helleseth
Shoichi Hirose
Tetsu Iwata
Pascal Junod
Charanjit Jutla
Grigory Kabatyanskiy
Jonathan Katz
Alexander Kholosha
Yuichi Komano
Matthias Krause
Ulrich Kühn
Simon Künzli
Shreekanth Laksmeshwar
Joseph Lano
Cédric Lauradoux
Yi Lu
Marine Minier
Håvard Molland
Jean Monnerat
Shiho Moriai
Frédéric Muller
Sean Murphy
Philippe Oechslin
Kenji Ohkuma
Katsuyuki Okeya
Elisabeth Oswald
Souradyuti Paul
Gilles Piret
Zulfikar Ramzan
Vincent Rijmen
Akashi Satoh
Takeshi Shimoyama
Taizo Shirai
François-Xavier Standaert
Dirk Stegemann
Henk van Tilborg
Hiroki Ueda
Kan Yasuda
Erik Zenner

Table of Contents

Stream Ciphers II

Hash Functions

Modes of Operation

Stream Ciphers III

Block Ciphers II

Implementations

A New MAC Construction ALRED and a Specific Instance ALPHA-MAC

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 a new way to construct a MAC function based on a block cipher. We apply this construction to AES resulting in a MAC function that is a factor 2.5 more efficient than CBC-MAC with AES, while providing a comparable claimed security level.

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.

Basically, there are three approaches for designing MACs. The first approach is to design a new primitive from scratch, as for instance MAA [15] and, more recently, UMAC [8]. This approach allows to optimize the security-performance trade-off. The second approach is to define a new *mode of operation* for existing primitives. In this category, we firstly have numerous variants based on the CBC encryption mode for block ciphers, e.g. CBC-MAC [5], OMAC [16], TMAC [22], XCBC [9], and RMAC [17]. Secondly, there are the designs based on an unkeyed hash function: NMAC, HMAC [7, 4]. Finally, one can design new MACs using components from existing primitives, e.g. MDx-MAC [24] and Two-Track MAC [10].

In this paper, we propose a new MAC design method which belongs in the third category, cf. Section 3. We also present a concrete construction in Section 5. Before going there, we start with a discussion of security requirements for MACs and we present a new proposal for MAC security claims in Section 2. We discuss internal collisions for the new model in Section 4, and for the concrete construction in Section 6. Section 7 contains more details on extinguishing differentials, a special case of internal collisions. We briefly discuss performance in Section 8 and conclude in Section 9.

* This researcher was supported financially by the A-SIT, Austria.

H. Gilbert and H. Handschuh (Eds.): FSE 2005, LNCS 3557, pp. 1–17, 2005.

2 Iterative MAC Functions and Security Claims

A MAC function maps a key-message pair to a 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. Usually, one defines a number of design objectives that a cryptographic primitive of a given type must satisfy in order to be considered as secure. For MAC functions, we find the following design objectives in [23–Table 9.2]:

- *Key non-recovery*: the expected complexity of any key recovery attack is of the order of 2^{ℓ_k} MAC function executions.
- *Computation resistance*: there is no forgery attack with probability of success above $\max(2^{-\ell_k}, 2^{-\ell_m})$.

Here ℓ_k is the key length and ℓ_m the tag length. By forgery one means the generation of a message-tag pair (m, t) using only information on pairs (m_i, t_i) with $m \neq m_i$ for all i.

2.1 Iterative MAC Functions

Most practical MAC functions are *iterative*. An iterative MAC function operates on a working variable, called the *state*. The message is split up in a sequence of message blocks and after a (possibly keyed) initialization the message blocks are sequentially injected into the state by a (possibly keyed) iteration function. Then a (possibly keyed) final transformation may be applied to the state resulting in the tag.

Iterative MAC functions can be implemented in hardware or software with limited amount of working memory, irrespective of the length of the input messages. They 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* [26].

