fonte: http://www.cert-rs.tche.br/docs_html/autentic.html

Autentificação

O que é Autentificação?

Quando você recebe uma mensagem de correio eletrônico, como você pode saber se é da pessoa que pretendeu enviá-la? Você não sabe. É claro, a mensagem tem um cabeçalho, mas o cabeçalho pode ter ser falsificado. Isso é fácil para um hacker não tão habilidoso que deseja que sua mensagem venha de alguém que ele gosta e de qualquer lugar que ele goste e faça isso tão bem que você seja enganado. Nunca é sábio acreditar completamente na linha de "From" de qualquer mensagem de e-mail.

Programas de segurança de e-mail podem garantir, ou autentificar, que uma certa mensagem é de fato de alguma pessoa de quem o nome aparece na linha de "From". Isto é algumas vezes conhecido como "autentificação de dados da origem" e é feito com alguma coisa chamada de *assinatura digital*. Antes de verificar ao equivalente digital, vamos olhar para as assinaturas manuscritas.

Autentificação no Papel

Assinaturas manuscritas em documentos foram muito tempo usadas como prova de autoria de, ou pelo menos de acordo com, os conteúdos do documento. O que é a assinatura em um documento e como o ato de assinar é tão compelido?

1. A assinatura não pode ser falsificada. A assinatura prova que o signatário assinou deliberadamente o nome dele no documento.
2. A assinatura é autêntica. A assinatura pode convencer o recipiente do documento que o signatário realmente assinou o nome dele ou dela nele.
3. A assinatura não é reutilizável. A assinatura é escrita no documento e não pode ser movida para um documento diferente. Em outras palavras, uma pessoa sem escrúpulos não pode mover uma assinatura de um documento para um documento completamente diferente.
4. O documento assinado é inalterável. Depois que o documento foi assinado, não pode ser alterado. Isto não é realmente verdade, mas se uma carta datilografada e assinada tem mudanças manuscritas substantivas no seu corpo, alguém vai se tornar suspeito.
5. A assinatura não pode ser repudiada. A assinatura e o documento são coisas físicas. O signatário não pode depois reivindicar que ele ou ela não assinou isto.

De fato, isto não é necessariamente verdadeiro para assinaturas manuscritas: Falsificação é possível, assinaturas podem ser fotocopiadas de um documento e podem ser postas em outro, documentos podem ser alterados facilmente depois da assinatura e as pessoas negam que assinam algo o tempo todo. Mas esta é a idéia.
Isso é até pior em computadores. "Atachando" simplesmente um bitmap de uma assinatura manuscrita a um documento de um computador não funciona. O documento poderia ser alterado facilmente depois da assinatura ser "atachada" e ninguém poderia descobrir isto. Até pior, a assinatura poderia ser facilmente copiada sobre outros documentos.

Assinaturas Digitais

Uma assinatura digital é uma seqüência de bits adicionados a documentos digitais (Veja a Figura 1).

Assim como a assinatura manuscrita, sua autenticidade pode ser verificada, mas distinto da assinatura manuscrita, ela é única ao documento sendo assinado. A assinatura manuscrita de alguém parece quase a mesma coisa, repetidas vezes. Aquela assinatura digital de uma pessoa parece diferente para cada documento diferente que ela assina.

Version: 2.5

Assinaturas digitais são outra aplicação de criptografia de chave pública. Se você se lembra, criptografia de chave pública utiliza uma chave pública e privada. Qualquer um pode codificar mensagens com a chave pública de Bob, mas só Bob pode decifrar mensagens com a sua chave privada.

