UCSal – Universidade Católica de Salvador
Redes – Prof. Marco Câmara

Criptografia

E

Compressão de Dados

Nome: Igor Harã Serafim da Silva
Criptografia

Chave simétrica

Os **algoritmos de chave única** são também conhecidos como **algoritmos de chave simétrica** e, caracterizam-se por utilizar a mesma chave tanto para a cifragem como para a decifragem dos dados, esse método funciona em aplicações limitadas, como as militares, onde o emissor e o receptor podem se preparar antecipadamente para trocar a chave.

Infelizmente, genericamente esse método não funciona muito bem, pois trocar chaves secretas com todas as pessoas a quem você queira enviar uma mensagem é praticamente impossível. Para ilustrar isso, considere o que você teria de fazer se tivesse de enviar um memorando confidencial para acionistas de uma empresa. Primeiro, você teria de entrar em contato com cada acionista individualmente para que pudesse fazer a troca das chaves secretas. Isso poderia ser feito ao telefone, mas se as mensagens fossem extremamente confidenciais, talvez fosse melhor você trocar chaves pessoalmente. Lembre-se de que você precisaria fazer isso para todas as pessoas; cada uma teria uma chave secreta separada. Para aumentar a complexidade desse sistema, você também deverá se lembrar de qual chave serve para cada cliente. Se você as misturar, os clientes não serão capazes de ler as suas mensagens. Obviamente, esse tipo de sistema não é viável para transações comerciais comuns.

Algoritmo Simétrico	Bits	Descrição
DES	56	O Data Encryption Standard (DES) é o algoritmo simétrico mais disseminado no mundo. Foi criado pela IBM em 1977 e, apesar de permitir cerca de 72 quadrilhões de combinações (2^{56}), seu tamanho de chave (56 bits) é considerado pequeno, tendo sido quebrado por "força bruta" em 1997 em um desafio lançado na Internet. O NIST (National Institute of Standards and Technology), que lançou o desafio mencionado, recertificou o DES pela última vez em 1993 e desde então está recomendando o 3DES. O NIST está também propondo um substituto ao DES que deve aceitar chaves de 128, 192 e 256 bits, operar com blocos de 128 bits, ser eficiente, flexível e estar livre de "royalties". O novo padrão, denominado AES (Advanced Encryption Standard), está sendo estudado desde 1997 a partir de vários algoritmos apresentados pela comunidade. Os finalistas são: Serpent, Mars, RC6, Twofish e Rinjdael, e o resultado deverá ser divulgado no final de 2000.
Triple DES	112 ou 168	O 3DES é uma simples variação do DES, utilizando-o em três ciframentos suscessivos, podendo empregar um versão com duas ou com três chaves diferentes. É seguro, porém muito lento para ser um algoritmo padrão.
IDEA	128	O International Data Encryption Algorithm foi criado em 1991 por James Massey e Xuejia Lai e possui patente da suíça ASCOM Systec. O algoritmo é estruturado seguindo as mesmas linhas gerais do DES. Mas na maioria dos microprocessadores, uma implementação por *software* do IDEA é mais rápida do que uma implementação por *software* do DES. O IDEA é utilizado principalmente no mercado financeiro e no PGP, o programa para criptografia de e-mail pessoal mais disseminado no mundo.
Blowfish	32 a 448	Algoritmo desenvolvido por Bruce Schneier, que oferece a escolhe entre maior segurança ou desempenho através de chaves de tamanho variável. O autor aperfeiçoou-o no Twofish, concorrente ao AES.

RC2	8 a 1024	Projetado por Ron Rivest (o R da empresa RSA Data Security Inc.) e utilizado no protocolo S/MIME, voltado para criptografia de e-mail corporativo. Também possui chave de tamanho variável. Rivest também é o autor do RC4, RC5 e RC6, este último concorrente ao AES.

Chave assimétrica

Os **algoritmos de chave pública e privada**, também chamados de **algoritmos de chave assimétrica**, utilizam duas chaves: uma pública que pode ser divulgada e outra secreta conhecida somente por pessoas autorizadas. Em um sistema de chave pública, cada pessoa tem duas chaves: uma chave pública e uma chave privada. As mensagens criptografadas com uma das chaves do par só podem se decriptografadas com a outra chave correspondente; portanto, qualquer mensagem criptografada com a chave privada só pode ser decriptografada com a chave pública e vice-versa. Como o nome sugere, normalmente a chave pública é mantida universalmente disponível. A outra chave, a chave privada, é mantida em segredo.

Vamos ver com a juda de um exemplo o funcionamento da chave pública. Se Frank e Andrea quiserem se comunicar secretamente usando a criptografia com chave pública, eles terão de fazer o seguinte:

1. Frank escreve uma mensagem e a criptografa utilizando a chave pública de Andrea. A chave está disponível para qualquer pessoa.
2. Frank envia a mensagem para Andrea através da Internet.
3. Andréa recebe a mensagem e a decriptografa utilizando sua chave privada.
4. Andréa lê a mensagem. Se quiser responder, ela deverá fazer o mesmo, a diferença é que dessa vez a chave de Frank será utilizada.

Como apenas Andrea tem acesso à sua própria chave privada (presupondo-se que a segurança do sistema seja eficiente), somente ela poderá ler a mensagem, pois existe confidencialidade. A grande vantagem deste sistema é que não só Frank pode enviar mensagens secretas para Andrea, todo mundo pode. Tudo o que o emissor precisa é da chave pública de Andrea.

Lembre-se que o sigilo da chave privada é importantíssimo. O criptossistema inteiro se baseia no fato de que a chave privada é realmente privada. Se um invasor conseguir roubar a sua chave privada, tudo estará perdido. Não importa qual seja a eficiência do algoritmo de criptografia - o intruso vencerá e poderá ler e criar mensagens utilizando seu nome.

Algoritmo	Descrição
RSA	O RSA é um algoritmo assimétrico que possui este nome devido a seus inventores: Ron Rivest, Adi Shamir e Len Adleman, que o criaram em 1977 no MIT. É, atualmente, o algoritmo de chave pública mais amplamente utilizado, além de ser uma das mais poderosas formas de criptografia de chave pública conhecidas até o momento. O RSA utiliza números primos. A premissa por trás do RSA é que é fácil multiplicar dois números primos para obter um terceiro número, mas muito difícil recuperar os dois primos a partir daquele terceiro número. Isto é conhecido como *fatoração*. Por exemplo, os fatores primos de 3.337 são 47 e 71. Gerar a chave pública envolve multiplicar dois primos grandes; qualquer um pode fazer isto. Derivar a chave privada a partir da chave pública envolve fatorar um grande número. Se o número for grande o suficiente e bem escolhido, então ninguém pode fazer isto em uma quantidade de tempo razoável. Assim, a segurança do RSA baseia-se na dificuldade de fatoração de números grandes. Deste modo, a fatoração representa um limite superior do tempo necessário para quebrar o algoritmo. Uma chave RSA de 512 bits foi quebrada em 1999 pelo Instituto Nacional de Pesquisa da Holanda, com o apoio de cientistas de mais 6 países. Levou cerca de 7 meses e foram utilizadas 300 estações de trabalho para a quebra. Um fato

