Utilização do Conceito de Assinatura Digital na Segurança de Software

Rafael Luiz Xavier[1], Luís Augusto Mattos Mendes[1]

[1]Departamento de Ciência da Computação – Universidade Presidente Antônio Carlos (UNIPAC)
Campus Magnus – Barbacena – MG – Brasil

rafaelnordavind@gmail.com, luisaugustomendes@yahoo.com.br

Resumo. *Este artigo demonstra a utilização da assinatura digital como uma nova ferramenta de proteção para o software no combate à pirataria, fazendo um estudo breve e objetivo sobre a segurança de software, comparando os algoritmos mais utilizados para se chegar à conclusão de qual algoritmo é o mais indicado para atender a atual necessidade.*

Palavras-chave: assinaturas digitais; certificação digital; segurança de software; funções *hash* criptográficas.

1. Introdução

Nos tempos atuais, pode se observar a grande expansão da tecnologia da informação, e com essa expansão, surge também a necessidade de se ter uma maior segurança da informação. A perda dessas informações pode significar grandes prejuízos para toda uma organização, empresa ou usuário, podendo ser desde a cópia não autorizada de informações pessoais chegando até ao roubo de segredos empresariais, governamentais e militares. [1] A segurança da informação então, se preocupa em oferecer garantias de proteção como privacidade, confidencialidade, autenticidade, integridade e disponibilidade, para que eliminem os riscos, as ameaças e as vulnerabilidades que as pessoas, empresas e instituições estão sujeitas a todo o momento, preservando responsabilidades e protegendo reputações.

Em relação à segurança de software, existe uma grande necessidade ainda não efetivamente explorada pelas *Software Houses,*[1] que é combater a pirataria, que juntamente com a divulgação e crescimento dos softwares livres vem proporcionando grandes prejuízos a essas empresas. A pirataria dos softwares comerciais é a distribuição gratuita ou não, de forma ilegal dos chamados "*warez*" ou "*cracks"* que são os programas com as políticas de segurança quebradas, permitindo então o seu uso não autorizado. Os denominados *"crackers"* são os responsáveis por esta quebra de segurança, utilizando conceitos de engenharia reversa para obter o entendimento do sistema e então capturar, modificar ou recriar os requisitos implementados, e quebrar as políticas de segurança

[1] Software Houses - Empresas desenvolvedoras de softwares comerciais.

adotadas. Desta forma, eles podem fazer a disponibilização do *warez* para que usuários de todo o mundo possam utilizá-lo gratuitamente. Muitas vezes, esses *cracks* podem vir com códigos maliciosos embutidos, que podem danificar o computador de um usuário instalando algum software que venha a provocar algum tipo de prejuízo, como os *vírus*, *spywares*[2] e *trojans*[3].

Diante ao exposto, percebe-se a grande necessidade das empresas em aplicar políticas de segurança cada vez mais rigorosas, para garantir a integridade e confiabilidade do seu produto, assegurando que ele não seja utilizado por usuários não autorizados. E conseqüentemente obrigando os usuários interessados a adquirir o seu produto.

Visto que a principal característica das assinaturas digitais é proporcionar a garantia de integridade e confiabilidade da informação [2], este trabalho propõe a sua utilização no desenvolvimento do software para protegê-lo contra a tentativa de quebra de sua segurança, permitindo saber que o software desenvolvido não sofreu alterações em seu código original. Assim disponibilizando para as empresas uma arma adicional contra a pirataria de software, e para os usuários, a garantia de adquirir um software original sem possíveis alterações maliciosas.

O artigo está organizado em 6 seções: A seção 2 apresenta a conceituação das assinaturas digitais, descrevendo de forma objetiva todo o seu processo de funcionamento, falando também do principal uso das assinaturas digitais: a Certificação Digital; a seção 3 discorre sobre as funções *hash* criptográficas, que são algoritmos fundamentais para o processo das assinaturas digitais, mostrando suas características, aplicações, algoritmos mais utilizados e fazendo uma breve comparação entre eles para concluir qual algoritmo será utilizado no trabalho; a seção 4 apresenta as definições e os aspectos técnicos e legais a respeito da segurança de software mostrando a atual necessidade no cenário dos softwares comerciais e mostrando também um exemplo de como a assinatura digital poderia ser empregada como uma nova ferramenta de apoio à segurança do software; a seção 5 refere-se às considerações finais e sugere possibilidades para a pesquisa e desenvolvimento de trabalhos futuros que poderiam ser feitas nessa área; e por fim, na seção 6 é relatada a pesquisa bibliográfica que embasaram a construção deste trabalho.

2. Assinaturas Digitais

Em criptografia, a assinatura digital é um mecanismo de autenticação digital. Esses mecanismos foram criados com o objetivo de substituir a assinatura manuscrita, por uma que levasse para o mundo digital as mesmas garantias do mundo real. A simples digitalização de uma assinatura manuscrita e anexada a um documento não satisfaz esse propósito, pois ela pode ser facilmente copiada e anexada a qualquer documento, tornando-a simples de ser forjada [2].

As Assinaturas Digitais surgiram para suprir talvez a maior necessidade da tecnologia da informação: fazer com que o homem moderno utilize todos os benefícios que a tecnologia oferece enquanto meio de comunicação e para fins profissionais sem correr riscos aos qual a tecnologia o expõe.

