A TIGHT BOUND FOR EXHAUSTIVE KEY SEARCH ATTACKS AGAINST MESSAGE AUTHENTICATION CODES *

Vinícius G. P. de Sá[1], Davidson R. Boccardo[2], Luiz Fernando Rust[2] and Raphael C. S. Machado[2,1]

Abstract. A message authentication code (MAC) is a function that takes a message and a key as parameters and outputs an authentication of the message. MAC are used to guarantee the legitimacy of messages exchanged through a network, since generating a correct authentication requires the knowledge of the key defined secretly by trusted parties. However, an attacker with access to a sufficiently large number of message/authentication pairs may use a brute force algorithm to infer the secret key: from a set containing initially all possible key candidates, subsequently remove those that yield an incorrect authentication, proceeding this way for each intercepted message/authentication pair until a single key remains. In this paper, we determine an exact formula for the expected number of message/authentication pairs that must be used before such form of attack is successful, along with an asymptotical bound that is both simple and tight. We conclude by illustrating a modern application where this bound comes in handy, namely the estimation of security levels in reflection-based verification of software integrity.

1991 Mathematics Subject Classification. 94A60

Keywords and phrases: Cryptography; Message Authentication Code; Asymptotic Analysis

* *This is a manuscript by the authors. The final article was published by EDP Sciences with DOI number 10.1051/ita/2012025.*

[1] Depto. de Ciência da Computação — Univ. Federal do Rio de Janeiro — Brazil; e-mail: vigusmao@dcc.ufrj.br

[2] Inmetro — National Institute of Metrology, Quality and Technology — Brazil; e-mail: drboccardo@inmetro.gov.br & lfrust@inmetro.gov.br & rcmachado@inmetro.gov.br

Introduction

Granting the legitimacy of messages exchanged through unsafe channels is primordial for several applications. A common security practice is to have each message be sent along with an *authentication*, so to say a digital fingerprint of the message. Such authentication is usually generated by a hash-based algorithm called a *message authentication code* (MAC), whose input comprises not only the message itself but also a predefined *key* only known to the communicating parties. In order to verify that the message has not been corrupted, the receiver computes its authentication anew and checks whether it matches the received version. Although MAC specifications are generally considered to be of public knowledge, the point is that an adversary, willing to maliciously corrupt a message, would have to either recompute its authentication—which requires knowledge of the key—or try to guess it. Indeed, both kinds of attacks—attempts to disclose the key and attempts to guess the authentication directly—do exist and demand attention.

We focus on the *exhaustive key search* (EKS) attack model, wherein the attacker aims to infer the secret MAC key out of a set of intercepted message/authentication pairs. Even though some intuitive upper bounds on the complexity of the EKS are well accepted [3], lower bounds are considered to be scarce [2]. We emphasize that, while upper bounds are important from the point of view of the attacker (to estimate the maximum amount of time it takes an attack to be successful), lower bounds are fundamental for the defender (to build a resilient defense strategy).

The paper is structured as follows. In Section 1, we define a MAC and the EKS attack model. In Section 2, we give our main contribution, namely a tight bound for its expected time complexity. Section 3 describes an application in software integrity verification, and in Section 4 we present some concluding remarks.

1. MAC and the exhaustive key search attack

A message authentication code is formally a function $f : M \times K \to A$ that takes two parameters—a message $\rho \in M$ and a key $\kappa \in K$—and outputs an authentication $\alpha \in A$. The sets M, K and A are, respectively, the message space, the key space and the authentication space of the MAC. An usual message exchanging protocol is such that all sent messages are accompanied by their authentications, as computed by the sender using a predefined key that is kept secret. In order to check the message integrity, the receiver computes its authentication from scratch using the same key agreed upon with the sender, and compares it with the received authentication.

In the *exhaustive key search* attack model [3], as it is known in the computer security community, an opponent gathers a number of message/authentication pairs (ρ_j, α_j), one at each *round* j, removing from a set K'_j of *candidate* keys every key $\kappa_i \in K'_j$ such that $f(\rho_j, \kappa_i) \neq \alpha_j$. This approach yields a sequence $K = K'_1 \supseteq K'_2 \supseteq \ldots \supseteq K'_t$, when $|K'_t| = 1$ reveals the key. The usual algorithm of the EKS attack against a MAC is illustrated in Figure 1.

```
EKS_Attack
input     MAC function f : M × K → A
output    MAC key κ
1.        K' ← K
2.        j ← 0
3.        repeat
3.1.        j ← j + 1
3.2.        obtain message/authentication pair (ρ_j, α_j) // MAC query
3.3.        for each κ_i ∈ K'
3.3.1         if f(ρ_j, κ_i) ≠ α_j // MAC computation
3.3.1.1         K' ← K' \ {κ_i}
4.        until |K'| = 1
5.        return κ ∈ K'
```

FIGURE 1. Exhaustive key search attack against a MAC.

