

# Context aware security approach for IoT environments

Giovani Ferreira and Rafael  
Universal Internet of Things (UIoT)  
Technology Faculty (FT)  
University of Brasilia (UFSC)  
70910-900 Brasilia, DF Brazil  
giovani.silva@redes.unb.br, caio.silva@redes.unb.br

**Abstract**—adhasjkdhaksjd [1]

**Index Terms**—security, quality of context, Internet of things

## I. INTRODUCTION

Collision search is an important tool in cryptanalysis. A broad range of cryptanalytic problems such as computing discrete logarithms, finding hash function collisions, and meet-in-the-middle attacks can be reduced to the problem of finding two distinct inputs,  $a$  and  $b$ , to a function  $f$  such that  $f(a) = f(b)$  [2].

## II. RELATED CONCEPTS

### A. Message Integrity

Message integrity is normally maintained via the protocol called MAC” or Message Authentication Code. To briefly explain its mechanism,

In the digital world, the techniques for transmission and integrity assurance of messages are in constant evolution, aiming more efficient and secure protocols. A protocol’s security can be threatened by innumerable different ways, some possibly unknown yet, and requires a reasonable time of study, tests and attacks to be considered for use in real applications.

O protocolo de autenticacao HMAC pode ser entendido em detalhes em [3], mas como uma breve introducao, consideremos o seguinte exemplo: Alice deseja se comunicar com Bob, a integridade deve ser mantida, ou seja, the message should not be tampered with or changed to contain false or modified information:

- 1) Alice gera uma signing tag  $S(k, m)$ , for  $m$  = message and  $k$  = secret key between Alice and Bob.
- 2) Bob ao receber a mensagem, runs a verification algorithm, defined by  $V(k, m, tag) = V(k, m, S(k, m))$

- 3) Using the same key, the algorithm will return ‘yes’ which shows that message integrity has been maintained ...

In that way, Bob is able to identify that the message sent by Alice was not compromised and its content is exactly what Alice had written.

### B. Hash Collision

Hash functions are designed to take a message of arbitrary bitlength and map it to a fixed size output called a hash result. Let  $H: M \rightarrow R$  be such a hash function. Typically, hash functions are constructed from a function  $h: BR \rightarrow R$  which takes a fixed size block of message bits together with an intermediate hash result and produces a new intermediate hash result. A given message  $m \in M$  is typically padded to a multiple of the block size and split into blocks  $m_1, \dots, m_l \in BR$ . The padding often includes a field which indicates the number of bits in the original message. Beginning with some constant  $r_0 \in R$ , the sequence  $r_i = h(m_i, r_{i-1})$  is computed for  $i = 1, \dots, l$ , and  $r_l$  is the hash result for message  $m$  [2].

### C. Birthday Attack

The birthday paradox is the counter-intuitive principle that for groups of as few as 23 persons there is already a chance of about one half of finding two persons with the same birthday (assuming all birthdays are equally likely and disregarding leap years). Compared to finding someone in this group with your birthday where you have 23 independent chances and thus a success probability of  $23/365 \approx 0.06$ , this principle is based on the fact that there are  $(23 * 22)/2 = 253$  distinct pairs of persons. This leads to a success probability of about 0.5 (note that this does not equal  $253/365 \approx 0.7$  since these pairs are not independently distributed) [4].

The following is the general algorithm for the Birthday Attack and in the next section I will discuss

the Birthday Paradox, which is a problem that gave birth to the Birthday Attack algorithm. 1 - Let  $H:M \rightarrow \{0,1\}^n$  be a hash function. From this we know that the size of the tag space is  $2^n$  bits and that  $|M| \geq 2^n/2$ . 2 - We choose  $2^{n/2}$  random messages in  $M$ , i.e.  $m_1, m_2, \dots, m_{2^{n/2}}$  pertenentes a  $M$ . 3 - For  $i = 1, 2, \dots, 2^{n/2}$  compute  $t_i = H(m_i)$ , where  $t_i$  is the hash value in the tag space. 4 - We then search for any collisions, i.e.  $t_i = t_j$  for  $i, j$  pertenentes  $1, 2, \dots, 2^{n/2}$ . If this is not found we go back to step 1 and repeat with different message samples.

#### D. Se pa - Distributed System

A distributed system is a collection of independent computers that appears to its users as a single coherent system [1].

### III. EXPERIMENTS AND EVALUATION

Foram aplicados tecnicas de paralelismo (openmp) e distribuicao (mpi) visando uma melhoria na performance da busca por colisao. A funcao hash usada nos testes foi a MD5.

### IV. CONCLUSIONS AND FUTURE WORK

#### REFERENCES

- [1] A. S. Tanenbaum and M. Van Steen, *Distributed systems: principles and paradigms*. Prentice hall Englewood Cliffs, 2002, vol. 2.
- [2] P. C. Van Oorschot and M. J. Wiener, "Parallel collision search with cryptanalytic applications," *Journal of cryptology*, vol. 12, no. 1, pp. 1–28, 1999.
- [3] H. Krawczyk, R. Canetti, and M. Bellare, "Hmac: Keyed-hashing for message authentication," 1997.
- [4] M. M. J. Stevens *et al.*, *Attacks on hash functions and applications*. Mathematical Institute, Faculty of Science, Leiden University, 2012.