---

[2] Spywares - Softwares espiões, que roubam informações dos usuários.
[3] Trojans - Softwares que abrem portas para possíveis invasões.

Para isso foi necessário evoluir os métodos de controle de acesso, autenticação e permitir garantias como as de sigilo e privacidade, integridade, autenticidade e confiabilidade para evitar, cada vez mais, atos ilícitos no meio digital. Diante disto, as assinaturas digitais permitiram que fosse criado o Certificado Digital.

O Certificado Digital atesta oficialmente a identidade de uma pessoa física ou de uma entidade jurídica, por meio de um documento assinado digitalmente [3]. Um Certificado digital válido precisa ser emitido por uma Autoridade Certificadora (AC): Presidência da Republica, Secretaria da Receita Federal, o Serpro, a Caixa Econômica Federal, a Serasa e a CertiSign; mediante preenchimento de formulário com os dados da pessoa, empresa ou instituição e pagamento de uma taxa. Depois o usuário deve procurar uma Autoridade de Registro (AR): Correios, Caixa Econômica Federal, Sincor, Banco do Brasil, Bradesco, Itaú, e Itautec; apresentar os seus dados pessoais para fazer o seu reconhecimento presencial.

Qualquer informação que foi assinada digitalmente por meio de um Certificado Digital válido tem garantias, inclusive jurídicas, de autenticidade e origem [3]. Torna as atividades virtuais mais seguras, como o uso do internet *banking,* declarações de imposto de renda, compras online. Em transações bancárias, por exemplo, é interessante que ambas as partes envolvidas tenham certeza da identificação inequívoca do outro.

Os Certificados Digitais garantem a autenticidade, integridade, confidencialidade entre as partes e o não repúdio, ou seja, o emissor não poderá negar a autoria e autenticidade das transações efetuadas e dos documentos assinados [3].

Existem diversas maneiras de assinar digitalmente um documento, mas a forma mais usual e eficaz envolve processos criptográficos utilizando algoritmos de funções unidirecionais ou *hash* criptográficas.

Essas funções são normalmente utilizadas para efetuar cálculos de integridade de mensagens. Nas assinaturas digitais, o seu uso permite a garantia de integridade do documento.

Além da integridade, uma assinatura digital também deve garantir a confiabilidade de um documento. Diante disto, uma assinatura digital utiliza algoritmos de criptografia assimétrica, que diferentemente dos algoritmos de criptografia simétrica, possuem não só uma única chave para criptografar e decriptografar um documento, e sim duas chaves, uma privada e outra pública que são usadas para criptografar e decriptografar respectivamente o documento.

A mensagem é cifrada através da chave privada do assinante. O algoritmo de chave pública garante que se uma determinada mensagem for cifrada com a chave privada, ela só poderá ser decifrada com sua chave pública correspondente.

Então, na prática o que se faz é aplicar o cálculo do algoritmo *hash* criptográfico em um documento e cifrar o *hash* resultante.

Logo, o processo de Assinatura Digital poderia ser da seguinte forma [2]:

Os passos para a criação são:

1. Obtém-se o Documento e a chave privada do assinante*;*
2. Obtém-se o *hash* (resumo) do documento e o criptografa usando a chave privada do assinante*;*
3. Envia-se o documento juntamente com a sua assinatura.

A Figura 1 demonstra a seqüência de criação da Assinatura Digital:

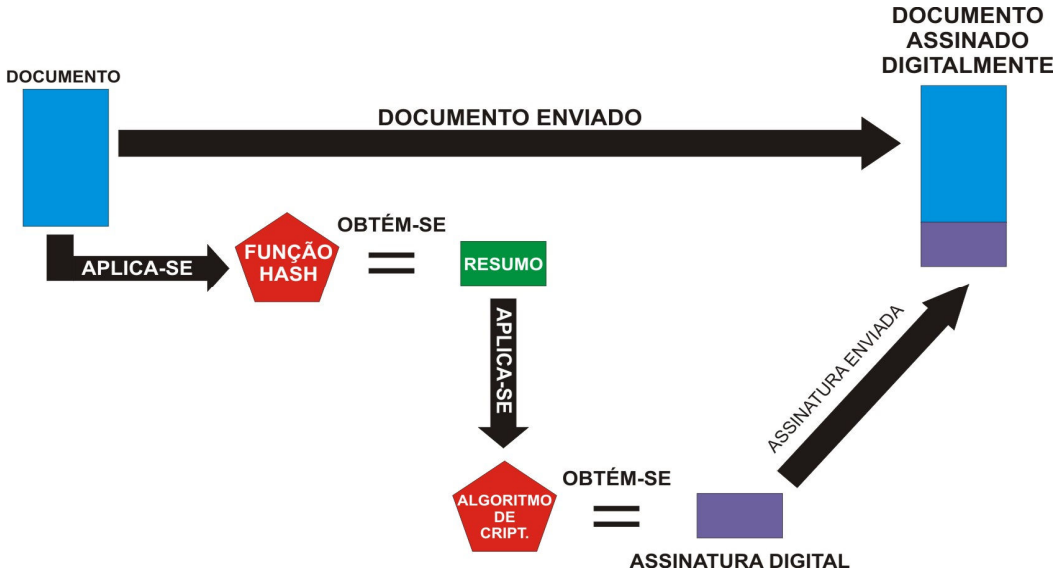

Figura 1. Processo de criação da Assinatura Digital

Os passos para o reconhecimento são [2]:

1. Obtém-se o documento a assinatura e a chave pública do assinante*;*
2. Obtém-se o *hash* original do documento decriptando a assinatura digital com a chave pública do assinante;
3. Compara-se o *hash* da mensagem original com o novo *hash* calculado. Se os dois *hashes* são iguais então a assinatura está correta.