	preocupante: cerca de 95% dos sites de comércio eletrônico utilzam chaves RSA de 512 bits.
ElGamal	O ElGamal é outro algoritmo de chave pública utilizado para gerenciamento de chaves. Sua matemática difere da utilizada no RSA, mas também é um sistema comutativo. O algoritmo envolve a manipulação matemática de grandes quantidades numéricas. Sua segurança advém de algo denominado problema do logaritmo discreto. Assim, o ElGamal obtém sua segurança da dificuldade de se calcular logaritmos discretos em um corpo finito, o que lembra bastante o problema da fatoração.
Diffie-Hellman	Também baseado no problema do logaritmo discreto, e o criptosistema de chave pública mais antigo ainda em uso. O conceito de chave pública aliás foi introduzido pelos autores deste criptosistema em 1976. Contudo, ele não permite nem ciframento nem assinatura digital. O sistema foi projetado para permitir a dois indivíduos entrarem em um acordo ao compartilharem um segredo tal como uma chave, muito embora eles somente troquem mensagens em público.
Curvas Elípticas	Em 1985, Neal Koblitz e V. S. Miller propuseram de forma independente a utilização de curvas elípticas para sistemas criptográficos de chave pública. Eles não chegaram a inventar um novo algoritmo criptográfico com curvas elípticas sobre corpos finitos, mas implementaram algoritmos de chave pública já existentes, como o algoritmo de Diffie e Hellman, usando curvas elípticas. Assim, os sistemas criptográficos de curvas elípticas consistem em modificações de outros sistemas (o ElGamal, por exemplo), que passam a trabalhar no domínio das curvas elípticas, em vez de trabalharem no domínio dos corpos finitos. Eles possuem o potencial de proverem sistemas criptográficos de chave pública mais seguros, com chaves de menor tamanho. Muitos algoritmos de chave pública, como o Diffie - Hellman, o ElGamal e o Schnorr podem ser implementados em curvas elípticas sobre corpos finitos. Assim, fica resolvido um dos maiores problemas dos algoritmos de chave pública: o grande tamanho de suas chaves. Porém, os algoritmos de curvas elípticas atuais, embora possuam o potencial de serem rápidos, são em geral mais demorados do que o RSA.

Assinatura Digital:

Ter duas chaves separadas proporciona outro benefício: a **assinatura digital**. Imagine usar o sistema de forma invertida. Em vez de Frank criptografar a mensagem com a chave pública de Andréa, ele utiliza sua própria chave privada. Você deve observar que agora todo mundo pode ler a mensagem, ele deixou de ser privada. Isso é verdade, mas também é verdade que apenas Frank poderia ter escrito a mensagem. Pois ele é a única pessoa capaz de criar mensagens que possam ser lidas com sua chave pública, pressupondo-se que Frank não tenha compartilhado sua chave privada com ninguém mais e que ela seja realmente secreta.

Vejamos um exemplo. Frank quer enviar uma mensagem para todos os seus contatos informando-os que mudou de emprego. Na verdade, ele não se importa com quem lerá a mensagem, mas quer ter certeza de garantir a seus contatos que a mensagem realmente é dele, e não de outra pessoa.

1. Frank escreve a mensagem e a criptografa utilizando sua chave privada.
2. Frank envia a mensagem a seus contatos através da Internet.
3. Os contatos recebem a mensagem e a decriptografam utilizando a chave pública de Frank.

O fato de a chave pública de Frank ter decriptografado a mensagem garante aos contatos que a mensagem realmente é de Frank. Qualquer mensagem decriptografada com a chave pública de Frank só poderia ter sido criada com sua chave privada.

Isso é muito importante. Na criptografia com chave pública, cada par de chaves é único. Só existe apenas uma chave pública para cada chave privada e vice-versa. Se isso não fosse verdade, a assinatura digital não seria possível; um impostor poderia utilizar outra chave privada para criar uma mensagem que pudesse ser lida pela chave pública fornecida.

Podemos observar que a assinatura digital assegura aos contados que a mensagem não foi alterada (integridade) e que vei de Frank (autenticidade). Além disso, Frank é o único com acesso a sua chave privada.

Agora combinando so dois métodos, Frank pode enviar uma mensagem privada e assinada para Andrea, vejamos :

1. Frank escreve a mensagem e a criptografa utilizando sua chave privada (assinatura da mensagem).
2. Em seguida, ele criptografa a mensagem com a chave pública de Andrea (tornando-a privada)
3. Frank envia a mensagem duplamente criptografada para Andrea através da Internet.
4. Andrea recebe a mensagem.
5. Ela decriptografa a mensagem duas vezes. Primeiro, ela utiliza sua chave privada e, depois, a chave pública de Frank.
6. Agora Andrea pode ler a mensagem e tem certeza que ela é secreta e vei de Frank. Ela tem certeza de que a mensagem não foi modificada; para alterá-la, o invasor teria de acessar a chave privada de Frank.

Certificados :

Como acabamos de ver, a criptografia com chave pública pode ser usada para proporcionar confidencialidade, integridade não-repudiação. A assinatura digital exige que o verificador esteja certo de que tem uma chave pública pertencente à pessoa que assinou a mensagem. A confiança do verificador na assinatura deverá ser igual à sua confiança no proprietário da chave pública. Por exemplo, se um violador quiser forjar documentos eletrônicos, uma estratégia seria criar um par de chave pública/chave privada e divulgar a chave pública com o nome de outra pessoa. Quaisquer documentos assinados com a chave privada do violador serão verificados com a chave pública correspondente - uma chave pública que tenha sido anunciada como pertencente a outra pessoa.

Existem diversas estratégias para solucionar esse problema. Uma delas é trocar chaves públicas através de um meio diret, como uma reunião. Infelizmente, esse método não funciona bem quando temos muitas pessoas envolvidas. Uma opção melhor seria a utilização de **certificados e de autoridades de certificação**.

Um **certificado** é um documento digital contendo informações de identificação e uma chave pública. Em geral, os certificados têm um formato comum, normalmente baseados no padrão ITU-T X.509. Mas ainda não podemos ter certeza de que o certificado é genuíno e não é falso. Uma forma de descobrir isso é utilizar **autoridades de certificação ou CAs**.