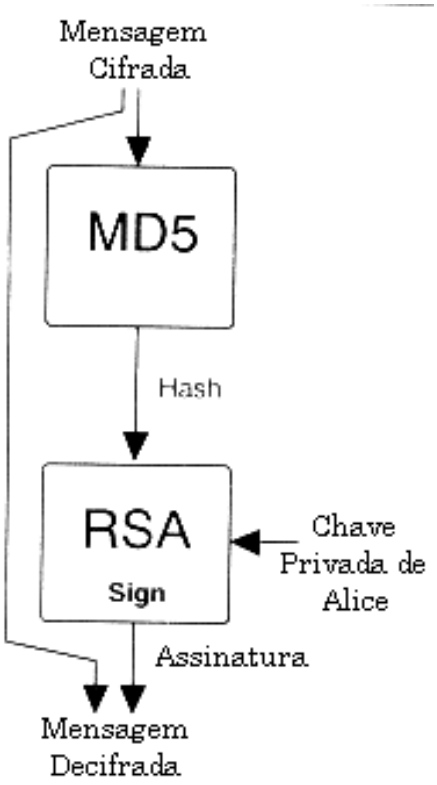

Figura 1. Assinando um mail eletrônico.

Existe outra coisa que você pode fazer com criptografia de chave pública.
Você pode inverter os papéis da chave pública e privada. A Alice poderia codificar a sua mensagem com a sua chave privada e poderia enviar isto para Bob. Esta não é nenhuma comunicação segura. Bob, ou qualquer um outro no que diz respeito ao assunto, poderia decifrar a mensagem com a chave pública de Alice e poderia ler isto. (Lembre-se, a chave pública dela é pública. Está extensamente disponível.) Mas ninguém mais tem a chave privada de Alice. Ninguém mais poderia ter codificado isto, e isto autentica a origem da mensagem.

Esta é a idéia central e é melhor discutí-la com alguma profundidade. Só a Alice conhece sua chave privada, logo somente ela pode codificar a mensagem. Todo mundo conhece a chave pública de Alice, assim qualquer um pode decifrar a mensagem. Codificando a mensagem com a chave privada dela, ela está efetivamente assinando a mensagem. Ela está fazendo algo com a mensagem que somente ela pode fazer.

Quando Bob decifra a mensagem com a chave pública de Alice, ele verifica a assinatura dela, o que autentica a origem da mensagem. Para a decifragem produzir uma mensagem legível, ela deve ter sido assinada pela chave privada de Alice. Qualquer um pode verificar a assinatura de Alice.

As assinaturas digitais satisfazem os cinco critérios das assinaturas de papel que foram listadas anteriormente:

1. A assinatura não pode ser falsificada; só Alice conhece sua chave privada.
2. A assinatura é autêntica; quando Bob verifica a assinatura com a chave pública de Alice ele sabe que ela assinou (codificou) isto.
3. A assinatura não é reutilizável; a assinatura em um documento não pode ser transferida para qualquer outro documento.
4. O documento assinado é inalterável; qualquer alteração de um documento (se ele foi ou não codificado) e a assinatura não é mais válido.
5. A assinatura não pode ser repudiada. Bob não precisa da ajuda de Alice para verificar sua assinatura.

Existe uma simplificação importante na explicação acima: assinar não significa realmente codificar com a chave privada e a verificação não é realmente decifrar com a chave pública. A matemática geralmente é mais complicada.

Você deve pensar melhor nestas operações de autenticação bem como a assinatura e a verificação e praticá-las.

Algoritmos de Assinatura Digital

Da mesma maneira que muitos algoritmos de chaves públicas diferentes realizam administração de chaves, muitos algoritmos de chaves públicas diferentes fazem assinaturas digitais. Serão discutidas somente as que sejam provável que você veja em uso, ou em versões atuais ou futuras do PGP ou do PEM.

RSA

Além de codificar chaves, o algoritmo de chave pública RSA pode ser também usado para gerar assinaturas digitais. As matemáticas são as mesmas se você utiliza o RSA para administração de chaves ou assinaturas digitais: existe uma chave pública e privada, e a segurança do sistema está baseado na dificuldade de fatorar números grandes. Ambos PGP e PEM utilizam o RSA para assinaturas digitais.

DSA

O DSA é um algoritmo de assinatura digital (Digital Signature Algorithm). (Também existe um Padrão de Assinatura Digital (Digital Signature Standard), ou DSS. O padrão implementa o algoritmo - realmente é só uma diferença na terminologia.) é um algoritmo de chave pública, mas só pode ser usado para assinaturas digitais.