There are two crucial operations in the EKS algorithm: the obtainment of each message/authentication pair (ρ_j, α_j) in line 3.2 (often referred to as a *MAC query*), and the evaluation of $f(\rho_j, \kappa_i)$ in line 3.3.1 (or *MAC computation*). Its complexity analysis must therefore take into account (i) the number X of disclosed message/authentication pairs, and (ii) the number Y of times the algorithm computes f before it halts. In the next section, we determine these values and obtain a general formula for the time complexity of an EKS attack against a MAC.

2. Complexity of the EKS

Let f be a MAC with message space M, key space K and authentication space A. Both the key and the authentication normally have a fixed bitlength, say k and a, respectively, whereas the message has arbitrary bitlength. We therefore have $|K| = 2^k$ and $|A| = 2^a$.

First, we model the attack using a simple card game analogy. There are $|K|$ distinct *candidate* piles of cards (one pile for each possible key $\kappa_i \in K$), where each card has one among $|A|$ possible face values with equiprobability.[1] A special pile, called the *query* pile, is kept apart, and its cards can be revealed one-by-one in a timely fashion. At each round j of the game, the j-th card a_j of the query pile is revealed and all candidate piles whose j-th card is not a match for a_j are eliminated. The goal is to find which pile, among those $|K|$ undisclosed candidate piles, has a card sequence that is identical to that in the query pile. Since it is a given that exactly one such candidate pile exists, the game stops whenever a single candidate pile remains.

[1]We hold to the usual assumption that a MAC output approximates a uniform random variable. The reader may want to check the so-called *random oracle model* introduced by Bellare and Rogaway [1, 8].

The analogy is straightforward: the act of observing the j-th card from the query pile corresponds to the act of obtaining a new message/authentication pair (ρ_j, a_j), and the observation of the j-th card from the i-th candidate pile corresponds to the calculation of $f(\rho_j, \kappa_i)$ performed by the attacker.

Theorem 2.1. *The expected number of MAC queries in the EKS attack against a MAC is* $\Theta(k/a)$.

Proof. Back to our card game analogy, let X be a random variable for the number of rounds until only one candidate remains. The probability that X is equal to j is the probability that j rounds are sufficient to eliminate $|K|-1$ candidates, but $j-1$ rounds are not. Now, for $d \geq 0$, the probability that d rounds are sufficient to eliminate $|K|-1$ candidates is $(1-(1/|A|)^d)^{|K|-1}$, hence

$$\mathbf{Pr}[X=j] = \left(1-\frac{1}{|A|^j}\right)^{|K|-1} - \left(1-\frac{1}{|A|^{j-1}}\right)^{|K|-1},$$

and we can obtain the expectation of X as follows:

$$\begin{aligned}
\mathbf{E}[X] &= \sum_{j=1}^{\infty} j\ \mathbf{Pr}[X=j] = \\
&= \sum_{j=1}^{\infty} j \left[\left(1-\frac{1}{|A|^j}\right)^{|K|-1} - \left(1-\frac{1}{|A|^{j-1}}\right)^{|K|-1}\right] = \\
&= \sum_{j=1}^{\infty} j \left(1-\frac{1}{|A|^j}\right)^{|K|-1} - \sum_{j=1}^{\infty} j \left(1-\frac{1}{|A|^{j-1}}\right)^{|K|-1} = \\
&= \underbrace{\lim_{J\to\infty}\left[\sum_{j=1}^{J} j \left(1-\frac{1}{|A|^j}\right)^{|K|-1}\right]}_{L_1} - \underbrace{\lim_{J\to\infty}\left[\sum_{j=1}^{J} j \left(1-\frac{1}{|A|^{j-1}}\right)^{|K|-1}\right]}_{L_2}.
\end{aligned}$$

We will now handle separately the two limits above, which we will refer to as L_1 and L_2, respectively. We start by isolating the last term in the summation of L_1, which then becomes

$$\begin{aligned}
L_1 &= \lim_{J\to\infty}\left[\sum_{j=1}^{J-1} j \left(1-\frac{1}{|A|^j}\right)^{|K|-1} + J\left(1-\frac{1}{|A|^J}\right)^{|K|-1}\right] = \\
&= \lim_{J\to\infty}\left[\sum_{j=1}^{J-1} j \left(1-\frac{1}{|A|^j}\right)^{|K|-1} + J\right],
\end{aligned}$$

since $\lim_{J\to\infty}\left(1-\frac{1}{|A|^J}\right)^{|K|-1} = 1$.