A Figura 2 demonstra a seqüência de reconhecimento da Assinatura Digital:

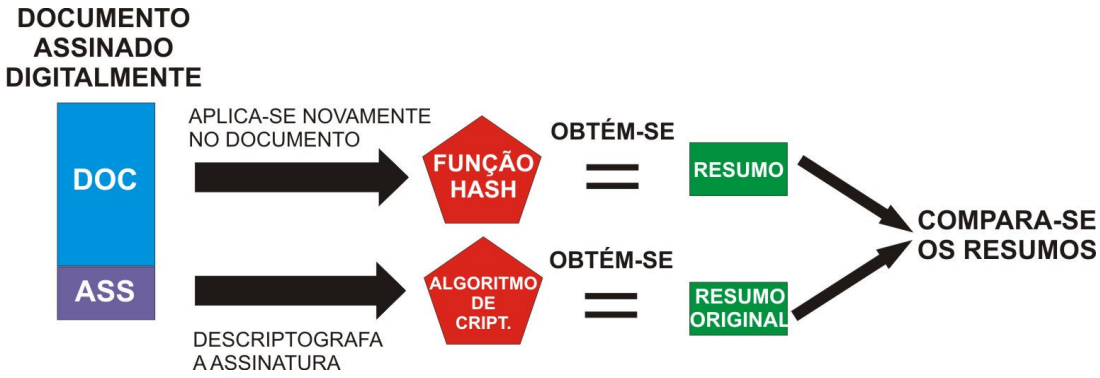

Figura 2. Processo de reconhecimento da Assinatura Digital

3. Funções Hash Criptográficas

As funções *hash* criptográficas, também denominadas *Message Digest*, *One-Way Hash Function*, Função de Condensação ou Função de Espalhamento Unidirecional é um mecanismo fundamental para as assinaturas digitais. Pois ao usar apenas algoritmos de chave pública e privada nas Assinaturas Digitais e ao assinar documentos grandes gastaria muito tempo devido à lentidão dos algoritmos de chave pública e privada. Ao invés disso, é empregada uma função *Hashing*, que gera um valor pequeno (um resumo), de tamanho fixo, derivado da mensagem que se pretende assinar, de qualquer tamanho. Assim, a função *Hashing* oferece agilidade nas Assinaturas Digitais.

O conceito teórico diz que *hash* é a transformação de uma grande quantidade de informações em uma pequena quantidade de informações.

3.1. Características

Uma função *hash* criptográfica deve possuir as seguintes características:

- Deve ser capaz de transformar uma mensagem de qualquer tamanho em um resumo ininteligível de tamanho fixo [2];
- Deve ser fácil de calcular, para garantir que a sua aplicação não demande tempo;
- Deve ser impossível encontrar a mensagem original através da *hash* da mensagem, ou seja, é computacionalmente inviável, dado y, encontrar x tal que $h(x) = y$;
- Deve ser computacionalmente inviável, dado x_1, encontrar $x_2 \neq x_1$ tal que $h(x_1)=h(x_2)$;
- Deve ser impossível também encontrar duas mensagens diferentes que levam a um mesmo *hash*, a esse fato dá se o nome de colisão, ou seja, é computacionalmente inviável, encontrar quaisquer x_1 e x_2 tais que $h(x_1)=h(x_2)$;
- Outra característica importantíssima de uma função *hash* criptográfica é a de que qualquer alteração da mensagem original, por menor que seja (1 bit, por exemplo.) gera uma alteração significativa no valor do *hash* correspondente.

3.2. Aplicações

Algoritmos *hash* com melhor desempenho se adaptam melhor para aplicações que exigem como prioridade a velocidade, o tempo de resposta e cuja função é somente para cálculos de integridade como, por exemplo: em Redes, onde são usados para verificar se o arquivo não foi alterado ou corrompido durante a transmissão dos dados; em aplicações Antivírus, onde é utilizado para identificação de vírus e para proteção da integridade de arquivos; em Banco de Dados, protegem as senhas no banco, onde o valor *hash* é armazenado no lugar da senha.

Já outras aplicações como as Assinaturas Digitais e os Certificados Digitais necessitam não só de desempenho, mas também existe uma preocupação quanto à segurança, pois se tratam de aplicações que envolvem conceitos de confiabilidade, onde pessoas, entidades ou empresas necessitam garantir integridade e autoria de informações.

3.3. Algoritmos

As rotinas para o cálculo da integridade são feitas por algoritmos *hash* criptográficos, e existem vários algoritmos desenvolvidos para esse propósito.