2.2 Internal Collisions

Internal collisions can be used to perform forgery. Assume we have two messages m_1 and m_2 that result in an internal collision. Then for any string m_3 the two messages $m_1 \| m_3$ and $m_2 \| m_3$ have the same tag value. So given the tag of any message $m_1 \| m_3$, one can forge the tag of the message $m_2 \| m_3$. Internal collisions can often be used to speed up key recovery as well [25]. If the number of bits in the state is n, finding an internal collision takes at most $2^n + 1$ known pairs. If the state transformation can be modeled by a random transformation, one can expect to find a collision with about $2^{n/2}$ known pairs due to the birthday paradox. One may consider to have a final transformation that is not reversible to make the detection of internal collisions infeasible. However, as described in Appendix A, this is impossible.

The presence of internal collisions makes that even the best iterative MAC function cannot fulfill the design objectives given above: if the key is used to

generate tags over a very large number of messages, an internal collision is likely to occur and forgery is easy.

For many MAC schemes based on CBC-MAC with the DES as underlying block cipher, internal collisions can be used to retrieve the key: the ISO 9797 [5] schemes are broken in [11, 18]. More sophisticated variants like Retail MAC [1] and MacDES [19] are broken in [25], respectively [12, 13].

One approach to avoid the upper limit due to the birthday paradox in iterative MAC functions is *diversification*. The MAC function has next to the message and the key a third input parameter that serves to diversify the MAC computation to make the detection of internal collisions impossible. For the proofs of security that accompany these schemes, the implementation of a tag generating device must impose that its value is non-repeating or random. This method has several important drawbacks. First of all, not only the tag must be sent along with the message, but also this third parameter, typically doubling the overhead. In case of a random value, this puts the burden on the developer of the tag generating device to implement a cryptographic random generator, which is a non-trivial task. Moreover the workload of generating the random value should be taken into account in the performance evaluation of the primitive. In case of a non-repeating value the MAC function becomes stateful, i.e., the tag generation device must keep track of a counter or multiple counters and guarantee that these counters cannot be reset. But in some cases even the randomization mechanism itself introduces subtle flaws. The best known example of a randomized MAC is RMAC [17] cryptanalyzed in [21].

Another way to avoid internal collisions is to impose an upper bound on the number of tags that may be generated with a given key. If this upper bound is large enough it does not impose restrictions in actual applications. This is the approach we have adopted in this paper.

2.3 A Proposal for New Security Claims

We formulate a set of three orthogonal security claims that an iterative MAC function should satisfy to be called secure.

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^{ℓ_k-1} *MAC function executions.*

We model the effect of internal collisions by a third dimension parameter, 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.

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* $1 - \exp(-A^2/2^{\ell_c+1})$.

Note that for $A < 1/4 \times 2^{\ell_c/2}$ we have $1 - \exp(-A^2/2^{\ell_c+1}) \approx A^2/2^{\ell_c+1}$. In the best case the capacity ℓ_c is equal to the number of bits of the state. It is up to the designer to fix the value of the capacity ℓ_c used in the security claim.

3 The ALRED Construction

We describe here a way to construct a MAC function based on an iterated block cipher. The key length of the resulting MAC function is equal to that of the underlying block cipher, the length of the message must be a multiple of ℓ_w bits, where ℓ_w is a characteristic of a component in the MAC function. In our presentation, we use the term *message word* to indicate ℓ_w-bit message blocks and call ℓ_w the *word length.*

3.1 Definition

The ALRED construction consists of a number of steps:

1. Initialization:
   (a) Initialize the state with the all-zero block.
   (b) Apply the *block cipher* to the state.
2. Chaining: for each message word perform an iteration:
   (a) Map the bits of the message word to an *injection input* that has the same dimensions as a sequence of r round keys of the block cipher. We call this mapping the *injection layout.*
   (b) Apply a sequence of r *block cipher round functions* to the state, with the injection input taking the place of the round keys.
3. Final transformation:
   (a) Apply the *block cipher* to the state.
   (b) Truncation: the tag is the first ℓ_m bits of the state.