Uma **autoridade de certificação** assina certificados de chave pública digitalmente. Ao assinar um certificado, a CA garante sua validade. No entanto um problema persiste: como a chave pública da CA é distribuída? Também existem muitas estratégias para esse problema. Em uma delas, se a CA for muito conhecida, como é o caso do serviço postal americano, ele poderá divulgar amplamente sua chave pública. Outro método seria que a CA tivesse seu próprio certificado assinado por outra CA, também conhecido pelo destinatário. Essa idéia de encadeamento de certificação pode avançar ainda mais, com várias CAs organizadas em uma hierarquia onde cada CA subordinada valida sua assinatura com a assinatura de uma CA mais alta na hierarquia. Obviamente, as CAs de nível mais alto deverão reverter para o método de divulgação direta.

Quem oferecerá esses serviços? Há muito para se discutir a respeito de quem deverá manter CAs na Internet. Muitas organizações, incluindo instituições financeiras, fornecedores de aplicações e até mesmo o serviço postal americano expressaram interesse em oferecer serviços de certificação. Em 1995, com a ajuda de muitos parceiros, a RSA Data Security criou uma empresa, a VeriSign, inteiramente dedicada ao fornecimento de serviços de certificação para usuários de chaves públicas.

Em um fututo próximo, provavelmente existirão CAs na Internet, cada uma com exigências variáveis para comprovar identidade. Algumas podem exigir que você venha pessoalmente com uma cópia da sua certidão de nascimento, enquanto outras podem assinar certificados com a garantia de sua palavra. A questão aqui é que o fato de um documento ter sido assinado por uma CA não o torna necessariamente válido.

Questões práticas no uso da criptografia :

Vejamos algumas questões de gerenciamento relacionadas ao uso da criptografia:

- **Tamanho da chave** - A criptoanálise, a ciência da decodificação de cifras, se baseia no fatoramento de números grandes. Conseqüentemente, quanto maior a chave, mais difícil decifrá-la. No entanto, as empresas que conduzem negócios nos Estados Unidos e no exterior são limitadas pelas leis americanas à utilização de chaves de 40 bits em suas aplicações. Quarenta bits já são o bastante? Depende do que está sendo criptografado; as chaves RC4 de 40 bits podem ser decifradas com uma certa facilidade. Portanto, você não vai quere usá-las para criptografar informações durante um período muito longo. Apesar de exigir um esforço considerável, a decifração de uma mensagem criptografada com uma chave RC4 de 40 bits é possivel. As chaves de 40 bits podem ser seguras para criptografar ordens de compra ou mensagens de correio eletrônico (que só precisam permanecer secretas até que o destinatário as receba), mas talvez não sejam suficientes para proteger segredos vitais para as empresas.
- **Revogação de certificado** - O que acontece quando uma chave privada é comprometida, ou uma chave pública passa a ser inválida? Nesse caso, os certificados deixam de ser confiáveis, pois as informações que eles estão certificando não são mais verdadeiras. Como podemos impedir que os certificados sejam usados? Muitas autoridades de certificação divulgam periodicamente listas de certificados que não devem mais ser considerados válidos, as CRLs (certificate revocation lists). Os certificados dessas listas podem ter expirados, ou o par de chaves associados ao certificado pode ter sido decifrado. As CRLs são muito eficientes para verificar a precisão de um certificado em um determinado momento. No entanto, as CRLs não lhe dão qualquer garantia de que um certificado não foi revogado desde sua divulgação. Por essa razão, muitas organizações estão procurando outras soluções para lidar com a verificação de certificados.
- **Estrutura de CA** - Já falamos sobre CAs e como podem ser "encadeadas" em hierarquias de certificação. Portanto, as grandes organizações podem optar por ter dentro delas várias CAs (uma para o departamento de pesquisa, outra para o departamento de bolsas e assim por diante) e ter uma única CA de nível mais alto para certificar as CAs dos departamentos. No entanto, manter vários níveis de CAs pode ser complicado. Além disso, ter uma única CA na empresa cria um "único ponto de falha", pois o comprometimento do par de chaves da CA corporativa pode resultar na perda das chaves de todos os funcionários da empresa. Outras empresas optam por uma estrutura plana de CAs, na qual cada CA departamental tem muitas outras CAs parceiras, que correspondem ao outros departamentos da empresa. É melhor ter uma hierarquia de CAs, na qual o comprometimento de uma única CA pode resultar no comprometimento de todos os certificados da empresa, ou é melhor ter uma estrutura plana, na qual várias CAs devem ser controladas e devem trocar mensagens umas com as outras?
- **Fazer ou não fazer caução** - Esse é um dos assuntos mais debatidos no que se refere à criptografia. Muitas empresas afirmam que, como a comunicação dos funcionários é propriedade da empresa, a organização deverá ter acesso às chaves dos funcionários, recuperar mensagens (quando houver desligamento de funcionários ou quando eles perderem suas chaves, por exemplo). Por outro lado, a maior parte dos defensores da privacidade é veemente contra essa idéia, citando a emenda federal que garante os direitos dos funcionários à privacidade.
- **O que fazer com todas essas informações** - O arquivamento de chaves e de dados criptografados (mensagens de correio eletrônico criptografadas, ordens de compra assinadas,...) é uma questão extremamente difícil. Muitas empresas têm de armazenar informações durante muitos tempo, devido a regulamentos ou outras restrições. Mas, com freqüência, armazenar mensagens ou ordens de compra pode envolver muito mais do que simples manutenção de informações. Considere, por exemplo, o armazenamento de ordens de compras assinadas.

Como as assinaturas só podem ser verificadas com a chave pública apropriada, todas as chaves públicas (e suas cadeias de certificação) devem ser guardadas para sempre. Mais uma vez, para grandes empresas, isso pode ser complicado.

PGP

PGP (abreviação de Pretty Good Privace - "Privacidade Boa o Suficiente") é um programa de criptografia de chave pública escrito originalmente por Phil Zimmermann em 1991. Com o passar de poucos anos, PGP recebeu a adesão de milhares de usuários sobre todo o planeta e se tornou um padrão de fato para criptografia de correio eletrônico sobre a Internet. Se voce não sabe o que o PGP pode fazer por você, por favor utilize um pouco do seu tempo para ler o artigo *abaixo* escrito pelo autor do software.

"Por que usar o PGP?

É pessoal. É particular. E não é da conta de mais ninguém a não ser você. Você pode estar planejando uma campanha política, discutindo seus impostos, ou tendo um caso ilícito. Ou você pode estar fazendo algo que você sente que não deveria ser ilegal, mas é. Qualquer que seja o caso, você não quer que seu correio eletrônico particular (E-mail) ou documentos confidenciais sejam lidos por mais ninguém. Não há nada errado em assegurar sua privacidade. Privacidade é algo tão natural e respeitável quanto a Constituição.

Talvez você pense que seu E-mail seja tão lícito que criptografia não seja justificável. Se você realmente é um cidadão que respeita as leis estritamente com nada a esconder, então por que você não envia sempre sua correspondência em cartões-postais? Por que não se submeter a testes de drogas sempre que requisitado? Por que exigir um mandado para revistas da polícia em sua casa? Você está tentando esconder algo? Você deve ser um subversivo ou um traficante de drogas se você esconde sua correspondência em envelopes. Ou talvez um maluco paranóico. Cidadãos que respeitam as leis têm alguma necessidade de criptografar seus E-mails?