A Tabela 1 apresenta os algoritmos mais conhecidos atualmente e apontam algumas características básicas desses algoritmos:

Tabela 1. Algoritmos *hash* mais conhecidos

Algoritmos					
MD4	**MD5**	**SHA-1**	**RIPEMD**	**WHIRLPOOL**	**HAVAL**
Criado por Ron Rivest da empresa RSA; Entrada de qualquer tamanho Saída de 128 bits; Bastante rápido;	Extensão do MD4; Entrada de qualquer tamanho; Saída de 128 bits; Rápido; Já foram encontradas fraquezas.	Criado pelo NIST; Entrada de qualquer tamanho; Saída de 160 bits; Existem versões posteriores: SHA-256, SHA-384, SHA-512; Considerado forte.	Criado pelo projeto RIPE; Entrada de qualquer tamanho; Saída de 160 bits; Considerado forte.	Criado por Vincent Rijmen e Paulo S. L. M.Barreto; Entrada de qualquer tamanho; Saída 512 bits; Considerado forte.	Criado por Yuliang Zheng, Josef Pieprzyk e Jennifer Seberry; Entrada de qualquer tamanho; Saída de 128, 160, 192, 224 e 256 bits; Frágil na versão de 128 bits.

Cada algoritmo possui suas características que o difere dos demais e para cada aplicação distinta, um algoritmo é mais indicado para a utilização.

Como visto anteriormente as características que mais envolvem na escolha de um algoritmo *hash* geralmente são relacionadas ao desempenho do algoritmo e na sua segurança, ou seja, na capacidade de resistir aos ataques para encontrar colisões.

Diante disto, será realizado um estudo comparativo entre as características dos algoritmos MD5 e SHA-1 para que se possa optar pelo algoritmo que se enquadre melhor aos requisitos deste trabalho. Pois se tratam dos algoritmos mais conhecidos e difundidos no mercado.

3.3.1. MD5

O algoritmo MD5[4] foi projetado para ser bastante eficiente em máquinas de arquitetura de 32 bits (como a Intel, por exemplo) e para softwares de alto-desempenho. Este algoritmo recebe como entrada um documento qualquer sob a forma digital com um tamanho de até 2^{64} bits -1 (18.446.744.073.709.551.615 bits) ou 2.305.843.009.213.693.952 "Bytes" ou caracteres, e gera como saída um resumo de tamanho 128 bits. Na execução do algoritmo, primeiramente se inicia um vetor com 4

variáveis de 32 bits A, B, C e D com os respectivos valores fixados pela definição do algoritmo: *A:0x01234567, B: 0x 89abcdef, C: 0xfedcba98 e D: 0x76543210.*

Logo depois se verifica o tamanho da entrada e concatena-se com bits de ajuste para que a entrada tenha um tamanho múltiplo de 512. Depois a entrada é dividida em blocos de 512 bits e cada bloco de 512 é subdividido em 16 blocos de 32 bits. Os blocos são processados e o valor *hash* é dado pela concatenação das variáveis *A, B, C* e *D* resultando um *hash* de 4*32 =128 bits.

Logo o parâmetro de segurança do MD5 em relação à resistência a colisões segundo o tamanho da sua saída de 128 bits é de 2^{64}.

Uma equipe de pesquisadores da Shandong University, China, liderada pela pesquisadora Xiaoyum Wang, anunciou na *CRYPTO 2004* (Conferência Internacional de Criptologia) que poderiam achar várias colisões no MD5 na ordem de 2^{37} passos. Logo, estava estampado que o algoritmo apresenta vulnerabilidades [5].

3.3.2. SHA-1

O algoritmo SHA-1[6] funciona em alguns pontos de forma semelhante ao MD5. Ele trabalha com blocos 512 bits, realizando o mesmo processo de verificação de tamanho e ajuste que o MD5 realiza, inclusive na concatenação de uma string binária que representa o tamanho da entrada, de tamanho 64 bits. Portanto a entrada pode ter tamanho de até 2^{64} – 1 bit. Além do mais, ele divide cada bloco de 512 bits em sub-blocos de 16 por 32 bits da mesma forma que o MD5. Mas as semelhanças acabam por aqui, pois existem várias diferenças. Por exemplo, a saída do algoritmo resulta numa string binária de 160 bits, diferente do MD5, que tem como saída uma string binária de 128 bits. No SHA-1 cinco variáveis são iniciadas com os respectivos valores:

A: 0x67452301, B: 0xefcdab89, C: 0x98badcfe, D: 0x10325476 e E: 0xe3d2e1f0

O processamento dos blocos também ocorre de forma diferente e após esse processamento o valor *hash* é dado pela concatenação das variáveis *A, B, C, D* e *E* resultando um *hash* de 5*32 =160 bits.

Logo o parâmetro de segurança do SHA-1 em relação à resistência a colisões segundo o tamanho da sua saída de 160 bits é de 2^{80}.

Já o SHA-1 possui um desempenho um pouco mais lento, devido a possuir mais etapas de processamento, mas o algoritmo oferece uma maior segurança. Atualmente, não há nenhum ataque de criptoanálise conhecido contra o SHA-1 que tenha surtido efeito em tempo hábil. O melhor parâmetro de ataque contra o SHA-1 foi encontrado pelos mesmos pesquisadores que anunciaram o parâmetro de ataque de 2^{37} passos para o MD5 em conjunto com Andrew Yao e Frances Yao na *CRYPTO 2005*. Eles encontraram para o SHA-1 o parâmetro de ataque na ordem de 2^{63} passos, mas mesmo assim o ataque da força bruta torna-se impraticável, devido ao seu valor *hash* de 160 bits, visto o tempo e recurso necessário para se achar uma colisão. Para se ter uma idéia, mesmo um computador executando 1 Mega SHA-1's por segundo(1024^2 execuções é uma super-estimativa, serve para se ter uma idéia do limite superior de tempo), levaria 2^{43} segundos para se achar uma colisão e como um ano possui aproximadamente 1 Giga segundos, uma colisão seria achada em 8192 anos [5].