We proceed with a slight modification in the index and bounds of the summation in L_2, yielding

$$L_2 = \lim_{J\to\infty}\left[\sum_{j=0}^{J-1}(j+1)\left(1-\frac{1}{|A|^j}\right)^{|K|-1}\right].$$

We now isolate the first term in the summation above, allowing us to rewrite L_2 as follows:

$$\begin{aligned} L_2 &= \lim_{J\to\infty}\left[(0+1)\left(1-\frac{1}{|A|^0}\right)^{|K|-1}+\sum_{j=1}^{J-1}(j+1)\left(1-\frac{1}{|A|^j}\right)^{|K|-1}\right] = \\ &= \lim_{J\to\infty}\sum_{j=1}^{J-1}(j+1)\left(1-\frac{1}{|A|^j}\right)^{|K|-1} = \\ &= \lim_{J\to\infty}\left[\sum_{j=1}^{J-1} j\left(1-\frac{1}{|A|^j}\right)^{|K|-1}+\sum_{j=1}^{J-1}\left(1-\frac{1}{|A|^j}\right)^{|K|-1}\right]. \end{aligned}$$

We can now resume the calculation of $\mathbf{E}[X]$, and we obtain

$$\begin{aligned} \mathbf{E}[X] &= L_1 - L_2 = \\ &= \lim_{J\to\infty}\left[\sum_{j=1}^{J-1} j\left(1-\frac{1}{|A|^j}\right)^{|K|-1}+J\right]- \\ &\quad \lim_{J\to\infty}\left[\sum_{j=1}^{J-1} j\left(1-\frac{1}{|A|^j}\right)^{|K|-1}+\sum_{j=1}^{J-1}\left(1-\frac{1}{|A|^j}\right)^{|K|-1}\right] = \\ &= \lim_{J\to\infty}\left[J-\sum_{j=1}^{J-1}\left(1-\frac{1}{|A|^j}\right)^{|K|-1}\right], \end{aligned}$$

by canceling out the first term of L_1 with the first term of L_2. Now, after substituting J with $1+\sum_{j=1}^{J-1}1$ and a few easy manipulations, we obtain

$$\begin{aligned} \mathbf{E}[X] &= \lim_{J\to\infty}\left[1+\sum_{j=1}^{J-1}1-\sum_{j=1}^{J-1}\left(1-\frac{1}{|A|^j}\right)^{|K|-1}\right] = \\ &= 1+\lim_{J\to\infty}\sum_{j=1}^{J-1}\left[1-\left(1-\frac{1}{|A|^j}\right)^{|K|-1}\right] = \\ &= 1+\sum_{j=1}^{\infty}\left[1-\left(1-\frac{1}{|A|^j}\right)^{|K|-1}\right]. \end{aligned}$$

The trick now is to split the infinite summation above into two partial sums S_1 and S_2, where

$$\begin{aligned} S_1 &= \sum_{j=1}^{L} \left[1 - \left(1 - \frac{1}{|A|^j} \right)^{|K|-1} \right], \\ S_2 &= \sum_{j=L+1}^{\infty} \left[1 - \left(1 - \frac{1}{|A|^j} \right)^{|K|-1} \right], \end{aligned}$$

so that $\mathbf{E}[X] = 1 + S_1 + S_2$.

By appropriately choosing $L = \lceil \log_{|A|} |K| \rceil$, we will be able to show that

(i) $0.28k/a \leq S_1 \leq 1 + k/a$, and
(ii) $0 < S_2 \leq 1$,

therefore proving our desired expectation satisfies

$$1 + 0.28k/a \leq \mathbf{E}[X] \leq 3 + k/a,$$

and the theorem ensues.

