Context aware security approach for IoT environments

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

***Abstract*—adhaskjdhaksjd [1]**

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

I. Introduction

Before we begin discussing the Birthday Attack, we have to be aware that we have a goal of maintaining message integrity in computer security, i.e. the message that a person A sends to another person B should not be tampered with or changed to contain false or modified information. Message integrity is normally maintained via the protocol called MAC" or Message Authentication Code. To briefly explain its mechanism, let us use an example of a communication between two people, in which there is a sender and a receiver. The sender and the receiver both have a shared key, k, that they can use for the MAC signing algorithm. Now, the sender generates a tag 2 that goes along with the message that he/she has sent. When the receiver receives the message and the tag, the receiver runs the MAC verification tag and this verification algorithm outputs yes or no depending on the validity of the message. To describe this whole procedure step by step using some cryptographic notations: 1. Sender generates a signing tag S(k, m), where k is the shared key, and m is the message. 2. The receiver runs a verification algorithm, defined by V (k, m, tag) = V (k, m, S(k, m)). 3. If the sender signs with a particular key, and the receiver verifies with the same key, then the result of V (k, m, S(k, m)) = yes which shows that message integrity has been maintained and the receiver is able to view the correct information that the sender transmitted.

II. Related Concepts

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

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].

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 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 0.7 since these pairs are not independently distributed) [3].

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 R be such a hash function. Typically, hash functions are constructed from a function h: BR 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 M is typically padded to a multiple of the block size and split into blocks m 1 , ... m l B. The padding often includes a field which indicates the number of bits in the original message. Beginning with some constant r 0 R, the sequence r i = h(m i , r i1) is computed for i = 1, ... l, and r l is the hash result for message m [2].

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 -¿ 0,1ĥ be a hash function. From this we know that

the size of the tag space is 2n̂ bits and that —M— ¿¿ 2n̂ 2 - We choose 2(̂n/2) random messages in M, i.e. m1, m2, ... , m2(n/2) pertencentes a M. 3 - For i = 1,2, ... , 2(n/2) compute ti = H(mi), where ti is the hash value in the tag space. 4 - We then search for any collisions, i.e. ti = tj for i, j pertencentes 1, 2, ... , 2(̂n/2). It this is not found we go back to step 1 and repeat with different message samples.

III. Experiments and Evaluation

Foram aplicados tecnicas de paralelismo (openmp) e distribuicao (mpi) visando uma mlehora 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] M. M. J. Stevens *et al.*, *Attacks on hash functions and applications*. Mathematical Institute, Faculty of Science, Leiden University, 2012.