

# Instructions for Authors of SBC Conferences Papers and Abstracts

Giovani Ferreira<sup>1</sup>, Rafael Marconi<sup>1</sup>

<sup>1</sup>CEUB - Centro Universitário de Brasília  
Caixa Postal 4488 – 70.904-970 – Brasília – DF – Brazil

**Abstract.** *adhasjkdhaksjd [Tanenbaum and Van Steen 2002]*

## 1. Introduction

Apesar de técnicas para evitar o ataque do aniversário já terem sido criadas [Aiello and Venkatesan 1996].

Busca por colisão é uma ferramenta importante em criptoanálise. Uma gama grande de problemas criptoanalíticos tais como computar logaritmos discretos, achar colisões de função hash e o ataque por encontro a médio caminho, meet-in-the-middle, pode ser reduzido ao problema de achar duas entradas distintas,  $a$  e  $b$ , para uma função  $f$  de forma que  $f(a) = f(b)$  [Van Oorschot and Wiener 1999].

As formas mais comuns de uso de função hash na criptografia são com assinaturas digitais e para integridade dos dados. Na assinatura digital, uma mensagem longa geralmente passa pelo processo de hash (usando uma função hash publicamente disponível) e somente o resultado do hash é assinado. A parte destinatária da mensagem aplica o processo de hash na mensagem, verifica que a assinatura recebida é correta para esse valor de hash. Isso salva tanto tempo quanto espaço comparado com assinar a mensagem diretamente, o que iria tipicamente envolver dividir a mensagem em blocos de tamanho apropriado e assinar cada bloco individualmente. Note aqui que a inabilidade de achar duas mensagens com o mesmo valor de hash é uma necessidade de segurança, visto que de outra maneira, a assinatura em um valor hash de uma mensagem seria o mesmo que o de outra, permitindo que o assinante assine uma mensagem e em um outro ponto no tempo, reivindicar que ele assinou outra [Menezes et al. 1996]. Funções hash podem ser usadas para manter a integridade da seguinte maneira. O valor do hash correspondente a uma entrada particular é computado em algum ponto do tempo. A integridade desse valor hash é protegido de alguma maneira. Em um ponto subsequente no tempo, para verificar que o dado de entrada não foi alterado, o valor do hash é computado novamente usando a entrada disponível e comparado por igualdade com o valor do hash original. Aplicações específicas incluem proteção contra vírus e distribuição de software [Menezes et al. 1996].

## 2. Related Concepts

### 2.1. Hash Functions

Uma função hash é a função computacionalmente eficiente que mapeia uma sequência binária de tamanho arbitrário para uma sequência binária de tamanho fixo, chamada de hash-value, hash ou digest [Menezes et al. 1996].

## 2.2. Hash Collision

**Collision Resistance** - It is computationally infeasible to find any two distinct input  $x, x'$  which hash to the same output, i.e., such that  $h(x) = h(x')$ . (Note that here there is free choice of both inputs.) [Menezes et al. 1996]

A hash function  $h$  is called *collision free*, if it maps messages of any length to strings of some fixed length, but such that finding  $x, y$  with  $h(x) = h(y)$  is a hard problem. Note that we are concentrating here on publicly computable hash functions, i.e. functions that are not controlled by a secret key [Damgård 1989].

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  $h : B \times R \rightarrow R$  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  $m \in \mathbb{M}$  is typically padded to a multiple of the block size and split into blocks  $m_1, m_2, \dots, m_l \in B$ . 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  $i = 1, 2, \dots, l$ , and  $r_l$  is the hash result for message  $m$  [Van Oorschot and Wiener 1999].

## 2.3. Birthday Paradox

O paradoxo do aniversário é o princípio contraintuitivo que para grupos de somente 23 pessoas, existe a chance de aproximadamente 50% de encontrar duas pessoas com o mesmo aniversário (Assumindo que todos os aniversários são igualmente prováveis e desconsiderado anos bissexto). Comparado a probabilidade de encontrar alguém nesse grupo com o mesmo aniversário que o seu, aonde se têm 23 chances independentes e portanto uma probabilidade de sucesso de  $\frac{23}{365} \approx 0.06$ , esse princípio é baseado no fato de que existe  $\frac{23 \cdot 22}{2} = 253$  pares distintos de pessoas. Isso leva a uma probabilidade de sucesso por volta de 0.5 (note que isso não é igual a  $\frac{253}{365} \approx 0.7$  já que esses pares não são independentemente distribuídos) [Stevens et al. 2012].

## 2.4. Birthday Attack

O algoritmo genérico para o ataque do aniversario é composto pelas seguintes etapas:

1. Dada a função hash  $H : M \rightarrow \{0, 1\}^n$  e sabendo que o tamanho do conjunto das *tags* é  $\approx 2^n$  bits e que  $|M| \gg 2^n$
2. Escolhe-se  $2^{\frac{n}{2}}$  mensagens aleatórias em  $\mathbb{M}$ , de forma que  $m_1, m_2, \dots, m_{2^{\frac{n}{2}}} \in \mathbb{M}$ .
3. Para  $i = 1, 2, \dots, 2^{\frac{n}{2}}$  computa-se  $t_i = H(m_i)$ , aonde  $t_i$  é o hash no conjunto das *tags*
4. Busca-se por qualquer colisão, ou seja,  $t_i = t_j$  para  $i, j \in 1, 2, \dots, 2^{\frac{n}{2}}$ . Caso não seja encontrada, volta-se a etapa 1 e repete-se com uma amostra diferente de mensagens.

## 2.5. Se pa - Distributed System

A distributed system is a collection of independent computers that appears to its users as a single coherent system [Tanenbaum and Van Steen 2002].

### 3. Experiments and Evaluation

Foram aplicadas técnicas de paralelismo (openmp) e distribuição (mpi) visando uma melhoria na performance da busca por colisões. A função hash usada nos testes foi a MD5.

### 4. Conclusions and Future work

#### Referências

- Aiello, W. and Venkatesan, R. (1996). Foiling birthday attacks in length-doubling transformations. In *Advances in Cryptology—EUROCRYPT’96*, pages 307–320. Springer.
- Damgård, I. B. (1989). A design principle for hash functions. In *Advances in Cryptology—CRYPTO’89 Proceedings*, pages 416–427. Springer.
- Menezes, A. J., Van Oorschot, P. C., and Vanstone, S. A. (1996). *Handbook of applied cryptography*. CRC press.
- Stevens, M. M. J. et al. (2012). *Attacks on hash functions and applications*. Mathematical Institute, Faculty of Science, Leiden University.
- Tanenbaum, A. S. and Van Steen, M. (2002). *Distributed systems: principles and paradigms*, volume 2. Prentice hall Englewood Cliffs.
- Van Oorschot, P. C. and Wiener, M. J. (1999). Parallel collision search with cryptanalytic applications. *Journal of cryptology*, 12(1):1–28.