(i) The upper bound is straightforward. Since all terms in S_1 are less than or equal to 1, S_1 must be less than or equal to the number L of terms, hence $S_1 \leq \lceil \log_{|A|} |K| \rceil \leq 1 + k/a$. For the lower bound, we argue that the sequence of terms in S_1 is monotonically decreasing, so S_1 must be greater than or equal to the number L of terms multiplied by the last term, that is,

$$\begin{aligned} S_1 &\geq L \cdot \left[1 - \left(1 - \frac{1}{|A|^L} \right)^{|K|-1} \right] \\ &\geq \log_{|A|} |K| \cdot \left[1 - \left(1 - \frac{1}{|K|+1} \right)^{|K|-1} \right]. \end{aligned}$$

Now, since $|K| \geq 2$, the well-known inequality $(1 - 1/x)^y \leq e^{-y/x}$ for $x, y \geq 1$ allows us to write the suitable bound

$$\left(1 - \frac{1}{|K|+1} \right)^{|K|-1} \leq e^{\frac{1-|K|}{|K|+1}},$$

and we obtain

$$\begin{aligned} S_1 &\geq \log_{|A|} |K| \cdot [1 - e^{\frac{1-|K|}{|K|+1}}] \\ &\geq \log_{|A|} |K| \cdot [1 - e^{-1/3}] \\ &\geq 0.28k/a. \end{aligned}$$

(ii) Since all terms in S_2 are positive, S_2 is positive. For the upper bound, we recall that

$$x^n - y^n = (x-y)\sum_{m=1}^{n} x^{n-m}y^{m-1}.$$

By making $x = 1, y = 1 - \frac{1}{|A|^j}$ and $n = |K| - 1$ in the expression above, the general term in S_2 can be bounded as follows:

$$\begin{aligned} 1 - \left(1 - \frac{1}{|A|^j}\right)^{|K|-1} &= 1^{|K|-1} - \left(1 - \frac{1}{|A|^j}\right)^{|K|-1} = \\ &= \frac{1}{|A|^j}\sum_{m=1}^{|K|-1}\left(1 - \frac{1}{|A|^j}\right)^{m-1} \leq \\ &\leq \frac{|K|-1}{|A|^j}, \end{aligned}$$

where the inequality comes from the fact that the terms in the summation are all less than 1. Thus, we can write

$$S_2 \leq \sum_{j=L+1}^{\infty} \frac{|K|-1}{|A|^j}.$$

Now, since the sum of the terms of an infinite geometric progression $x_1 r, x_1 r^2, x_1 r^3, \ldots$ for $0 < r < 1$ converges to $x_1 \cdot \frac{1}{1-r}$, and because $|A|^j > |K| - 1$ for all $j > L$, we obtain

$$\begin{aligned} S_2 &\leq \sum_{j=L+1}^{\infty} \frac{|K|-1}{|A|^j} = \\ &= \frac{|K|-1}{|A|^{L+1}} \cdot \frac{|A|}{|A|-1} = \\ &= \frac{|K|-1}{|A|^{L} \cdot |A|} \cdot \frac{|A|}{|A|-1} \leq \\ &\leq \frac{|K|-1}{|K| \cdot |A|} \cdot \frac{|A|}{|A|-1} = \\ &= \frac{|K|-1}{K} \cdot \frac{1}{|A|-1} \leq \\ &\leq 1, \end{aligned}$$

where the penultimate inequality comes from the fact that L had been chosen as $\lceil \log_{|A|} |K| \rceil$. This concludes the proof. □

Theorem 2.2. *The expected number of MAC computations in the EKS attack against a MAC is* $\Theta(2^k)$.

Proof. Let Y be a random variable for the total number of MAC computations during the EKS attack. We show that $\mathbf{E}[Y]$ is $(|K|-1)\frac{|A|}{|A|-1}$. Using the card game analogy, the probability that a pile is eliminated by the time a card from a candidate pile is observed is the probability that that card does not match the corresponding card in the query pile, namely $p = (|A|-1)/|A|$. Since the outcomes of all card observations are independent from one another, the number Y_q of observations between the $(q-1)$-th elimination of a candidate and the q-th elimination of a candidate is a geometric random variable, whose expectation is $1/p = |A|/(|A|-1)$. Clearly, $Y = \sum_{q=1}^{|K|-1} Y_q$. By the linearity of expectations,

$$\mathbf{E}[Y] = \mathbf{E}\left[\sum_{q=1}^{|K|-1} Y_q\right] = \sum_{q=1}^{|K|-1} \mathbf{E}[Y_q] = (|K|-1)\frac{|A|}{|A|-1}$$

and the theorem follows. □

As a corollary of the previous results, the EKS attack takes expected time $\Theta((k/a)t_q + 2^k t_c)$, where t_q is the MAC *query time* (the time to obtain each message/authentication pair) and t_c is the MAC *computation time* (the time to output an authentication for a given message/key pair).