E se todos acreditassem que cidadãos que respeitam as leis devessem usar cartões-postais para sua correspondência? Se alguma brava alma tentasse assegurar sua privacidade ao usar um envelope para sua correspondência, isso levantaria suspeitas. Talvez as autoridades abririam sua correspondência para ver o que ela estaria escondendo. Felizmente, nós não vivemos naquele tipo de mundo, porque todos protegem a maior parte de sua correspondência com envelopes. Então, ninguém levanta suspeitas ao assegurar sua privacidade com um envelope. Não há segurança em números. Analogamente, seria bom se todos rotineiramente usassem criptografia para todo seu E-mail, inocente ou não, de modo que ninguém levantaria suspeitas ao assegurar sua privacidade no E-mail com criptografia. Pense nisso como uma forma de solidariedade.

Hoje, se o Governo quer violar a privacidade de cidadãos comuns, ele tem que gastar uma certa quantidade de recursos e esforço para interceptar e abrir dissimuladamente correspondência em papel, e ouvir e possivelmente transcrever conversas faladas ao telefone. Este tipo de monitoramento trabalhoso e intensivo não é prático em grande escala. Isto é feito somente em casos importantes quando parece valer a pena.

Mais e mais das nossas comunicações particulares estão sendo roteadas por canais eletrônicos. Correio eletrônico está gradualmente substituindo a correspondência convencional em papel. Mensagens por E-mail são simplesmente fáceis demais de se interceptar e de se vasculhar em busca de palavras interessantes. Isto pode ser feito facilmente, rotineiramente, automaticamente, e indetectavelmente em grande escala. Cabogramas internacionais já são vasculhados deste modo em grande escala pela NSA.

Nós estamos nos dirigindo a um futuro no qual a nação será entrecortada por redes de dados de fibra ótica de alta capacidade, ligando todos os nossos cada vez mais onipresentes computadores. O E-mail será a norma para todos, não a novidade que é hoje. O Governo protejerá nosso E-mail com protocolos de criptografia projetados pelo Governo. Provavelmente a maioria das pessoas se sujeitará a isso. Mas talvez algumas pessoas preferirão suas próprias

medidas de proteção.

O Projeto de Lei 266 do Senado (dos EUA), um extenso projeto anti-crime de 1991, tinha uma perturbadora medida encravado nele. Se esta resolução sem restrições tivesse se tornado uma lei real, ela teria forçado fabricantes de equipamentos de comunicação segura a inserir alçapões nos seus produtos, de modo que o Governo pudesse ler as mensagens criptografadas de qualquer um. Eis seu conteúdo:

"É o julgamento do Congresso que fornecedores de serviços de comunicações eletrônicas e fabricantes de equipamentos de serviços de comunicações eletrônicas devem assegurar que sistemas de comunicações permitam ao Governo obter o conteúdo integral de voz, dados e outras comunicações quando apropriadamente autorizado pela lei." Esta medida foi derrotada após rigorosos protestos de civis libertários e de grupos industriais.

Em 1992, a proposta de grampo da Telefonia Digital do FBI foi apresentada ao Congresso. Ela requereria que todos os fabricantes de equipamentos de comunicações embutissem portas especiais de grampo remoto as quais capacitariam o FBI a grampear remotamente todas as formas de comunicação eletrônica a partir de escritórios do FBI. Apesar de nunca ter atraido qualquer apoio no Congresso por causa da oposição popular, ela foi reapresentada em 1994.

O mais alarmante de tudo é a nova iniciativa de política de criptografia da Casa Branca, em desenvolvimento na NSA desde o início da administração Bush, e revelada em 16 de abril de 1993. A peça central desta iniciativa é um dispositivo de criptografia construído pelo Governo, chamado de chip Limitador, contendo um novo algorítmo confidencial NSA. O Governo está encorajando a indústria privada a projetá-lo em todos seus produtos de comunicação segura, como telefones seguros, fax seguro, etc. A AT&T agora está colocando o Limitador nos seus produtos de voz seguros. A trama: No momento de sua fabricação, cada chip Limitador será carregado com sua própria chave única, e o Governo fica com uma cópia, a qual é arquivada. Nada com o que se preocupar, entretanto -- o Governo promete que eles usarão estas chaves para ler seu tráfego somente quando apropriadamente autorizados pela lei. É claro, para fazer o Limitador completamente efetivo, o próximo passo lógico seria tornar ilegais outras formas de criptografia.

Se a privacidade se tornar ilegal, somente criminosos terão privacidade. Agências de inteligência têm acesso a uma ótima tecnologia criptográfica. Assim como os grandes traficantes de armas e drogas. Assim como fornecedores de sistemas militares, empresas petrolíferas, e outros gigantes empresariais. Mas pessoas comuns e organizações políticas populares majoritariamente não têm tido acesso a tecnologia criptográfica de chave-pública de nível militar. Até agora.

O PGP dá o poder às pessoas para tomar sua privacidade em suas próprias mãos. Há uma crescente necessidade social para ele. É por isso que eu o escrevi."·(Translation from English to Brazilian Portuguese by Roberto Lopes)"

SSL

SSL significa **Secure Socket Layer** (servidor com encriptação de dados para utilização em transações via Web). É um padrão (protocolo) desenvolvido pela Netscape Communications para transferir informações de modo seguro na internet, desde que ambos, o servidor e o cliente, apoiem o protocolo.

O SSL providencia autenticação, confidencialidade e integridade dos dados, sendo planejado para autenticar o servidor e opcionalmente o cliente. Como este padrão é aberto, vários desenvolvedores podem aprimorá-lo, inclusive implementar com novas características e funções. O SSL permitirá que o cliente se conecte ao Web Site e, de forma transparente será criado um canal de comunicação seguro entre o Site e o Cliente. Uma vez que esta conexão é feita, informações, como o número de cartões de crédito, senhas de contas corrente, poderão

ser fornecidas sem que alguma outra pessoa possa interceptar os dados, ou seja, de uma maneira segura. Esta segurança é garantida pela encriptação, os usuários que interceptarem a mensagem no caminho, ficam impedidos de acessar o conteúdo da mensagem, já que não conseguirão entender o que está sendo transmitido.

Todo site que quiser usar SSL precisará de um certificado de autenticação "assinado" por uma entidade certificadora, como a Verisign, por exemplo. Se não for de desejo ter um certificado próprio, é possível usar o certificado da RapidSite, isso vai fazer com que as páginas façam referência a http://www.rapidsite.net ao invés de fazer referência ao nome do domínio que está sendo utilizado.