O DSA foi inventado no NSA e é patenteado pelo governo. (Se ou não esta patente é válida não está claro.) O NIST aprovou o DSA como um padrão de assinatura digital federal.
Quando o DSA foi primeiramente proposto, a Indústria RSA de Segurança de Dados e companhias que já tinham autorizado o algoritmo RSA lançaram uma campanha contra o DSA. Eles aludiram uma possível "porta de armadilha" que permitiria ao governo quebrar o DSA. Eles se queixaram da velocidade do algoritmo, o fato que não pode ser usado para cifragem e o tamanho da chave.

O único assunto real de segurança é o tamanho da chave. Quando o padrão foi primeiramente proposto, o tamanho da chave era de 512 bits - muito pequeno. O padrão final permite chaves de até 1024 bits - mais que suficiente para necessidades de segurança de qualquer um.

O DSA obtém sua segurança do problema de logaritmo discreto. As matemáticas são muito diferentes do RSA, mas a segurança é semelhante para chaves semelhantes em tamanho. Embora exceções são teoricamente possíveis, é provável que qualquer grande avanço na quebra do RSA também implique um avanço semelhante na quebra do DSA, e vice-versa. Não existe nenhuma vantagem na segurança usando um algoritmo em cima do outro.

Ocasionalmente comparações entre a velocidade e eficiência entre os dois algoritmos são publicadas. Não surpreendentemente, os publicados pela Indústria RSA de Segurança de Dados mostram que o RSA é melhor que o DSA e os publicados pelo NIST mostram que o DSA é melhor que o RSA. Com o RSA, leva-se mais tempo para assinar uma mensagem que verificar uma assinatura. Com o DSA, ambas operações de assinatura e de verificação levam a mesma quantia de tempo. Em realidade, estas diferenças são secundárias; com a finalidade da segurança de um programa de correio eletrônico elas são desprezíveis.

Atualmente nenhum programa de segurança de correio eletrônico utiliza o DSA, mas existem conversas dele ser utilizado para assinaturas digitais em uma versão de PEM (a mesma versão que usa o ElGamal em vez do RSA para administração de chave). Novamente, não se preocupe se você vê-lo; ele é da mesma forma tão seguro quanto o RSA.

Escolhendo um Tamanho para a Chave

Esta é uma decisão importante. As tabelas nos dizem que chaves de 1024 bits são requeridas para qualquer segurança a longo prazo.

Como isto Realmente Funciona

O mundo real é, como sempre, mais complicado. A complicação é análoga ao problema de usar criptografia de chave pública para cifrar a mensagem: Algoritmos de chave pública são incômodos e complicados e demora muito para assinar uma mensagem inteira. A maneira que programas de segurança de correio eletrônico utilizam assinaturas digitais é pegando uma impressão digital (fingerprint) do documento e assinando aquela impressão digital (fingerprint).
Esta função de fingerprinting é chamada uma função de hash de uma só via. (Veja a próxima seção, "Funções de Hash de uma só Via, para detalhes.) Uma função de hash de uma só via pega um tamanho arbitrário da mensagem e produz uma impressão digital de tamanho fixo daquela mensagem. A impressão digital é uma função matemática única da mensagem e é grande o suficiente de forma que a chance de duas mensagens diferentes terem o mesmo valor de hash é astronomicamente pequeno.

Assim, o programa de segurança de correio eletrônico utiliza uma função de hash de uma só via para gerar um valor de hash do documento a ser assinado e então utiliza um algoritmo de assinatura digital para criptografia de chave pública para assinar o valor de hash.

Um exemplo poderia tornar as coisas claras. Alice está usando PEM:MD5 como uma função de hash de uma só via e o RSA para assinaturas digitais. (Você irá aprender um pouco como o MD5 trabalha.) Ela quer enviar uma mensagem assinada para Bob. Estes são os passos que ela tem que seguir:

1. Alice escreve a mensagem.
2. A Alice gera um hash de uma só via da mensagem e usa uma função de hash de uma só via, como o MD5.
3. Alice assina o valor de hash com um algoritmo de assinatura digital de chave pública, como o RSA e a chave privada dela.
4. Alice concatena a mensagem e a assinatura para adquirir uma nova, assinada, mensagem.
5. Alice envia por e-mail esta mensagem assinada para Bob.

