

Implementação do algoritmo hash SHA-1 em processador digital de sinais (DSP)

Douglas B. S. Figueiredo, Walter Antônio Gontijo

Curso de Engenharia de Computação – Universidade do Vale do Itajaí (UNIVALI)
Rodovia SC 407 – Km 04 – Bairro Sertão do Maruim CEP 88122-000 – São José – SC

Douglas@ecilinformatica.com.br, wgontijo@univali.br

Abstract. *Ahead of the increasing use of digital documents, arises the needs to use a technology that becomes possible the authentication of these documents, aiming confirmation of their authors and preventing fraud. This paper presents the implementation of SHA-1 algorithm (Secure Hash Algorithm) defined by National Institute of Standards and Technology (NIST). This algorithm was chosen for being considered by NIST, a standard of hash algorithm to implement digital signature systems. To perform this implementation was used a digital signal processor (DSP), because of its high capacity to perform the operations contained in the algorithm. The performance of the implemented algorithm is compatible with commercial implementations.*

Resumo. *Diante do uso crescente de documentos digitais, surge a necessidade de utilizar uma tecnologia que torne possível a autenticação destes documentos, visando à confirmação de seus autores e também impedir fraudes. Este trabalho tem como objetivo a implementação do algoritmo SHA-1 (Secure Hash Algorithm) definido pela National Institute of Standards and Technology (NIST). Este foi escolhido por ser considerado pela NIST o algoritmo hash padrão na implementação de um sistemas de assinatura digital. Para realizar esta implementação será utilizado um processador digital de sinais (DSP), devido a sua alta capacidade de realizar as operações contidas no algoritmo. O desempenho do algoritmo implementado é compatível com implementações comerciais.*

1. Introdução

Os algoritmos *hash* são comumente utilizados em casos que o usuário não deseja tornar os dados ineleáveis, porém, quer ter a certeza de que o receptor saberá se ocorreu alteração nos dados. Por não tornar os dados ineleáveis, os algoritmos *hash* são muito mais rápidos que os de criptografia assimétrica, já que, apenas geram um resumo do documento [Burnett 2002]. A utilização de algoritmos *hash* e de criptografia assimétrica torna possível realizar a implementação de um sistema de assinatura digital. Este sistema tem por objetivo substituir as assinaturas tradicionais por uma assinatura eletrônica.

Este sistema de assinatura digital pode ser observado no fluxograma para geração da assinatura digital mostrado na Figura 1 e apresenta duas etapas: “SHA-1” e “Operação de assinatura do DSA¹”. Estas etapas processam a mensagem original, obtendo sua respectiva assinatura. Na primeira etapa, a função “SHA-1” processa a mensagem, obtendo seu respectivo resumo. Na segunda etapa, a função “Operação de assinatura do DSA” utiliza uma chave privada para processar o resumo obtido pela função “SHA-1”, obtendo sua respectiva assinatura digital.