Let the message words be denoted by x_i, the state after iteration i by y_i, the key by k and the tag by z. Let f denote the iteration function, which consists of the combination of the injection layout and the sequence of r block cipher round functions. Then we can write:

$$y_0 = \mathrm{Enc}_k(0) \tag{1}$$

$$y_i = f(y_{i-1}, x_i), \quad i = 1, 2, \ldots, q \tag{2}$$

$$z = \mathrm{Trunc}(\mathrm{Enc}_k(y_q)) \tag{3}$$

The construction is illustrated in Figure 1 for the case $r = 1$. With this approach, the design of the MAC function is limited to the choice of the block cipher, the number of rounds per iteration r, the injection layout and ℓ_m. The goal is to choose these such that the resulting MAC function fulfills the security claims for iterated MAC functions for some value of ℓ_m and ℓ_c near the block length.

3.2 Motivation

Prior to the chaining phase, the state is initialized to zero and it is transformed by applying the block cipher, resulting in a state value unknown to the attacker. In the chaining phase every iteration injects ℓ_w message bits into the state with an

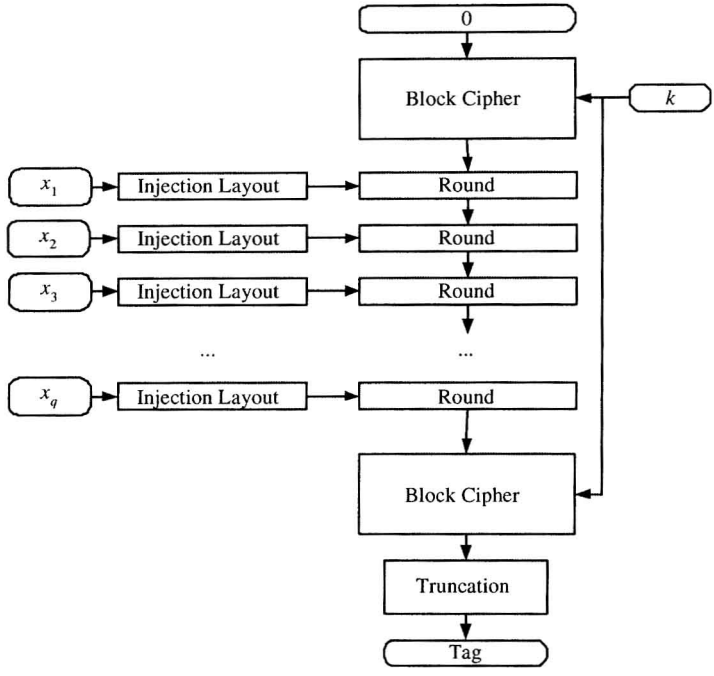


Fig. 1. Scheme of the ALRED construction with $r = 1$

unkeyed iteration function. Without the block cipher application in the initialization, generating an internal collision would be similar to finding a collision in an unkeyed hash function which can be conducted without known message tag pairs. The initial block cipher application makes the difference propagation through the chaining phase, with its nonlinear iteration function, depend on the key.

The iteration function consists of r block cipher rounds where message word bits are mapped onto the round key inputs. The computational efficiency of ALRED depends on the word length. Where in CBC based constructions for long messages there is one block cipher execution per block, ALRED takes merely r rounds per word. Clearly, the performance of ALRED becomes interesting if the word length divided by r is larger than the block length divided by the number of rounds of the block cipher.

Decreasing the message word length with respect to the round key length makes the MAC function less efficient, but also reduces the degrees of freedom available to an attacker to generate internal collisions (see Section 6.1 for an example). Another way to reduce these degrees of freedom is to have the message words first undergo a message schedule, and apply the result as round keys. This is similar to the key schedule in a block cipher, the permutation of message words between the rounds in MD4 [27] or the message expansion in SHA-1 [2]. However, such a message schedule also introduces need for additional memory and additional workload per iteration. Therefore, and for reasons of simplicity,

we decided to limit the message injection to a simple layout. Limiting the word length and carefully choosing the injection layout allows to demonstrate powerful upper bounds on the probability of sets of known or chosen messages with any chosen difference leading to an internal collision.