3. Application to software integrity verification

A fundamental issue in critical software dependent systems is the verification of whether the software that runs in some electronic device—such as mobile phones, credit card terminals, pay-TV interfaces, smart meters etc.—corresponds to a previously validated base version, not tampered with by some ill-intentioned agent. The simplest method of verifying the integrity of a software is to access it directly and compare it against the base version. However, when this verification is done remotely over a transmission channel with limited bandwidth, it is often infeasible to have the whole code be sent through. Moreover, this approach could compromise the developer's intellectual property in many cases. Modern software integrity verification methods (SIVM) sort out both problems by using the concept of reflection, where a cryptographic digest of the most critical parts of the software is obtained in a MAC-like fashion. This allows for the comparison of the (smaller, not prone to copyright issues) cryptographic digest, instead of the code itself. Recently, several works [4–7, 9] have dealt with such reflection-based SIVM. However, a rigorous discussion on their robustness was still missing.

The results for the EKS attack against a MAC can be naturally leveraged to the scope of reflection-based SIVM. To establish the correspondence of both scenarios, we note that the software digest produced by the latter is similar to the authentication produced by a MAC, in the sense that it is also the output of a non-injective function which takes two parameters, one that is public (in this case an integer usually called *seed*) and another which is secret (in this case some subset of the software code itself, henceforth referred to as *code chunk*). During

the verification process, a seed is provided, and the function computes the intended digest. An attacker aims to replace parts of the software with malicious code while still being able to produce the expected digest for whatever seed is provided, thus deceiving future verifications. To do so, it is necessary to figure out and retain a copy of the original code chunk which is processed by the digest-generating function. For this purpose, an attacker may carry on a brute force attack in the likes of the EKS by collecting a sufficient number of seed/digest pairs and filtering out non-matching candidates from an initially big set of code chunks.

By reproducing the same reasoning as in Section 2, we observe that the expected number of seed/digest pairs before the attack is successful is here again $\Theta(k/a)$, where k is the bitlength of the code chunk considered by the digest-generating function and a is the bitlength of the digest. It thus becomes clear that such kind of attack can be precluded by setting a reasonable ratio between the size of the code chunk and the size of the digest. One must notice, however, that assigning too small a value for a has the drawback of reducing the injectivity of the digest function, which increases the probability that a malicious software returns the expected digest out of pure chance. A complementary approach is to use the bounds given by Theorem 2.1 to conceive a limit for the maximum number of digests that can be queried within a certain period of time.

4. Final considerations

Estimating the threat posed by brute force attacks is vital when assessing the robustness of MAC algorithms. The emergence of new powerful computing paradigms—such as grid computing—may lead to situations where an attacker has huge processing power[2]. To make sure no adversary with "close-to-infinite" computational power can cause any damage, one must enforce a safe *lower* bound to the time complexity of the attacks. Quite surprisingly, good lower bounds on the complexity of the exhaustive key search attack were not known, so the software security personnel could do no better than underestimating it for the sake of security. A tight bound on that complexity certainly means economy of resources.

References

[1] M Bellare and P Rogaway. *Random oracles are practical: a paradigm for designing efficient protocols*. Proc. 1st ACM conference on Computer and communications security (1993) 62–73.

[2] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*, CRC Press, USA, 1996.

[3] B. Preneel. *Hash functions and MAC algorithms based on block cyphers.* In Cryptography and Coding, 6th IMA International Conference, Lecture Notes in Computer Science 1355:270–282, 1997.

[2]The Great Internet Mersenne Prime Project (`http://www.mersenne.org/primenet`) recently accomplished 50 TFLOP (trillions of operations per second) using a network of over three hundred thousand computers.

[4] D. Spinellis. *Reflection as a Mechanism for Software Integrity Verification.* ACM Transactions on Information and System Security, 3(1): 51–62, 2000.

[5] A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla. *Swatt: Software-based attestation for embedded devices.* In 2004 IEEE Symposium on Security and Privacy, page 272, Los Alamitos, CA, 2004.

[6] A. Seshadri, M. Luk, E. Shi, A. Perrig, L. van Doorn, and P. Khosla. *Pioneer: verifying code integrity and enforcing untampered code execution on legacy systems.* SIGOPS Oper. Syst. Rev., 39(5):1–16, 2005.

[7] A. Seshadri, M. Luk, A. Perrig, L. van Doorn, and P. Khosla. *Externally verifiable code execution.* Commun. ACM, 49(9): 45–49, 2006.

[8] D. R. Stinson. 2006. *Some Observations on the Theory of Cryptographic Hash Functions.* Des. Codes Cryptography, 38 (2006) 259–277.

[9] Y. Yang, X. Wang, S. Zhu, and G. Cao. *Distributed software-based attestation for node compromise detection in sensor networks.* In Proceedings of the IEEE Symposium on Reliable Distributed Systems, pp. 219–228, 2007.