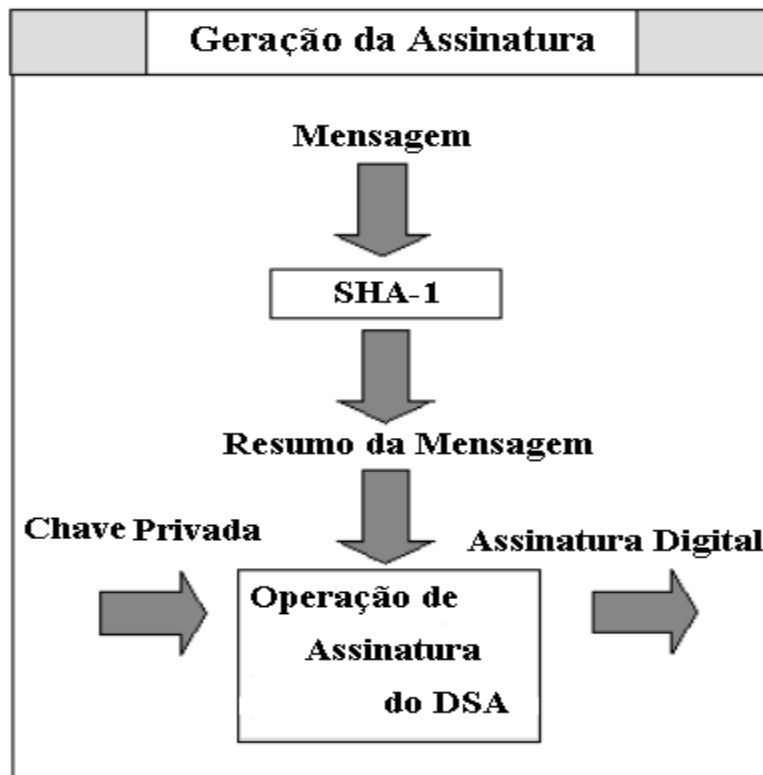


Figura 1. Sistema para geração de assinatura digital.

Fonte: Adaptado de NIST (1995).

De forma correspondente, o fluxograma de um sistema para verificação da assinatura digital é mostrado na Figura 2 e apresenta duas etapas: “SHA-1” e “Operação de verificação do DSA”. Estas etapas processam a mensagem recebida, obtendo sua respectiva assinatura, visando compará-la com a assinatura recebida. Na primeira etapa a função “SHA-1” processa a mensagem, obtendo seu respectivo resumo. Na segunda etapa a função “Operação de verificação do DSA” utiliza uma chave pública para processar o resumo obtido pela função “SHA-1”, adquirindo sua respectiva assinatura digital. Após obter a assinatura da mensagem, esta é comparada com a assinatura recebida e, somente se ambas forem iguais, a assinatura é considerada válida.

DSA (Digital Signature Algorithm) é o algoritmo padrão utilizado em operações de assinatura de documentos especificado no DSS (Digital Signature Standard) [NIST 1995].¹

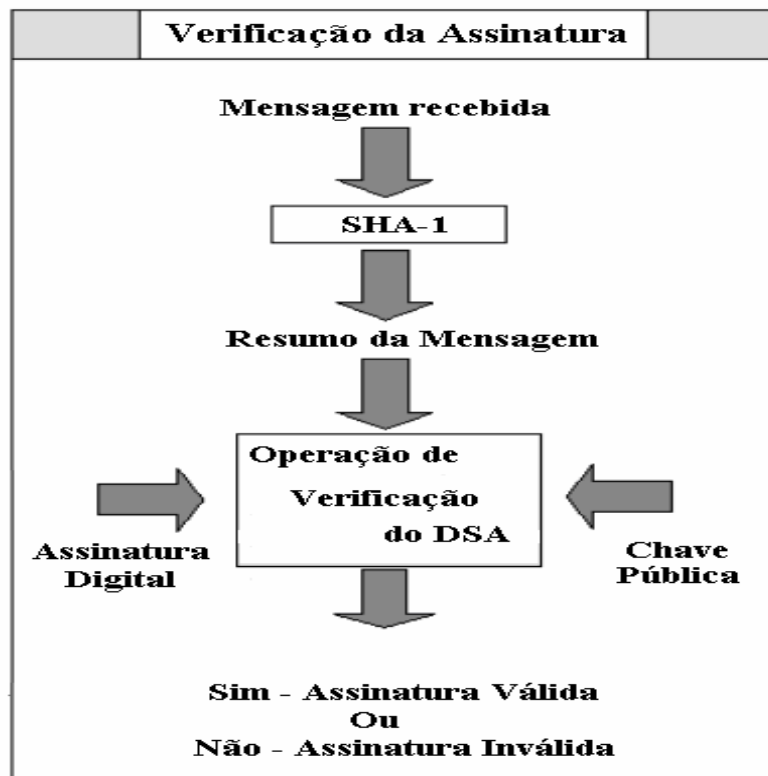


Figura 2. Sistema para verificação de assinatura digital.

Fonte: Adaptado de NIST (1995).