3.3 Provability

ALRED has some similarity to constructions based on block ciphers in CBC mode. The modes typically come with security proofs that make abstraction of the internal structure of the used cryptographic primitive. In this section we prove that an ALRED MAC function is as strong as the underlying block cipher with respect to key recovery and, in the absence of internal collisions, is resistant against forgery if the block cipher is resistant against ciphertext guessing.

Observation: The proofs we give are valid for any chaining phase that transforms y_0 into y_{final} parameterized by a message. In the proofs we denote this by $y_{\text{final}} = F_{\text{cf}}(y_0, m)$.

Theorem 1. *Every key recovery attack on* ALRED, *requiring t (adaptively) chosen messages, can be converted to a key recovery attack on the underlying block cipher, requiring $t+1$ adaptively chosen plaintexts.*

Proof: Let $\mathcal{A}$ be an attack requiring the t tag values corresponding to the t (adaptively) chosen messages m_j, yielding the key. Then, the attack on the underlying block cipher works as follows.

1. Request $c_0 = \text{Enc}(k, 0)$, where '0' denotes the all-zero plaintext block.
2. For $j = 1$ to t, compute $p_j = F_{\text{cf}}(c_0, m_j)$.
3. For $j = 1$ to t, request $c_j = \text{Enc}(k, p_j)$.
4. Input the tag values $\text{Trunc}(c_j)$ to $\mathcal{A}$ and obtain the key.

□

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

Proof: Let $\mathcal{B}$ be an attack, not involving internal collisions, requiring the t tag values corresponding to the t (adaptively) chosen messages m_j yielding a forged tag for the message m. Then, the ciphertext guessing attack on the underlying block cipher works as follows.

1. Request $c_0 = \text{Enc}(k, 0)$, where '0' denotes the all-zero plaintext block.
2. For $j = 1$ to t, compute $p_j = F_{\text{cf}}(c_0, m_j)$.
3. For $j = 1$ to t, request $c_j = \text{Enc}(k, p_j)$.
4. Input the tag values $\text{Trunc}(c_j)$ to $\mathcal{B}$ and obtain the tag for the message m.
5. Compute $p = F_{\text{cf}}(c_0, m)$.
6. If there is a j for which $p = p_j$, then $\mathcal{B}$ has generated an internal collision, which conflicts with the requirement on $\mathcal{B}$. Otherwise, input the tag values $\text{Trunc}(c_j)$ to $\mathcal{B}$ and obtain the tag, yielding the truncated ciphertext of p.

□

3.4 On the Choice of the Cipher

One may use any block cipher in the ALRED construction. The block length imposes an upper limit to the capacity ℓ_c relevant in the number of tags that may be generated with the same key before an internal collision occurs. When using ciphers with a block length of 64 bits as (Triple) DES and IDEA, the number of tags generated with the same keys should be well below 2^{32}.

The use of the round function for building the iteration function restricts the ALRED construction somewhat. Ciphers that are well suited in this respect are (Triple) DES, IDEA, Blowfish, Square, RC6, Twofish and AES. An ALRED construction based on Serpent, with its eight different rounds, would typically have $r = 8$, with the iteration function consisting of the eight Serpent rounds. MARS with its non-uniform round structure is less suited for ALRED. The choice of the injection layout is a design exercise specific for the round function of the chosen cipher. Note that whatever the choice of the underlying cipher, the strength of the ALRED construction with respect to key search is that of the underlying cipher.