No lado de Bob, ele pode ler a mensagem sem fazer qualquer esforço. Mas ele quer verificar a assinatura de Alice. Estes são os passos ele tem que seguir (veja a Figura 2):

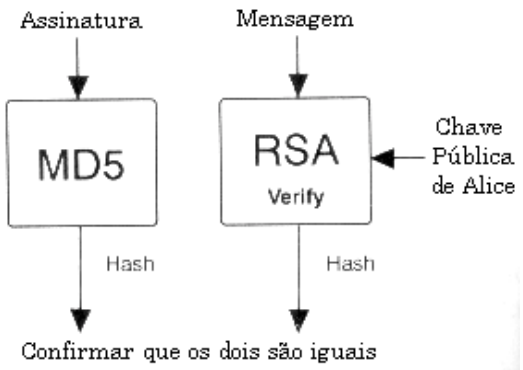

Figura 2. Verificando a assinatura de um mail eletrônico.

1. Bob separa a mensagem da assinatura.
2. Usando uma função de hash de uma só via, Bob computa o valor de hash da mensagem.
3. Bob adquire a chave pública de Alice.
4. Usando um algoritmo de assinatura digital de chave pública e a chave pública de Alice, Bob decifra a assinatura de Alice.
5. Bob compara a assinatura decifrada de Alice com o valor de hash da mensagem.
6. Se eles são o mesmo, Bob verificou a assinatura de Alice e aceitou a mensagem como genuína. Se eles são diferentes, Bob rejeita a assinatura.

Eve não pode forjar a assinatura de Alice. Eve não conhece a chave privada de Alice, assim ela não pode assinar "Alice" em uma mensagem. Eve não pode pegar uma assinatura de uma mensagem válida assinada por Alice e atachá-la a uma nova mensagem; quando Bob tenta verificar a assinatura ele perceberá que os dois valores de hash são diferentes e assinatura não é válida.

Isto poderia parecer um procedimento complicado, mas lembre-se de que programas de segurança de e-mail cuidam dos detalhes. Tudo que a Alice tem que fazer é indicar que ela deseja enviar uma mensagem assinada e o programa cuida do resto. Ele gera o valor de hash de uma só via, assina o hash, concatena a mensagem e o hash junto e envia a coisa inteira para Bob. No lado do receptor, o programa de Bob faz a operação de verificação facilmente.

Funções de Hash de uma só Via

Uma função de hash de uma só via converte uma mensagem de tamanho arbitrário em um hash de tamanho fixo. Este é um dos outros truques de criptografia. Como um algoritmo de cifragem, uma função de hash de uma só via converte uma mensagem de texto plano em palavras sem nexo. Porém, distinto de um algoritmo de cifragem, não existe nenhuma maneira de voltar para trás com uma função de hash de uma só via. Com a chave correta, a pessoa pode decifrar um texto cifrado codificado com um algoritmo de cifragem; é impossível inverter uma função de hash de uma só via para adquirir a entrada original do valor da saída.
Esta é uma diferença importante: Um algoritmo de cifragem não destrói nenhuma informação. Para qualquer determinado texto cifrado (e uma chave), existe somente um texto plano correto que pode ter resultado aquele texto cifrado. Uma função de hash de uma só via destrói a informação. Para qualquer saída resultante de uma função de hash de uma só via, várias mensagens podem ter produzido aquela saída.
Outra diferença entre algoritmos de cifragem e funções de hash de uma só via é que as funções de hash de uma só via não têm uma chave (veja a Figura 3). Nenhum segredo é envolvido na função de hash de uma só via; a segurança está na falta da habilidade de ir para outro modo. Esta propriedade faz disso uma maneira útil para identificar uma mensagem.
Pense em uma função de hash de uma só via como uma impressão digital. Da mesma maneira que uma impressão digital identifica exclusivamente um indivíduo, uma função de hash de uma só via identifica exclusivamente uma mensagem de tamanho arbitrário. Pelo menos, essa é a idéia. Tecnicamente isso é uma mentira.