O SSL utiliza como protocolo de transporte o TCP, que providencia uma transmissão e recepção confiável dos dados. Uma vez que o SSL reside no nível de socket, ele é independente das aplicações de mais alto nível, sendo assim considerado um protocolo de segurança independente do protocolo aplicacional. Como tal, o SSL pode providenciar serviços seguros para protocolo de alto nível, como por exemplo TELNET, FTP e HTTP.

Como funciona SSL?

Existem três componentes principais para um site Web seguro:

- 1. Servidor: o melhor lugar da Internet para armazenar o Web Site.

- 2. Software Seguro: este é o software instalado no servidor, que faz todo o trabalho de criptografia.

- 3. Certificado de Assinatura: é como uma "assinatura digital".

Como funciona o servidor compartilhado SSL: existe uma cópia do software SSL rodando no servidor principal, é acrescentado o domínio no arquivo de configuração como um domínio adicional.

Preços do servidor compartilhado SSL: a Verisign cobra U$ 349 por esse serviço. É um pagamento único com uma taxa de renovação de U$ 95 por ano.

Como funciona o SSL para um servidor dedicado: o software SSL na verdade, funciona como um servidor Web que compartilha seu domínio. Isso requer que seja gerada uma "chave" para o servidor. Esta é uma chave de 512 bits, que é criptografada e assinada digitalmente pela certificadora.

HTTPS X SHTTP

Existem duas grandes abordagens para a solução do problema de seguranca no nível dos protocolos da camada de aplicação na arquitetura Internet: o HTTPS e o SHTTP.

HTTPS é a utilização do protocolo HTTP (HyperText Transfer Protocol) em conjunto com o protocolo SSL (secure Sockets Layer), que é um protocolo proposto por um grupo liderado pela Netscape Communications, pela Verisign e pela Sun desenvolvido e especificado para prover uma camada de segurança entre a camada de transporte (TCP) e os protocolos de aplicação tais como HTTP, TELNET, FTP, NNTP, SMTP, etc.

Este protocolo provê encriptação de dados, autenticação de servidor, integridade de mensagem e, opcionalmente, autenticação de cliente para uma conexão TCP/IP.

SHTTP (secure HTTP) é uma extensão do protocolo http proposta pelo EIT no começo de 1994 que provê transações seguras pela incorporação de criptografia, mecanismos de autenticação no protocolo HTTP permitindo transações seguras fim-a-fim entre cliente e servidor WWW.

SSL e SHTTP tem diferentes motivações: as camadas de segurança SSL ficam sob os protocolos de aplicação, como HTTP, NNTP e TELNET, enquanto que HTTPS adiciona segurança baseada nas mensagens especificamente do protocolo HTTP no nível da aplicação. Estas duas aplicações, longe de serem mutuamente exclusivas podem coexistir perfeitamente de forma complementar com o protocolo HTTPS atuando sobre a camada SSL.

Uma das vantagens do protocolo SSL é o fato de ser um protoclo independente da aplicação. Um protocolo de alto nível pode ser suportado sobre o protocolo SSL de forma transparente. Os principais objetivos do protocolo SSL, em ordem de prioridade são:

- **Segurança criptográfica:** o SSL deve ser usado para estabelecer uma conexão segura entre um cliente e um servidor.

- **Interoperabilidade:** programadores independentes devem ser capazes de desenvolver aplicações utilizando SSL que possam trocar parâmetros entre si sem conhecerem os códigos umada outra, com sucesso.
- **Extensibilidade:** SSL busca prover um framework no qual novas chaves públicas e métodos de encriptação possam ser incorporados, sem a necessidade de desenvolver novos protocolos.
- **Relativa eficiência:** operadores de criptografia costumam ter processamentos pesados, particularmente em operações de chaves públicas de encriptação. Por esta razão, o protocolo SSL incorpora um sistema de caching que reduz o número de conexões necessárias e procura reduzir ao mínimo as atividades de rede.

O SSL consiste em dois protocolos:

- **SSL Handshake Protocol:** é usado para negociar os parâmetros de segurança na conexão SSL.
- **SSL Record Protocol:** especifica o encapsulamento de todas as transmissões e recepções de dados. Faz parte das negociações entre o cliente e o servidor, o emissor pode identificar qual o algoritmo de cifragem suportado.

Endereço das páginas SSL:

As páginas SSL têm um endereço diferente das páginas normais. Este endereço deve ser utilizado para que a página possa ser acessada corretamente. O endereço é iniciado com https:// ao invés de http:// como é o endereço normal.

Em, geral, as páginas SSL são pequenas porções de um determinado site e por isso o endereço específico não é divulgado. Ao invés disso, o endereço é utilizado como um link do site normal para o SSL. Para fazer isso, basta incluir o https:// no endereço do link. Exemplo: https://banking1.caixa.gov.br.

Um pouco mais:

- As páginas SSL apesar de serem HTML da mesma forma que as normais, têm a forma de trabalho um pouco diferente, por causa da criptografia.
- SSL exige que as páginas sejam criptografadas no servidor e descriptografadas no cliente. Isso gera um processamento a mais para esses passos. Por isso, o SSL deve ser utilizado onde for realmente necessário no site. As páginas que não tiverem nenhuma informação confidencial e forem públicas não tem a necessidade de serem criptografadas.
- As páginas criptografadas não entram no cache dos browsers e devem ser carregadas novamente toda vez que o usuário faz o acesso. Isso é mais um motivo para usar o SSL somente quando necessário.
- Os recursos de SSL estão disponíveis nos browsers Netscape (3.0 ou superior) e o Internet Explorer (3.01 ou superior).

Compressão de Dados

Para que comprimir?

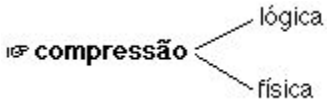
Ao contrário do que possa parecer, comprimir não é somente reduzir o tamanho de um arquivo: há várias outras aplicações que utilizam a compressão de dados. Obviamente, a **redução do espaço físico utilizado** é um conjunto de aplicações, mas há outro relativo à **agilização da transmissão de dados**.

A redução do espaço físico é mais comumente utilizada em bancos de dados que, incorporando a compressão no projeto de seus registros, permite um significativo ganho em termos de ocupação em disco e velocidade de acesso.

Como foi mencionado anteriormente, a utilização mais conhecida da compressão de dados é a redução do espaço ocupado por arquivos. De fato, esta é aplicação mais comum e mais difundida comercialmente. Afinal, dado um dispositivo restrito de armazenamento que é um disquete, e um grande depositório de arquivos que é um disco rígido, faz-se evidente a necessidade de muitas vezes realizar-se a redução do tamanho de arquivos para transportá-los ou simplesmente armazená-los.

A compressão também é utilizada para agilizar a transmissão de dados basicamente de duas formas: alterando a taxa de transmissão e permitindo o aumento no número de terminais de uma rede.