Porém, não há provas de que, no futuro, alguém não possa descobrir como quebrar o SHA-1 em tempo hábil, visto que o poder computacional irá aumentar e o uso de sistemas distribuídos e computação paralela reduziriam drasticamente os esforços para a quebra do algoritmo [5]. Se no futuro ataques forem bem sucedidos contra o SHA-1, seus criadores recomendam a migração para os algoritmos da família SHA que possuem um valor *hash* de saída maior como o SHA-256, SHA-384 e o SHA-512, o que dificultaria ainda mais ataques.

3.3.3. Comparação entre MD5 x SHA-1

A partir da pesquisa realizada foi possível destacar as características mais importantes relacionadas ao trabalho, onde os requisitos necessários se baseiam no desempenho e na segurança dos algoritmos.

Outro aspecto observado foi a de se utilizar tecnologias mais difundidas no mercado a fim de poder garantir a interoperabilidade e segurança relativa ao uso de algoritmos confiáveis que estão amplamente sendo testados e aprovados pelo mercado.

A Tabela 2 demonstra a comparação das principais características dos algoritmos MD5 e SHA-1:

Tabela 2. Comparação das Características

MD5				
Menos etapas de processamento	Menor saída (128 Bits) Melhor performance	Parâmetro de segurança menor (2^{64})	Colisões encontradas em 2^{37} passos.	Logo, Colisões são encontradas em tempo hábil.
SHA-1				
Mais etapas de processamento	Maior saída (160 Bits) Pior performance	Parâmetro de segurança maior (2^{80})	Colisões encontradas em 2^{63} passos.	Logo, Colisões são encontradas em 8192 anos.

Visto que o algoritmo MD5 possui menos etapas de processamento e menor saída do que o SHA-1 pode se concluir que o MD5 é o mais rápido. Entretanto, o fato dele produzir um valor *hash* de somente 128 bits faz com que seu parâmetro de segurança seja apenas de 2^{64} sendo assim mais fácil encontrar uma colisão em tempo computacional viável, tornando o algoritmo menos seguro em relação ao SHA-1.

Para satisfazer a necessidade da utilização da Assinatura Digital, como um mecanismo de proteção ao software, a segurança torna-se um aspecto de maior relevância que o desempenho, onde se precisa garantir integridade, confiabilidade e autenticidade.

Após esse estudo comparativo entre os algoritmos mais utilizados pode concluir que o SHA-1 é o mais indicado, pois ele oferece uma maior segurança comparada ao MD5.

4. Segurança de Software

Com a grande expansão da tecnologia da informação, nos tempos atuais, a segurança tornou-se um aspecto cada vez mais preocupante, pois a necessidade de proteção, privacidade e confiabilidade da informação está crescendo. Logo se inicia a infinita busca por proteções mais eficientes para assegurar que elas não sejam mais "quebradas" e garanta assim a segurança do software.

4.1. A História do Cracking

A história do software *cracking* se inicia juntamente com a própria história do software comercial [7]. No entanto, o *cracking* efetivamente evoluiu em um grande cenário *underground*[4] no começo dos anos 80. Assim como as diversas comunidades digitais, os primeiros *hackers*[5] envolvidos na quebra de segurança dos softwares proprietários se juntaram em grupos compostos de:
CRACKERS, que são os próprios programadores que quebravam as proteções de segurança dos softwares;
SUPPLIERS (Fornecedores), que são pessoas que tinha acesso aos softwares e muitas vezes antes mesmo do seu lançamento;
TRADERS ou *COURIERS*, que eram pessoas que faziam a distribuição dos *cracks* o mais rápido possível.

Esses grupos eram alimentados pelo ego, disputavam entre si a fama de quem distribuía mais rápido a última versão de softwares ou jogos e em raras vezes faziam *cracks* com fins lucrativos [7]. Foram também propulsores de outras comunidades de *crackers*, como por exemplo, os *phreakers* que burlavam o sistema de telefonia pública fazendo ligações gratuitamente, os *hackers* de sistemas que quebravam a segurança da rede para se ter acesso a máquinas pela internet e os criadores de vírus.

4.2. O Cracking e a Engenharia Reversa

"Nem toda atividade *cracker* tem relação com a engenharia reversa, e nem sempre a engenharia reversa é ligada a atividades *crackers*. No entanto, ambas tem laços muito estreitos, e usualmente são associadas, já que a grande responsável pela atividade *cracker* é a engenharia reversa, e a grande aplicação da engenharia reversa é a atividade *cracker*" [7].

A engenharia reversa, ou reengenharia, como o próprio nome indica, é a engenharia "ao contrário", é o processo inverso da tradicional engenharia de software progressiva. Ela é caracterizada pelas atividades retroativas do ciclo de vida do software, partindo de um alto nível de abstração, para um nível mais baixo. Utiliza-se a Engenharia Reversa de Software nos seguintes casos:

1. Para adaptar o software a novos computadores;
2. Para atualizar o software (novas bibliotecas, novas linguagens de programação, novas ferramentas);

[4] Underground - Cenário Escondido, restrito, sem divulgação para a maioria da população.
[5] Hackers - Pessoas com conhecimentos avançados em computação, capazes de proteger e burlar sistemas.