Figura 3. Função de hash de uma só via.

Valores de hash de uma só via são normalmente pequenos: 16 ou 20 bytes. As mensagens podem ser grandes, muito grandes. Isso é um meio simples para mostrar que um número infinito de mensagens diferentes irá resultar um hash para o mesmo valor de hash de uma só via. Mas lembre-se, as chances de que duas mensagens quaisquer resultem um hash do mesmo valor é quase impossível e pode ser considerado desprezível.
Mostrando que existe um número arbitrariamente grande em uma coisa, mas achando duas mensagens que resultam em um hash para o mesmo valor é outro. Isto é onde os espertos matemáticos realmente entram. São projetadas funções de hash de uma só via de forma que seja praticamente impossível criar uma mensagem que resulte um hash de um valor particular, ou criar duas mensagens diferentes que resultem um hash de um mesmo valor.

Tabela 1: Tempo Necessário para Montar um Ataque de Força Bruta de $1 Milhão em Hardware para Achar uma Mensagem que Gere um Hash de um Valor Particular

	Tamanho do Hash em Bits	
Ano	64	128
1995	38 dias	10^{18} anos
2000	4 dias	10^{17} anos
2005	9 horas	10^{16} anos
2010	1 hora	10^{15} anos
2015	5.5 minutos	10^{14} anos
2020	31 segundos	10^{13} anos
2025	3.1 segundos	10^{12} anos
2030	.3 segundos	10^{11} anos

Tabela2: Tempo Necessário para Montar um Ataque de Força Bruta de $1 Milhão em Hardware para Achar uma Mensagem que Gere o Hash de um Mesmo Valor

	Tamanho do Hash em Bits			
Ano	64	128	160	256
1995	19 dias	38 dias	7,000 anos	10^{18} anos
2000	2 dias	4 dias	700 anos	10^{17} anos
2005	4.7 horas	9 horas	70 anos	10^{16} anos
2010	28 min	1 hora	7 anos	10^{15} anos
2015	2.8 min	5.5 min	251 dias	10^{14} anos
2020	17 seg	31 seg	25 dias	10^{13} anos
2025	1.7 seg	3 seg	2.5 dias	10^{12} anos
2030	.2 seg	.3 seg	6 horas	10^{11} anos

Isto é bom. Se Eve pudesse criar duas mensagens que resultem um hash de um mesmo valor, ela poderia criar uma mensagem inócua para a Alice assinar e uma mensagem incriminando que declara que Alice deve a Eve $1 milhão. Alice assinaria a mensagem inócua. Então Eve poderia levar a assinatura e poderia juntar a mensagem incriminada. Agora, desde que ambas as mensagens resultem um hash de um mesmo valor, Eve tem a assinatura válida de Alice na mensagem incriminada.

Eve sempre poderia tentar um ataque de força bruta contra uma função de hash de uma só via. Isso quer dizer, ela poderia gerar um hash da mensagem após mensagem, procurando por uma mensagem que resulte um hash de um valor particular, ou duas que resultem um hash de um mesmo valor. A taxa de sucesso deste ataque depende do tamanho do valor de hash.

As Tabelas 1 e 2 mostram ataques através de hardware especializados; ataques pelos computadores mais rápidos são 1.000 vezes mais lentos ou 1.000 vezes mais caros. Como o número de mensagens que têm que ser testadas excede de longe a quantia de tempo necessário para testar qualquer mensagem única, estes números podem ser examinados independente da função de hash de uma só via. Todas as tabelas incluem estimativas para 1995 e para o futuro e todas assumem que um hacker pode gastar $1 milhão.

Estimativas futuras assumem que o poder computacional, de hardware e de software, aumenta em um fator de 10 a cada cinco anos. Pode-se adquirir qualquer um deles dez vezes mais rápidos ou dez vezes mais baratos.
É muito mais fácil achar duas mensagens que resultem um hash de um mesmo valor randômico que achar uma mensagem que resulte um hash de um valor determinado. Isto é chamado o "paradoxo do aniversário".