O resumo da mensagem gerado pelo algoritmo *hash* é comumente confundido com a assinatura digital da mensagem. A assinatura digital é obtida por meio da codificação do resumo da mensagem, gerado por um algoritmo *hash*. Com os algoritmos *hash* é possível gerar um resumo da mensagem, que consiga representar unicamente qualquer mensagem que seja submetida ao algoritmo [Lima 2005]. Estes resumos possuem tamanhos fixos que variam de algoritmo para algoritmo. Quanto maior for o tamanho do resumo da mensagem maior será a segurança do mesmo [Volpi 2001].

Os algoritmos *hash* também são conhecidos como algoritmos de sentido único. Estes têm como característica a fácil obtenção do resultado (resumo da mensagem), porém, é computacionalmente improvável reproduzir a mensagem original por meio do resumo da mensagem. Outra característica é o fato dos resultados gerados serem únicos e de difícil duplicação, o que torna praticamente impossível encontrar o mesmo resumo para mensagens diferentes [Volpi 2001].

O principal objetivo para gerar um resumo de mensagem é a obtenção da autenticidade, ou seja, a garantia de que a mensagem ou arquivo não foi alterado durante o envio. O SHA-1 (*Secure Hash Algorithm*) é um algoritmo *hash* definido pela National Institute of Standards and Technology (NIST, órgão do governo norte americano responsável pela definição de padrões) [Menezes, Oorschot, Vanstone 1996]. Este artigo apresenta a implementação do algoritmo *hash* SHA-1 em DSP e é dividido nas seguintes seções: 1. Introdução sobre o algoritmo *hash* SHA-1; 2. Justificativa da escolha do algoritmo SHA-1 e da utilização do DSP para tal implementação 3. O

Algoritmo hash, nesta seção é apresentada características do algoritmo *hash* implementado; 4. Processador Digital de Sinais, nesta seção é introduzido o DSP e algumas características da implementação do algoritmo utilizando tal tecnologia; 5. Resultados obtidos por meio de validações e testes de desempenho; 6. Comentários e Conclusões da implementação abordada no artigo.

2. Justificativa

Existem diversos tipos de algoritmos SHA porém, o algoritmo SHA-1 é considerado pela NIST o algoritmo *hash* padrão na implementação de um sistema de assinatura digital. Além disto, o algoritmo SHA-1 foi escolhido como o algoritmo hash a ser implementado neste artigo, pois este é o primeiro trabalho de implementação em DSP, na UNIVALI SJ, de um algoritmo *hash*. O trabalho será utilizado como referência para outros alunos da UNIVALI SJ na implementação de algoritmos *hash* mais robustos.

A implementação do algoritmo *hash* SHA-1 tem por finalidade prover parte de um sistema de assinatura digital para sistemas embarcados, agregando funções a sistemas que já utilizam o processador DSP. Desta maneira, para agregar a funcionalidade de autenticação a sistemas telefônicos haveria somente o custo de software, tendo em vista que o hardware (DSP) já é utilizado em sua arquitetura.

3. O Algoritmo hash

No algoritmo *hash* SHA-1, o tamanho da mensagem deve ser múltiplo de 512 bits, pois o algoritmo trabalha com blocos de 512 bits, porém esta característica é transparente ao usuário. Todos os operandos das funções matemáticas utilizados pelo algoritmo são de 32 bits. A segurança refere-se à resistência às colisões, que é de aproximadamente $2^{n/2}$, onde n é o tamanho do resumo da mensagem. Isto significa que ao utilizar um ataque de força bruta no algoritmo SHA-1, será necessário testar 2^{80} mensagens para encontrar colisões. O fluxograma de funcionamento do algoritmo SHA-1 apresenta três etapas principais: “*Message Padding*”, “*Fragmenta Padded Message*” e “*Gera resumo da mensagem*”. Estas etapas processam a mensagem original, obtendo seu respectivo resumo. Na primeira etapa a função “*Message Padding*” altera o tamanho da mensagem original para um tamanho múltiplo de 512 bits. Na segunda etapa a função “*Fragmenta Message Padded*” fragmenta um bloco de 512 bits em 16 palavras de 32 bits. Na última etapa a função “*Gera resumo*” processa estas 16 palavras de 32 bits, obtendo o resumo deste bloco.

