Implementação do Algoritmo de Hash MD5 em Python

Felipe Cartaxo de Freitas Sheila Lima Lee

July 2024

Abstract. This technical report describes the implementation of the MD5 hash algorithm in Python, as described in RFC 1321. The implementation was carried out without the use of auxiliary libraries, adhering to the principles of the original algorithm. The developed function receives a string as input and returns the corresponding MD5 hash. The report details each step of the process, from parameter initialization to the generation of the final hash, and compares the results obtained with Python's hashlib function for correctness verification.

Resumo. Este relatório técnico descreve a implementação do algoritmo de hash MD5 em Python, conforme descrito na RFC 1321. A implementação foi realizada sem o uso de bibliotecas auxiliares, respeitando os princípios do algoritmo original. A função desenvolvida recebe uma string como entrada e retorna o hash MD5 correspondente. O relatório detalha cada etapa do processo, desde a inicialização dos parâmetros até a geração do hash final, e compara os resultados obtidos com a função hashlib do Python para verificação da correção.

1 Introdução

O algoritmo de hash MD5 (Message-Digest Algorithm 5) é amplamente utilizado em aplicações de segurança e integridade de dados. Este trabalho apresenta a implementação deste algoritmo em Python, seguindo as especificações da RFC 1321. A função desenvolvida foi testada e validada através da comparação com a função *hashlib.md5* do Python.

2 Metodologia

A implementação foi realizada conforme os passos definidos na RFC 1321, com as seguintes etapas principais:

2.1 Inicialização dos Parâmetros

Os parâmetros iniciais (A, B, C, D) são definidos conforme especificado:

```
a0 = 0x67452301
b0 = 0xefcdab89
c0 = 0x98badcfe
d0 = 0x10325476
```

2.2 Preprocessamento

A entrada é convertida em bytes e preenchida de acordo com o padrão MD5:

```
msg = bytearray(string, 'utf-8')
original_len_in_bits = (8 * len(msg)) & 0xffffffffffffffff
msg.append(0x80)
while len(msg) % 64 != 56:
    msg.append(0)
msg += original_len_in_bits.to_bytes(8, byteorder='little')
```

2.3 Processamento em Blocos

O processamento é feito em blocos de 512 bits, com operações de transformação:

```
for chunk_ofst in range(0, len(msg), 64):
    a, b, c, d = a0, b0, c0, d0
    chunk = msg[chunk_ofst:chunk_ofst + 64]
    # Transforma  o de cada bloco
    for i in range(64):
        f, g = 0, 0
        if 0 <= i <= 15:
            f = (b & c) | (~b & d)
            g = i
        # Outras opera  es de transforma  o
        f = (f + a + K[i] + int.from_bytes(chunk[4 * g:4 * g + 4],
           byteorder='little')) & 0xffffffff
        a, d, c, b = d, (b + leftrotate(f, s[i])) & 0xffffffff, b, c
    a0 = (a0 + a) & 0xffffffff
    b0 = (b0 + b) & 0xffffffff
    c0 = (c0 + c) & 0xffffffff
    d0 = (d0 + d) & 0xffffffff
```

2.4 Geração do Hash

Os valores finais são concatenados para formar o hash MD5:

```
hashString = ''.join([x.to_bytes(4, byteorder='little').hex() for x in [
    a0, b0, c0, d0]])
```

3 Resultados e Discussão

A função *hashMD5Aluno* foi comparada com a função *hashlib.md5* do Python. Ambos os métodos produziram resultados idênticos para diversas entradas de teste, confirmando a correção da implementação.

4 Conclusão

A implementação do algoritmo MD5 em Python foi bem-sucedida e validada com testes comparativos. Este trabalho demonstra a compreensão dos princípios do algoritmo de hash MD5 e sua aplicação prática.

5 Referências

- Rivest, R. L. (1992). The MD5 Message-Digest Algorithm. RFC 1321.
- Sociedade Brasileira de Computação. Instruções para Autores de Contribuições para o SIBGRAPI.