4 On Internal Collisions in ALRED

In general, any pair of message sequences, possibly of different length, that leads to the same value of the internal state is an internal collision. We see two approaches to exploit knowledge of the iteration function to generate internal collisions. The first is to generate pairs of messages of equal length that have a difference (with respect to some group operation at the choice of the attacker) that may result in a zero difference in the state after the difference has been injected. We call this *extinguishing differentials*. The second is to insert in a message a sequence of words that do not impact the state. We call this *fixed points*.

4.1 Extinguishing Differentials

Finding high-probability extinguishing differentials is similar to differential cryptanalysis of block ciphers. In differential cryptanalysis the trick is to find an input difference that leads to an output difference with high probability. For an iterative MAC function, the trick is to find an extinguishing differential with high probability. Resistance against differential cryptanalysis is often cited as one of the main criteria for the design of the round function of block ciphers. Typically the round function is designed in such a way that upper bounds can be demonstrated on the probability of differentials over a given number of rounds. One may design MAC functions in a similar way: design the iteration function such that upper bounds can be demonstrated on the probability of extinguishing differentials. In ALRED the only part of the iteration function to be designed is the injection layout. So the criterion for the choice of the injection layout is the upper bound on the probability of extinguishing differentials.

4.2 Fixed Points

Given a message word value x_i, one may compute the number of state values that are invariant under the iteration function $y_i = f(y_{i-1}, x_i)$, called *fixed points*. If the number of fixed points is w, the probability that inserting the message word x_i in a message will not impact its tag is $w \times 2^{-n}$ with n the block length.

We can try to find the message word value $x_{\max}$ with the highest number of fixed points. If this maximum is $w_{\max}$, inserting $x_{\max}$ in a message and assuming that the resulting message has the same tag, is a forgery attack with success probability $w_{\max} \times 2^{-n}$. Since Claim 3 requires that this probability be smaller than $1 - \exp(-2^2/2^{\ell_c+1}) = 1 - \exp(-(2^{1-\ell_c})) \approx 2^{1-\ell_c}$, this imposes a limit to the capacity: $\ell_c < n + 1 - \log_2 w_{\max}$.

If the iteration function can be modeled as a random permutation, the number of fixed points has a Poisson distribution with $\lambda = 1$. The expected value of $w_{\max}$ depends on the number of different iteration functions with a given message word, i.e. the word length ℓ_w. For example, the expected $w_{\max}$ values for 16, 32, 64 and 128 bits are 8, 12, 20 and 33 respectively. However, if $r = 1$, the iteration function is just a round function of a block cipher and it may not behave as a random function in this respect. If the round function allows it, one may determine the number of fixed points for a number of message word values to determine whether the Poisson distribution applies.

One may consider the number of fixed points under a sequence of g rounds. In the random model, the expected value of $w_{\max}$ over all possible sequences of g message words now is determined by the total number of messagebits injected in the g rounds. For most round functions determining the number of fixed points given the message word values is hard even for $g = 2$. However, for multiple iterations it is very likely that the random model will hold. The value of $w_{\max}$ grows with g but actually finding message word sequences with a number of fixed points of the order $w_{\max}$ becomes quickly infeasible as g grows. If we consider a sequence of iterations taking 500 message bits (for example 10 iterations taking each 50 message bits), the expected value of $w_{\max}$ is 128. In conclusion, if analysis of the iteration function confirms that the number of fixed points has a Poisson distribution, taking $\ell_c \leq n - 8$ provides a sufficient security margin with respect to forging using fixed points.

5 ALPHA-MAC

ALPHA-MAC is an ALRED construction with AES [3] as underlying block cipher. As AES, ALPHA-MAC supports keys of 16, 24 and 32 bytes. Its iteration function consists of a single round, its word length is 4 bytes and the injection layout places these bytes in 4 byte positions of the AES state. We have chosen AES mainly because we expect AES to be widely available thanks to its status as a standard. Additionally, AES is efficient in hardware and software and it has withstood intense public scrutiny very well since its publication as Rijndael [14].