Tipos de Compressão



Compressão Lógica

A compressão lógica refere-se ao projeto de representação otimizada de dados. Um exemplo clássico é o projeto de um banco de dados utilizando seqüências de bits para a representação de campos de dados. No lugar de seqüências de caracteres ou inteiros, utiliza-se bits, reduzindo significativamente o espaço de utilização do banco de dados.

Este tipo de compressão é possível de ser efetivada em campos projetados para representar dados constantes, como datas, códigos e quaisquer outros campos formados por números. A característica lógica da compressão encontra-se no fato dos dados já serem comprimidos no momento do armazenamento, não ocorrendo sua transformação de dados estendidos para comprimidos.

Compressão Física

A compressão física é aquela realizada sobre dados existentes, a partir dos quais é verificada a repetição de caracteres para efetivar a redução do número de elementos de dados. Existem dois tipos de técnicas para sinalizar a ocorrência de caracteres repetidos:

1. um deles indica o caracter (ou conjunto de caracteres) repetido através da substituição por um caracter especial;
2. outras técnicas indicam a freqüência de repetição de caracteres e representam isto através de seqüências de bits.

Técnicas de compressão com e sem perda

Se a informação, após sua compressão, pode ser exatamente reconstruída a técnica de compressão é dita sem perdas. Esta técnica deve ser utilizada obrigatoriamente para comprimir programas e documentos legais ou médicos. As técnicas de compressão sem perda não são idéias novas, elas são muito utilizadas. Estas técnicas exploram apenas estatísticas de dados (redundância de dados) e a taxa de compressão é normalmente baixa. Um exemplo deste tipo de compressão é substituir caracteres de espaços ou zeros sucessivos por um flag especial e o número de ocorrências. Como exemplo de técnicas sem perda temos: codificação aritmética, codificação Huffman e codificação Run-length.

Técnicas de compressão com perdas são utilizadas para compressão de áudio, imagens e vídeos, onde erros e perdas são toleráveis. Estas técnicas são baseadas normalmente em estatísticas de dados e propriedades da percepção humana. Com ela, altas taxa de compressão podem ser obtidas.

Técnicas por Entropia, Codificação na Origem e Híbrida

A codificação por entropia (*Entropy encoding*) trata de cadeias de bits sem levar em conta seu significado [Tanembau, 97]. É uma técnica genérica, sem perda e totalmente reversível, que pode ser aplicada a todos os dados. Nesta apostila são apresentados alguns exemplos de técnicas por entropia, que são a codificação run-length e de Huffman.

Codificação na origem (*Source coding*) processa o dado original distinguindo o dado relevante e o irrelevante. Elas levam em consideração a semântica dos dados. Removendo os dados irrelevantes comprime o dado original. Como exemplo de técnicas de compressão da origem, nós temos: DPCM (*Differential pulse code modulation*), DCT (*discrete cosine transform*) e DWT (*Discrete wavelet transform*). Codificação híbrida é a combinação de técnicas de compressão sem perdas e técnicas de codificação na origem. Normalmente, várias destas duas técnicas são agrupadas para formar uma nova técnica de codificação híbrida. Como exemplo deste tipo de técnica de compressão podemos citar os padrões H.261, H.263, JPEG, MPEG vídeo e áudio.

Codificação Run-Length

Codificação run-length é uma codificação por entropia. Parte dos dados de imagem, áudio e vídeo amostrados podem ser comprimidos através da supressão de seqüências de mesmos bytes. Estas seqüências são substituídas por um número de ocorrências e um símbolo padrão (padrão de bits) para anotar a repetição em sim. Obviamente, o fator de compressão alcançável depende do dado de entrada. Usando uma marca de exclamação como flag especial para indicar a codificação run-length, o seguinte exemplo mostra como um fluxo de dados pode ser comprimidos substituindo a seqüência de seis caracteres "H" por "!6H":

- Dado original UHHHHHHIMMG1223
- Dado comprimido: U!6HIMMG1223

É claro que esta técnica não é utilizada para seqüências de caracteres iguais ou menores que quatro. Isto pois nenhuma compressão seria obtida neste caso. Por exemplo, substituindo a seqüência de dois caracteres "M" com o código run-length "!2M" aumentaria o tamanho do código em um byte. Se o flag especial no nosso exemplo ocorrer no dado, ele deve ser substituído por duas marcas de exclamação (*byte stuffing*). O algoritmo apresentado acima pode ser facilmente otimizado; por exemplo, em vez de seqüências simples de caracteres, sentenças mais longas de diferentes caracteres podem também ser substituídas. Esta extensão requer que o tamanho da seqüência seja codificado ou pode-se utilizar um flag especial de fim. Existem diversas variações da codificação run-length.

Este método só traz ganhos relevantes se houver grandes agrupamentos de símbolos iguais. As principais aplicações do método de Run-Length, são em imagens binárias, imagens com grandes espaços envolvendo uma só cor e em imagens geradas por computador, onde os dados estão agrupados de forma mais geometricamente definida. Esse método é aplicado em formatos padrões como PCX, BMP(RLE).

Codificação de Huffman

Neste método de compressão, é atribuído menos bits a símbolos que aparecem mais freqüentemente e mais bits para símbolos que aparecem menos. Assim, o tamanho em bits dos caracteres codificados serão diferentes. Codificação de Huffman é um exemplo de técnica de codificação estatística, que diz respeito ao uso de um código curto para representar símbolos comuns, e códigos longos para representar símbolos pouco freqüentes. Esse é o princípio do código Morse, em que E é □e Q é --□-, e assim por diante.

Nós usaremos um exemplo para mostrar como a codificação de Huffman funciona. Suponha que nós temos um arquivo contendo 1000 caracteres, que são e, t, x e z. A probabilidade de ocorrência de e, t, x, e z são 0.8, 0.16, 0.02, e 0.02 respectivamente. Em um método de codificação normal, nós necessitamos 2 bits para representar cada um dos quatro caracteres. Assim, nós necessitamos de 2000 bits para representar o arquivo. Usando a codificação de Huffman, nós podemos usar quantidades de bits diferentes para representar estes caracteres. Nós usamos bit 1 para representar e, 01 para representar t, 001 para representar x e 000 para representar z. Neste caso, o número total de bits necessários para representar o arquivo é 1000*(1*0.8+2*0.16+3*0.02+3*0.02)=1240. Assim, embora tenhamos utilizado mais bits para representar x e z, desde que seus aparecimentos são mais raros, o número total de bits necessários para o arquivo é menor que o esquema de codificação uniforme. As regras para atribuir bits (códigos) aos símbolos é chamado um codebook. Codebooks são normalmente expressos em tabelas: w(e)=1, w(t)=01, w(x)=001, w(z)=000. Agora vamos ver como os códigos Huffman são gerados. O procedimento é o seguinte (Figura 1):

a) Coloque de todos os símbolos ao longo de uma linha de probabilidade acumulativa na seguinte ordem: probabilidade dos símbolos aumenta de baixo para cima. Se dois símbolos tem a mesma probabilidade, eles podem ser colocados em qualquer ordem.

b) Junta-se os dois símbolos de menor probabilidade a um nó para formar dois ramos na árvore.

c) A nova árvore formada é tratada como um símbolo único com a probabilidade igual a soma dos símbolos ramos.

d) Repita os passos (b) e (c) até que todos os símbolos sejam inseridos na árvore. O último nó formado se chama o nó raiz.

e) Partindo do nó raiz, atribua o bit 1 ao ramo de maior prioridade e bit 0 ao ramo de menor prioridade de cada nó.

f) O código para cada símbolo é obtido montando códigos ao longo dos ramos do nó raiz para a posição do símbolo na linha de probabilidade. Por exemplo, lendo do nó raiz ao símbolo x, nós obtemos o código 001 para x.