3. Para adaptar o software a novas regras;
4. Para disponibilizar novas funcionalidades;
5. Para corrigir *bugs* (erros de programação encontrados nos softwares)
6. Para permitir a interoperabilidade dos softwares, desenvolvimento de novas tecnologias e derivações tecnológicas.

Mas os *crackers* na maioria das vezes utilizam os conceitos de engenharia reversa para entender o funcionamento do software para então poder copiá-lo, aperfeiçoá-lo, modificá-lo, ou então quebrar a segurança de programas que exijam algum tipo de registro, podendo também liberar funções que estejam bloqueadas em versões *demos*[6] e distribuí-los gratuitamente.

"A Engenharia Reversa, além de motivações econômicas, militares, ou mesmo por necessidade, costuma normalmente ocorrer como resultado da curiosidade humana. As pessoas fazem para "ver se conseguem", por assim dizer. O ser humano tem uma necessidade de compreensão e de domínio sobre o mundo ao seu redor que causa avanços em todas as áreas de conhecimento humano, e é essa necessidade que impulsionou as bases da Física, da Filosofia, e de outras ciências" [7].

4.3. Aspectos Técnicos

"A checagem da autenticidade é o "coração" da maioria dos mecanismos de proteção. As empresas desenvolvedoras de softwares comerciais querem ter a certeza de que quem está usando o seu software é a pessoa autorizada que adquiriu o direito de usar o software. A palavra "pessoa" pode não significar apenas um usuário, e sim a totalidade de usuários que possuem a licença e a cópia do programa" [8].

4.3.1. Proteções

A cópia é protegida por direitos de *copyrights*[7], mas existem casos em que a cópia é permitida como, por exemplo, em *backups* (cópia de salvaguarda) de programa. Mas o que os *crackers* fazem é quebrar a segurança contra a cópia e uso não autorizados para ter livre acesso ao programa mesmo sem comprar a sua licença e ainda disponibilizá-lo na internet.

As proteções podem ser classificadas em:

1. Proteção baseada em conhecimento: como números de série, números de registro e senhas de usuários;
2. Proteção baseada em posse: como diskettes e cd's com chaves, chaves eletrônicas, *tokens*[8], cartões inteligentes. Como por exemplo, jogos que necessitam do cd original no drive para que sejam utilizados [8].

Com todas essas proteções ainda sim não é possível proteger o software 100% dos ataques dos *crackers*. Visto que a proteção se baseia em rotinas de software, uma vez que o software é completamente "descompilável", através da engenharia reversa, é possível

[6] Demos - Versão do software, às vezes com limitações, com o objetivo de demonstrar ao usuário o software.
[7] Copyrights - Lei de direito autoral que protege contra a cópia não autorizada de uma informação.

[8] Tokens – Dispositivo eletrônico que gera uma nova senha a cada 36 segundos oferecendo privacidade contra o roubo de senhas.

reconstruir o código dos programas a partir de seu executável. Logo toda proteção é quebrável, demandando tempo e esforço.

4.3.2. Técnicas de Proteção

Muitas técnicas utilizadas atualmente como os *packers* (programas que empacotam o executável para proteger o código original), criptografia e checagem de integridade através de CRC (checagem cíclica de redundância), prometem proteger o software contra a pirataria. Mas diversas técnicas já fracassaram, e não existe hoje, um padrão de desenvolvimento que garanta uma proteção absoluta.

A maioria das pessoas são usuários comuns, e incapazes de *crackear* um software, pois o processo de *cracking* não é trivial e exige bastante conhecimento. O máximo que podem fazer é buscar *cracks* em sites *warez* [7].

Mas existem os que são capazes, e o que se pode fazer é criar novos métodos para aumentar a dificuldade da quebra das proteções.

4.3.3. Ferramentas

As ferramentas mais comuns usadas pelos crackers para se fazer a reengenharia são:

EDITORES HEXADECIMAIS ou DUMPERS: Exibem o conteúdo binário de um software em formato hexadecimal. É utilizado para modificar strings e instruções.

DISASSEMBLERS: Lê o código binário e mostra cada instrução executável referente ao código. Os disassemblers mais modernos fazem análises complexas sobre os códigos de máquina e determinam diversas estruturas de programação, como *ifs*, *switchs*, e estruturas de dados comuns;

DEBUGGERS: Possui funcionamento semelhante ao disassembler, mas ao invés de fazer análise do código ele mostra as instruções e permite que o programa seja executado passo a passo, e que o *cracker* faça modificações (em tempo de execução) nas instruções do programa.

DESCOMPILADORES: Para reconstruir código em uma linguagem de alto nível utiliza-se um descompilador, que faz o *disassembly*[9] do programa e após isso faz uma análise baseada em características específicas da linguagem/compilador utilizados na compilação do software;

SYSTEM HOOKERS: Software que se "anexa" em determinadas posições de memórias do sistema operacional e passa a monitorar chamadas a determinadas funcionalidades. Pode-se monitorar, por exemplo, todas as aberturas de arquivos, ou leituras do registro do Windows. É uma das técnicas para identificar como o software faz a verificação de autenticidade;

UNPACKERS: Desempacotam o software criptografado, e em algum instante o software acaba sendo decriptado completamente em memória, e basta fazer um "*snapshot*"(fotografia) da memória neste instante para termos uma versão decriptada e *crackeável* do software.

[9] Disassembly - Traduz o executável para instruções de linguagem Assembler.

4.4. Aspectos Legais

Uma questão muito importante é também saber quais atos provenientes da engenharia reversa são considerados lícitos ou ilícitos.