3.1. Operações e Funções utilizadas na implementação

Operações: As operações usadas no algoritmo *hash* utilizam operandos de 32 bits e são apresentadas na Tabela 1.

Tabela 1 - Operações utilizadas pelo SHA-1.

| Operações | |
|--------------------|--|
| $X \wedge Y$ | Operação lógica “E”. |
| $X \vee Y$ | Operação lógica “OU”. |
| $X \text{ XOR } Y$ | Operação lógica “OU Exclusivo”. |
| $\sim X$ | Operação lógica “Complemento”. |
| $X + Y$ | Operação soma de inteiros. |
| $Rn(X)$ | Rotação circular, onde são rotacionados n bits de X. |

Função Message Padding: tem por objetivo tornar a mensagem de entrada múltipla de 512 bits, já que o algoritmo *hash* SHA-1 utiliza blocos com tamanho de 512 bits. Para realizar esta operação é acrescentado ao final do último bloco um bit ‘1’, seguido de n bits ‘0’, até que o bloco totalize 448 bits. Para que este bloco tenha o tamanho de 512 bits, é necessário acrescentar 64 bits referentes ao tamanho da mensagem original em bits.

Funções e constantes utilizadas na função Gera resumo: Para realizar o processamento do resumo da mensagem o algoritmo SHA-1 utiliza uma seqüência de funções lógicas e constantes. Estas funções utilizam três operandos de 32 bits (B, C e D) como entrada e têm como resultado um operando de 32 bits.

Gerando o resumo da mensagem: O resumo da mensagem é gerado utilizando o resultado da função *message padding*. Este é dividido em 16 palavras de 32 bits que são processadas para a obtenção do resumo.

4. Processador Digital de Sinais

Com a introdução dos microprocessadores em 1970 tornou-se possível a utilização do processamento digital de sinais em larga escala. Os processadores digitais de sinais são dispositivos programáveis que operam em tempo real. Estes dispõem de grande capacidade de processamento de dados e velocidade superior aos microprocessadores de propósito geral [Wolter 2007].

O processador digital de sinais escolhido para realizar a implementação em assembly é o TMS320C5410 da família C5000 da *Texas Instruments*. Este é um processador de baixo custo e é utilizado em aplicações de áudio, biometria, controle digital e telecomunicação [Ti 2008]. O DSP TMS320C5410 opera com frequência de 100MHz e realiza 100 MIPS², possui 128KB de memória Ram e 32KB de memória Rom. Este processador tem 64K palavras de 16 bits de memória de dados e sua memória de programa é configurável, podendo ser configurada com endereços internos ou externos [Ti 2000].

Uma grande vantagem deste DSP é possuir dois acumuladores auxiliares “A” e “B” de 40 bits, que permitem o processamento das operações de 32 bits existentes no algoritmo SHA-1. Outros recursos utilizados são as operações DLD (*Dual Load*) e DST (*Dual Store*), na qual DLD carrega simultaneamente dois operandos de 16 bits para o

² MIPS (Milhões de instruções por segundo).

acumulador “A” ou “B”. Já a operação DST realiza o armazenamento simultâneo de 32 bits do acumulador “A” ou “B” para duas variáveis de 16 bits alocadas sequencialmente no DSP.

Para a implementação foi utilizado o CCS (Code Composer Studio), ambiente de desenvolvimento da Texas Instruments, que oferece uma robusta plataforma de programação. Este ambiente possui ferramentas gráficas para que o sistema seja projetado com rapidez, e características que são utilizadas durante o ciclo de desenvolvimento [Ti 2008].

5. Resultados

Na Tabela 2 são apresentadas as cinco variações possíveis de entrada para o algoritmo *hash* SHA-1, deve-se frisar que esse conjunto de testes abrange todas as combinações possíveis de entrada para o algoritmo considerado. Como primeira entrada foi escolhida uma mensagem de tamanho menor que 448 bits afim de, validar o algoritmo para mensagens pequenas de apenas um bloco.