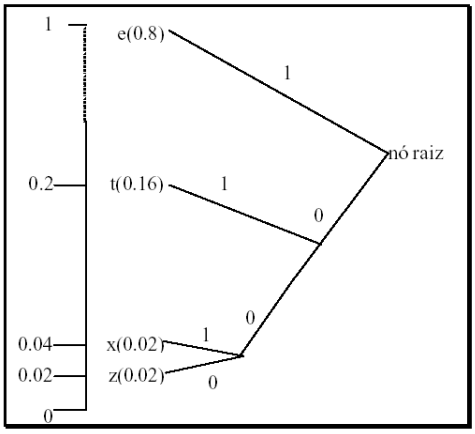


Figura 1. Exemplo de árvore de codificação Huffman

A operação computacional mais custosa na determinação do código Huffman é a adição de floats, mais especificamente, a adição da probabilidade da ocorrência no processo de redução. Isto ocorre no lado do codificador. No lado do decodificador, ele tem que apenas realizar uma simples verificação na tabela. Portanto, o decodificador necessita da tabela Huffman usado no codificador. Esta tabela é parte do fluxo de dados ou já é conhecida pelo decodificador. Em áudio e vídeo, tabelas Huffman padrões são com muita freqüência utilizadas, isto é, tabelas são conhecidas pelo codificador e decodificador. A vantagem é a obtenção de uma codificação mais rápida pois as tabelas não precisam ser calculadas. A desvantagem é que tabelas Huffman padrões obtém um fator de compressão um pouco menor porque as tabelas não são necessariamente ótimas para o dado a ser codificado. Portanto, métodos de compressão

executados em tempo-real usam normalmente tabelas padrões pois a codificação é mais rápida. Se alta qualidade é necessária mas tempo de codificação não é importante, tabelas Huffman otimizadas podem ser utilizadas. Normalmente nem todos os caracteres tem uma representação codificada na tabela Huffman: apenas aqueles caracteres com alta probabilidade de ocorrência. Todos os outros são codificados diretamente e marcados com um flag especial. Esta técnica é útil quando um número de caracteres diferentes é muito grande mas apenas alguns deles tem uma alta probabilidade de ocorrência.

Codificação de Lempel-Ziv-Welch (LZW)

Codificação LZW é baseada na construção de um dicionário de frases (grupos de um ou mais caracteres) a partir do fluxo de entrada. Quando uma nova frase é encontrada, a máquina de compressão adicionada ao dicionário e um token que identifica a posição da frase no dicionário substitui a frase. Se a frase já foi registrada, ela é substituída pelo token de posição no dicionário. Esta técnica é boa para compressão de arquivos textos, onde temos uma grande repetição de frases, como por exemplo em português: "ela", "Contudo,", ", "onde, ", aparecem freqüentemente no texto. O seguinte exemplo mostrará o poder da codificação LZW. Suponha que temos um arquivo de 10000 caracteres. Se nós representarmos o arquivo usando 8 bits por caractere, o arquivo requer 80000 bits para representá-lo. Usando o algoritmo LZW e assumindo que o arquivo tenha 2000 palavras ou frases das quais 500 são diferentes, então nós necessitamos 9 bits como token para identificar cada palavra ou frase distinta. Assim, nós precisamos de 9*2000 bits para codificar o arquivo. Com isto nós obtemos uma taxa de compressão de 4,4. Na prática, o dicionário armazenando todas as frases únicas deve ser armazenado também, baixando a taxa de compressão obtida.

Exemplo de Compressão

O funcionamento básico de LZW será ilustrado através da compressão da cadeia de caracteres ABACABA:

O primeiro passo consiste em inicializar uma tabela de códigos com todos os caracteres existentes na string que pretendemos comprimir: #0 = A, #1 = B, #2 =C, #3 = D.

Em seguida, começamos a analisar a string e verificamos que o primeiro caracter é o A, que já existe na tabela. Nesta situação, atribuímos um prefixo ao A e vamos analisar o caracter seguinte, que é o B. Juntando o prefixo com o novo caracter, obtemos a sub-string AB que não se encontra na tabela. Neste caso, enviamos o código #0, correspondente ao A e adicionamos a nova sub-string à tabela, fazendo #4 = AB.

Como próximo passo, tomamos para prefixo o caracter B e indo buscar o próximo caracter da string obtemos a sub-string BA, a qual também não se encontra na tabela. Repetindo o procedimento anterior, enviamos o código #1 e adicionamos a sub-string à tabela, fazendo #5 = BA.

O novo prefixo é o A, o qual concatenado com o próximo caracter da string forma a sub-string AC que também não está na tabela. Assim, enviamos o código #0 e fazemos #6 = AC.

O novo prefixo é o C, o qual concatenado com o próximo caracter da string forma a sub-string CA que também não se encontra na tabela. Enviamos o código #2,

correspondente ao prefixo e fazemos #7 = CA.

O novo prefixo é o A, o qual concatenado com o próximo caracter da string forma a sub-string AB, a qual já se encontra codificada na tabela. Desta forma, não enviamos nenhum código e tomamos para novo prefixo AB.

Este prefixo concatenado com o próximo caracter da string forma a sub-string ABA que não se encontra na tabela. Enviamos o código #4, correspondente a AB, e fazemos #8 = ABA.

O novo prefixo passa a ser A e como a string não tem mais caracteres, enviamos o código correspondente a este caracter, ou seja, o código #0.

Desta forma obtemos a seguinte seqüência codificada: #0 #1 #0 #2 #4 #0.

Padrão de Compressão de Imagens JPEG

O padrão JPEG (*Joint Photographic Expert Group*) foi desenvolvido, em 1992, pelo ISO/IEC em colaboração com a ITU-TS. Ele representa uma das melhores tecnologia de compressão de imagem. Dependendo da imagem, taxas de compressão podem alcançar 25 para 1 sem degradações notáveis. Em média, JPEG obtém uma taxa de compressão de 15:1.