No Brasil, os softwares são regulamentados pela lei 9609 de 19 de fevereiro de 1998 que dispõe sobre a proteção da propriedade intelectual do programa de computador. [9] No seu artigo 6º a lei diz não constituir ofensa aos direitos do titular do software os seguintes casos:

I - a reprodução, em um só exemplar, de cópia legitimamente adquirida, desde que se destine à cópia de salvaguarda ou armazenamento eletrônico, hipótese em que o exemplar original servirá de salvaguarda;

II - a citação parcial do programa, para fins didáticos, desde que identificados o programa e o titular dos direitos respectivos;

III - a ocorrência de semelhança de programa a outro, preexistente, quando se der por força das características funcionais de sua aplicação, da observância de preceitos normativos e técnicos, ou de limitação de forma alternativa para a sua expressão;

IV - a integração de um programa, mantendo-se suas características essenciais, a um sistema aplicativo ou operacional, tecnicamente indispensável às necessidades do usuário, desde que para o uso exclusivo de quem a promoveu.

Esta lei não especifica as infrações e penalidades que podem ocorrer caso alguém faça engenharia reversa para produzir *cracks* e o distribuir ou disponibilizá-lo para fins comerciais ou não. E também não contempla a hipótese de se utilizar o software com finalidade experimental, relacionada com estudos ou pesquisas tecnológicas. Para esses casos a LPI (Lei de Propriedade Industrial) pode ser aplicada visto que o software também é considerado como propriedade industrial podendo ser registrado e patenteado. E de acordo com a LPI a engenharia reversa de patentes é considerada lícita quando se utiliza para fins experimentais, relacionados a estudos ou pesquisas científicas ou tecnológicas. E é considerada ilícita a divulgação e exploração de conhecimentos, informações ou dados confidencias sem autorização. E é vedada a quem possuir a licença fazer ou permitir engenharia reversa, desmontagem e descompilação dos programas.

Nos Estados Unidos existe uma lei que proíbe (torna crime) a produção e disseminação de tecnologias que possam driblar medidas tomadas para proteger o *copyright*, e aumenta a pena para a quebra de *copyrights* na Internet e da produção e disseminação de tecnologias ou conhecimento.

Um exemplo foi a quebra do formato de encriptação do DVD. Um adolescente de 16 anos na Noruega criou programa para decriptar DVDs, e por estar na Noruega, ele não estava sujeito as leis americanas. No entanto, foi julgado por infração de *copyright* em seu próprio país, e o seu programa somente foi proibido de ser hospedado em web sites americanos.

Já a Rússia é um país mais flexível em relação aos *copyrights* e não é considerado crime a divulgação de certas informações, como por exemplo, a distribuição gratuita de mp3, de filmes e de programas, como se pode observar a infinidade de sites russos com essa finalidade.

Como a legislação varia de lugar para lugar em todo mundo, é impossível o controle definitivo da pirataria, visto que o mundo todo é interligado pela internet. E no Brasil as leis

ainda são fracas visto que as tecnologias avançaram em um grau muito maior que as nossas leis.

4.5. Exemplo da Implementação da Assinatura Digital na Segurança do Software

A Implementação da Assinatura Digital como mecanismo de proteção para a segurança do software pode ser da seguinte maneira:

1. O Software cliente, que é vendido para os usuários, e que sofre ataques dos *crackers,* é implementado com funções que ao ser executado é iniciado uma verificação da integridade do código aplicando-se o cálculo do *hash* utilizando o algoritmo SHA-1 que foi estabelecido como o mais adequado à aplicação;
2. Após obtenção do hash do software cliente, criptografa-se o *hash* usando a chave privada da aplicação;
3. Em seguida, envia-se para a aplicação servidora o *hash* criptografado;
4. Descriptografa-o usando o algoritmo de chave pública para se obter o *hash* da aplicação cliente;
5. Para então comparar o *hash* original armazenado no servidor, que é o *hash* que a aplicação cliente deverá ter, com o *hash* recebido. Se os *hashes* forem os mesmos a aplicação servidora autentica e o software cliente poderá ser executado. Caso contrário o software não será executado.

A Figura 3 demonstra o exemplo de implementação da Assinatura Digital como mecanismo de proteção para a segurança do software, citado acima:

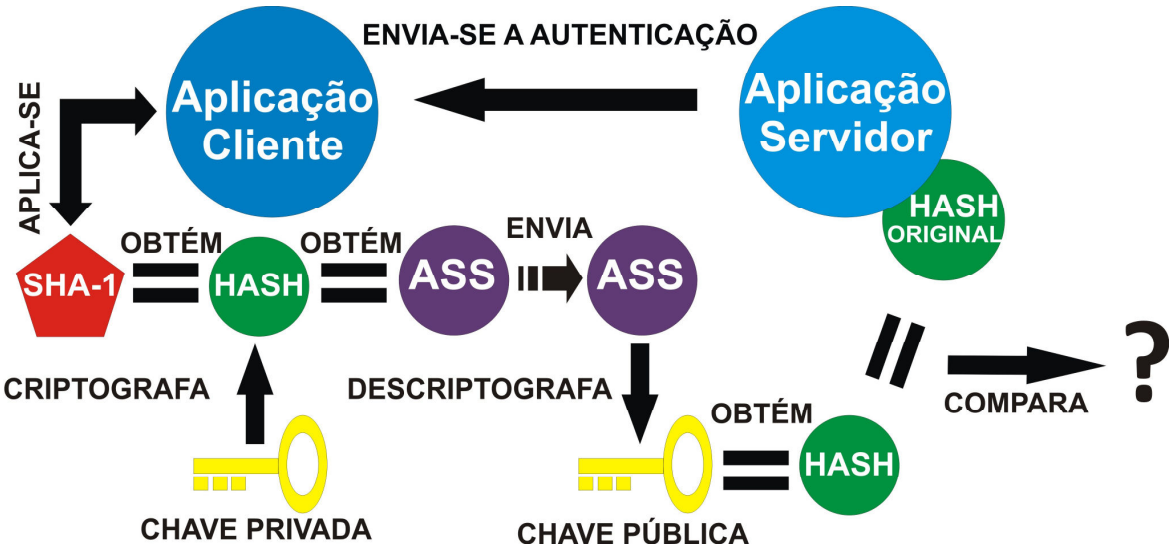

Figura 3. Exemplo da implementação da Assinatura Digital

5. Considerações Finais

No contexto atual, mais do que nunca, a segurança da informação é primordial para o bem estar e o sucesso das pessoas, empresas e instituições. Este trabalho conclui que a análise em um contexto mais específico como a segurança do software, demonstra que as políticas de segurança adotadas pelas *Softwares Houses* a fim de proteger a sua propriedade intelectual (o seu software comercial) tem se mostrado pouco eficientes, visto a infinidade de softwares *cracks* disponíveis na internet. Logo, novas pesquisas de segurança de software em relação aos métodos de proteção vêm sendo necessárias para proporcionar a essas empresas uma maior tranqüilidade e estabilidade diante do mercado, que se torna cada vez mais competitivo a todo momento.

Este trabalho se resume na pesquisa realizada sobre as Assinaturas Digitais e Certificações Digitais que são ferramentas que possuem propriedades fundamentais que garantem certos requisitos que a segurança da informação exige, tais como a confidencialidade, integridade e a autenticidade. A pesquisa também mostra que essas ferramentas não possuem vulnerabilidades computacionais consideráveis se utilizarmos nas Assinaturas Digitais algoritmos criptográficos considerados fortes. Logo, é justificada a escolha do algoritmo *hash* criptográfico SHA-1 para compor as assinaturas utilizadas neste trabalho, analisadas a sua segurança e o seu desempenho comparados a outros algoritmos como o MD5.

Como existe a necessidade de utilização de ferramentas ainda não quebradas pelos *crackers,* se pode pensar em utilizar as assinaturas digitais como uma nova ferramenta a ser implementada junto ao desenvolvimento do software a fim de obter uma melhor proteção para o software e garantir os direitos de autoria dos desenvolvedores.

Enfim, este trabalho deixa como sugestão para trabalhos e pesquisas futuras os seguintes temas que podem vir a ser desenvolvidos: Utilização do Conceito de Assinatura Digital e Certificação Digital nos métodos de Autenticação de Redes Wireless em Ambientes Corporativos e Desenvolvimento de Normas e Padrões de Engenharia de Software Para a Avaliação da Qualidade dos Métodos de Proteção ao Software;

6. Bibliografia

[1] CAMPOS, Fernanda da S. Fialho. **Responsabilidade Legal na Era Digital**. Módulo Security Magazine. Disponível em: <http://www.modulo.com.br> Acesso em: 10/08/08.

[2] ESEC, Tecnologia em segurança de dados. **Princípios de Assinatura Digital.** Brasília. 2001.

[3] NEXTG. **Certificação Digital.** Centro de Treinamento da Intel. 2008. Disponível em : <www.nextg.com.br> Acesso em: 25/08/08.

[4] RIVEST92, R.L. Rivest, **The MD5 Message Digest Algorithm,** RFC 1321, Abr 1992.

[5] VALADARES, Francisco de Assis Mesquita. **Uma avaliação Crítica sobre os Ataques às Funções MD5 e SHA-1**. Trabalho de graduação em ciência da computação – Universidade Federal de Pernambuco. 2006.

[6] FIPS180-2 National Institute of Standards and Technology, NIST **FIPS PUB 180-2,Secure Hash Standard**, U.S. Department of Commerce, Ago 2002.

[7] DRIZIN, Ricardo; BUONANNI, Ulisses. **A Cultura Cracker e a Engenharia Reversa**. 2005. Disponível em: <http://conhecimento.incubadora.fapesp.br> Acesso em: 10/03/08.

[8] KASPERSKY, K. **Hacker Disassembling Uncovered**. A-LIST Publishing, 2003.

[9] BRASIL. **Lei nº. 9.609, de 19 de fevereiro de 1998**. Dispõe sobre a proteção da propriedade intelectual de programa de computador, sua comercialização no País, e dá outras providências. Disponível em:<http://legislacao.planalto.gov.br/ > Acesso em: 10/03/08.

[10] OICILIS. **Coletânea de referencias para Segurança de Software**. 2003. Disponível em: <http://numaboa.com.br/informática/oicilis/> Acesso em: 10/03/08.

[11] PRESSMAN, Roger S. **Engenharia de Software.** Tradução de José Carlos Barbosa dos Santos. 3. Ed. São Paulo Makron Books, 1995.

[12] TANENBAUM, A, S. **Redes de Computadores**. Rio de Janeiro: Campus, 2003.