Quantas pessoas têm que estar no mesmo quarto antes que se tenha 50 porcento de chance de que uma delas tenha sua mesma data de aniversário? A resposta é 183. Certo, quantas pessoas têm que estar no mesmo quarto antes que se tenha 50 porcento de chance de que duas delas tenham a mesma data de aniversário? A resposta é surpreendentemente baixa: 23. Assim, quantos valores randômicos de hash de 64-bit você tem que gerar para achar um que resulte um hash para um valor particular? 2^64 (10^19). Quantos valores randômicos de hash de 64-bit você tem que gerar para achar dois que resultem um hash para o mesmo valor? 2^32 (4 bilhões).
O uso de no mínimo um valor de hash de 128-bit é essencial para segurança e um valor de hash de 160-bit é melhor. Qualquer coisa menos que isso estará colocando-o em problemas.
Claro que, esta análise global assume que não existem alguma forma de ataque melhor contra a função de hash de uma só via. Como a maioria das coisas de criptografia, ninguém pode provar que uma função de hash de uma só via é segura contra tudo, inclusive ao ataque de força bruta. Tudo que nós podemos fazer é inventar funções, sujeitá-las a escrutínios públicos e a uma grande revisão e esperar pelo melhor.

MD5

O MD5 é uma função de hash de uma só via inventada por Ron Rivest do MIT que também trabalha para a Indústria RSA de Segurança de Dados. O MD significa "Message Digest" e o algoritmo produz um valor de hash de 128-bit para um tamanho arbitrário da mensagem inserida.
O MD5 foi primeiramente proposto em 1991, depois de alguns ataques de criptoanálise descobertos contra a função de hash de uma só via utilizada no MD4 de Rivest. O algoritmo foi projetado para velocidade, simplicidade e segurança. É claro que os detalhes do algoritmo são públicos e foi analisado por uma variedade de criptógrafos.
Uma fraqueza foi descoberta em alguma parte do MD5, mas ela não afetou a segurança global do algoritmo. Porém, o fato que ele só produz um valor de hash de 128-bit é muito inquietante; É prefirido uma função de hash de uma só via que produza um valor mais longo.
O PGP e o PEM usam o MD5 como uma função de hash de uma só via.

SHA

O SHA é o Algoritmo de Hash Seguro, uma função de hash de uma só via inventado no NSA. Ele produz um valor de hash de 160-bit de um tamanho arbitrário da mensagem.
Os funcionamentos internos do SHA são bem parecidos aos do MD4, indicando que os criptógrafos do NSA levaram o algoritmo MD4 e melhoraram a sua segurança. De fato, a fraqueza na parte do algoritmo MD5 mencionada acima, descoberta depois que o SHA foi proposto, não funciona contra o SHA.
Atualmente, não existe alguma forma conhecida de ataque criptoanalítico contra o SHA com exceção do ataque de força bruta. E seu valor de 160-bit faz do ataque de força bruta ineficiente. É claro que não existe alguma prova que alguém não possa entender como quebrar o SHA amanhã. Atualmente nem o PGP, nem o PEM utilizam o SHA, mas espera-se que ambos ofereçam isto, pelo menos como uma opção, antes que se espere muito tempo.

MD2 e MD4

O MD4 é o precursor do MD5 e também foi inventado por Ron Rivest. Depois que algumas fraquezas de segurança foram descobertas no MD4, Rivest escreveu o MD5. O MD4 foi parte das especificações originais do PEM, mas já não é mais usado.
O MD2 é uma função de hash de uma só via simplificada e produz um hash de 128-bit. É uma opção no PEM; Não é recomendado usá-lo porque geralmente acredita-se que não é muito seguro.

Outras Funções de Hash de uma Só Via

Existem outras funções de hash de uma só via na literatura. Snefru e N-Hash foram quebrados; fique longe de qualquer programa que os utilize. Haval não foi, mas não é muito utilizado. RIPE-MD foi desenvolvido para a Comunidade européia. E existe uma pilha inteira de funções de hash de uma só via baseado em algoritmos de bloco como o DES e o IDEA, alguns seguros e alguns não muito seguros.

fonte: http://www.cert-rs.tche.br/docs_html/autentic.html