Tabela 2 - Entradas e saídas em linguagem “C” e Assembly.

| Nº | Mensagem de Entrada/Resumo de Mensagem |
|----|---|
| 1 | “doug” 0798EBEC FB7BC4DF B9D0445C CFA030EB 64E1FA1C |
| 2 | “abcdbcdecdefdefgefghfghighijhijkijklklmklmnlmnomnopnopq” 84983E44 1C3BD26E BAAE4AA1 F95129E5 E54670F1 |
| 3 | “abcdbcdecdefdefgefghfghighijhijkijklklmklmnlmnomnopnopqopqrpqrs” 79EA0C76 F0056373 FFD6A5AA D389DD90 8B0C0E94 |
| 4 | 200 vezes letra “a” E61CFFFE 0D9195A5 25FC6CF0 6CA2D771 19C24A40 |
| 5 | 250 vezes letra “a” B5D5E3E0 FCCCFB49 D704A1E1 0BC97CE9 761A14FE |

A segunda entrada possui tamanho de mensagem igual a 448 bits validando assim o algoritmo para mensagens pequenas, mas que não fosse possível processá-las utilizando apenas um bloco, sendo necessário utilizar um segundo bloco para inserir os 64 bits referentes ao tamanho da mensagem. A terceira entrada possui tamanho de mensagem de 512 bits afim de, validar o algoritmo para mensagens que necessitem processar n blocos para gerar o resumo de mensagem. A quarta entrada possui tamanho de mensagem de 1600 bits, validando o algoritmo para casos em que a mensagem necessite processar n blocos para gerar o resumo de mensagem sendo que o último bloco é menor que 448 bits, de forma similar à primeira entrada. E por fim a última entrada possui tamanho de mensagem de 2000 bits, validando o algoritmo para casos em que a mensagem necessite de n blocos para gerar o resumo de mensagem sendo que o último bloco é maior que 448 bits, caso similar à segunda entrada. Tanto o algoritmo em linguagem “C” quanto o algoritmo em linguagem Assembly obtiveram os mesmo resultados para as cinco mensagens de entrada assim como esperado, comprovando que a implementação está funcionando de maneira correta.

5.1. Recursos computacionais

Este tópico apresenta uma breve comparação do algoritmo implementado em relação a uma implementação comercial. A comparação foi baseada nos recursos computacionais utilizados para a implementação do algoritmo *hash* SHA-1 em linguagem assembly tais como o consumo de memória e a mipagem necessária para a execução do algoritmo.

5.2. Comparação com implementação comercial

A implementação comercial usada para esta comparação foi desenvolvida pela empresa Snapshield Ltd. conforme referência [Ti 2008b]. Para tal implementação foi utilizado um DSP da família TMS320C54XX. Para efeito de comparação, uma implementação comercial apresenta as características de recursos de memória e o número de instruções por segundo utilizadas. Na Tabela 3 pode-se observar o consumo computacional da implementação comercial e da desenvolvida.

Tabela 3 - Comparação com implementação de uso comercial.

| Implementação | Memória de programa | Memória Total | MIPS |
|-----------------|---------------------|---------------|------|
| Snapshield Ltd. | 1,4 kb | 1,5 kb | 3,2 |
| Desenvolvida | 1,58 kb | 1,59 kb | 4 |

Para a obtenção do consumo de memória do algoritmo desenvolvido foi utilizado o mapa de memória que é obtido por meio da compilação do programa utilizando o CCS, e os resultados obtidos foram de 1,58 kb de memória de programa e 1,59 kb de consumo total de memória. Comparando o consumo de memória da implementação comercial com a implementação desenvolvida pode-se se observar que os resultados são equivalentes.

Para realizar o cálculo da mipagem do algoritmo implementado é necessário saber quantos ciclos de *clock* foram utilizados para a execução do algoritmo por completo. Este valor é obtido utilizando a ferramenta *Profile* do CCS, que retorna quantos ciclos de *clock* foram necessários para a execução do algoritmo. Este recurso também permite analisar a demanda de processamento de cada função o que permite saber quais são as funções críticas do seu algoritmo, possibilitando assim realizar otimizações no software.

O valor obtido por meio do cálculo é de 4 MIPS (Milhões de instruções por segundo), já a implementação comercial possui 3,2 MIPS.. Em relação à mipagem, o algoritmo implementado demonstrou certa inferioridade, tendo em vista que a implementação comercial possui menor mipagem.

Uma alternativa para alcançar o desempenho de 3,2 MIPS obtida pela implementação comercial é a utilização de instruções em paralelo, a fim de, reduzir os ciclos de *clock* necessários para a execução do algoritmo. Algo que se deve levar em consideração é que o algoritmo comercial foi desenvolvido por uma empresa líder no que se refere a projetar e desenvolver sistemas criptográficos para redes de dados.

6. Comentários e Conclusões

6.1. Conclusões

Este trabalho abordou a implementação do algoritmo *hash* SHA-1 para o processador digital de sinais TMS320C5410 da Texas Instruments.

Inicialmente foi realizado a análise das principais características do algoritmo *hash* SHA-1, de forma a facilitar sua implementação.

Posteriormente o algoritmo SHA-1 foi implementado em linguagem “C”. O funcionamento de tal implementação foi validado para as diferentes combinações possíveis de entrada. Este resultado permitiu que a implementação em linguagem “C” fosse utilizada como referência para a implementação do SHA-1 em linguagem assembly.

Na seqüência o algoritmo SHA-1 foi implementado em linguagem assembly para o DSP TMS320C5410, sendo seu funcionamento validado por meio de simulações no computador, utilizando arquivos no qual já se conhecia o resumo de mensagem. Finalmente foram realizados testes de desempenho, onde a complexidade computacional e a ocupação de memória mostraram-se compatíveis com uma implementação comercial.

Algo que deve ser levado em consideração é o fato de não ter sido encontrada nenhuma implementação em DSP do algoritmo SHA-1 desenvolvida por empresas nacionais. Assim, há um custo elevado para aquisição desta implementação no exterior, tornando a implementação desenvolvida neste trabalho uma alternativa interessante para empresas nacionais.

6.2. Trabalhos futuros

Algumas sugestões de trabalhos futuros são:

- Desenvolvimento do sistema de assinatura digital integrando o SHA-1 e o DSA.
- Desenvolvimento de algoritmos hash mais robustos como SHA-256, SHA-512.
- Implementação do algoritmo SHA-1 em hardware.

Referências

- Burnett, Steve, Paine, Stephen. (2002) “Criptografia e segurança O guia oficial RSA”, 1. ed. Rio de Janeiro: Campus. ISBN 85-352-1009-1.
- Dsptutor (2008) www.dsptutor.freeuk.com, Abril.
- Lima, Marcelo Ferreira de. (2005) “Assinatura digital solução delphi & capicom”. Florianópolis: Visual Books LTDA. ISBN 85-7502-171-0.
- Menezes, Alfred J., Oorschot, Paul C. van, Vanstone, Scott A.. (1996) “Handbook of Applied Cryptography”, New York: CRC Press, Inc., ISBN 0849385237.
- Terada, Routo. (2000) “Segurança de dados Criptografia em redes de computador”, 1. ed. São Paulo: Edgard Blücher LTDA. ISBN 8521202830.

- Ti. (2008)
<http://focus.ti.com/paramsearch/docs/parametricsearch.tsp?family=dsp§ionId=2&tabId=132&familyId=324>, Abril.
- Ti. (2008b) “Implementação comercial”.
<http://focus.ti.com/dsp/docs/thirdparty/catalog/embeddedsoftwarefulldetails.tsp?productId=134>. Abril.
- Ti. (2000) “TMS320VC5410 Fixed-Point Digital Signal Processor Data manual”.
focus.ti.com/lit/ds/symlink/tms320vc5410.pdf, Abril.
- Wolter, Stefan Klaus. (2007) “Processador digital de sinais”.
<http://www.eletrica.ufpr.br/marcelo/TE810/012007/Stefan-DSP.pdf>, Abril.
- Volpi, Marlon Marcelo. (2001) “Assinatura digital: aspectos técnicos, práticos e legais”.
Rio de Janeiro: Axcel Books do Brasil Editora LTDA., ISBN 85-7323-151-3.