O JPEG é usado hoje em dia por muitas aplicações envolvendo imagens. Ele pode ser implementado em software e hardware. Embora este padrão ter sido projetado inicialmente para imagens, codificação e decodificação JPEG tempo-real tem sido implementada para vídeo. Esta aplicação é chamada de *Motion JPEG* (MJPEG). Um dos objetivos do JPEG é cobrir uma grande faixa de qualidades de imagens e permitir especificar o comportamento de codificador a partir de parâmetros. Assim, a relação entre a taxa de compressão e a qualidade resultante pode ser selecionada pelo usuário ou pelo software aplicativo que usa JPEG. Outro objetivo foi permitir que a compressão fosse possível em uma grande diversidade de computadores com diferentes poder de processamento. Esta meta levou a definição de quatro modos de operação:

Codificação seqüencial: modo com perdas baseada em DCT. Cada componente de imagem é codificado em uma única varredura da esquerda para direita e de cima para baixo. Este modo é chamado de *baseline* e deve ser suportado por toda implementação JPEG.

Codificação progressiva: com perdas baseada em DCT expandido. Fornece avanços ao modo *baseline*. Uma expansão importante é a codificação progressiva (varreduras sucessivas), em que a imagem é codificada em varreduras múltiplas para produzir uma imagem de maneira rápida e rústica quando a largura de banda de transmissão é baixa.

Codificação sem perda: o processo de compressão é reversível, assim a reprodução é exata. Este modo sem perda não permite a obtenção de altos fatores de compressão, mas muitas aplicações necessitam armazenamento sem perda de imagens, como fotografias de raio X.

Codificação hierárquica: a imagem é codificada em resoluções múltiplas. Neste esquema nós temos uma codificação piramidal de uma imagem em resoluções espaciais múltiplas. Assim, versões podem ser acessadas sem a necessidade de primeiro descompactar a imagem na resolução completa.

Taxas de compressão obtidas

Como já referimos, quanto maior for a taxa de compressão maior será o número de componentes de alta freqüência desprezados. Para obtermos taxas de compressão muito elevadas, temos que deitar fora um número bastante significativo de componentes de alta freqüência, levando ao aparecimento do efeito de bloco, ou seja, perda de definição nos contornos das imagens. Geralmente obtêm-se bons resultados com a utilização de taxas de compressão da ordem dos 10% a 50%. Comparando as taxas de compressão com a qualidade de imagem obtida, podemos afirmar que:

Taxas de compressão de 10:1 a 20:1 – Alta qualidade de imagem;
Taxas de compressão de 30:1 a 50:1 – Média qualidade de imagem;
Taxas de compressão de 60:1 a 100:1 – Fraca qualidade de imagem.

JPEG é para imagens fotográficas

Este formato apresenta ótimas taxas de compressão para imagens fotográficas naturais multi-tonais, diminuindo consideravelmente quando aplicado a imagens gráficas com contornos e áreas bem definidas de cor ou a imagens com texto, como é o caso dos logotipos. O JPEG introduz ruído nas zonas de imagem compostas por cores sólidas, o qual pode distorcer o aspecto geral da imagem. Comparado ao GIF, verifica-se que a imagem GIF comprime mais eficazmente que a JPEG e que apresenta uma melhor definição dos contornos do texto.

CCITT H.261

Reconhecendo a necessidade de fornecer serviços de vídeo onipresentes na Rede Digital de Serviços Integrados (ISDN), o CCITT Study Group XV estabeleceu em 1984 um Grupo Especialista em Codificação para Telefonia Visual com o objetivo de recomendar uma codificação padrão de vídeo para transmissão a taxas, surgiu então a recomendação H.261.

H.261 é um dos padrões da família H.320 para videofonia e tele-conferência na taxa de 64 Kbps a 2 Mbps. Ele fornece poucos quadros por segundos com um resolução cerca de oito vezes mais baixa que a qualidade TV PAL/SECAM. O padrão H.261, também chamado de px64, obtém grandes taxas de compressão para a transmissão de vídeo colorido tempo-real. O algoritmo combina codificação intraquadro e interquadro (redundância espacial e temporal) para fornecer um rápido processamento para compressão/descompressão tempo-real de vídeo, otimizado para aplicações tal como telecomunicações baseada em vídeo. Como estas aplicações usualmente não são a movimentos intensos, o algoritmo usa uma limitada estratégia de busca e estimação de movimento para obter taxas de compressão mais altas. H.261 pode obter taxas de compressão de 100:1 a mais de 2000:1.

A recomendação H.261 define um padrão de codificação de vídeo para transmissão na taxa de p*64Kbps (p=1,2,3,4...30) que cobre as capacidades do canal ISDN. Sendo que a aplicação alvo desta recomendação são a videofonia e a teleconferência, onde o algoritmo de compressão de vídeo opera em tempo-real com atraso mínimo. Para p = 1 ou 2, devido a limitação de taxa de bits, apenas movimentos lentos, comunicação visual face-a-face (videofonia) são apropriados. Para p > 5, com uma maior taxa de bits disponível, imagens mais complexas podem ser transmitidas com melhor qualidade (videoconferência). Note que a máxima taxa de bits disponível é 1,92 Mbps (p=30), que é suficiente para obter imagens de qualidade VHS [Raghavan, 98]. H.261 opera com dois formatos de imagem: CIF (*Common Intermediate Format*) e QCIF (quarter-CIF). CIF, de 320x288, permite usar um formato único dentro e entre regiões usando padrões de TV de 625 e 525 linhas. QCIF, de tamanho 160x144, é mais útil em taxas de bit menores (p<6). O algoritmo de codificação é um híbrido de predição inter-quadro, transform coding (DCT), similar ao JPEG, e compensação de movimento. A taxa de dados do algoritmo de codificação foi projetado para ser capaz de suportar 40 kbps e 2Mbps. A predição interquadro remove a redundância temporal. O transform coding remove a redundância espacial. Para remover redundâncias adicionais no bitstream a ser transmitido, uma codificação por entropia (normalmente codificação de Huffman) é utilizado para reduzir ainda mais o vídeo.

H.263

H.263 é um padrão de vídeo a baixa taxa de bits para aplicações de teleconferência que opera a taxas abaixo de 64 Kbps. A codificação de vídeo é uma extensão do H.261 e descreve um método de codificação DPCM/DCT. Uma idéia interessante do H.263 é o quadro PB. Ele consiste de duas imagens codificadas em uma unidade. O nome PB é derivado da terminologia MPEG dos quadros P e B. Assim, um quadro PB consiste de um quadro P que é produzido a partir do último quadro P decodificado e um quadro B que é produzido a partir do último quadro P decodificado e do quadro P sendo decodificado.

H.263 suporta cinco resoluções. Além do QCIF e CIF que é suportado pelo H.261, existem o SQCIF, 4CIF e 16CIF. SQCIF é aproximadamente a metade da resolução do QCIF. 4CIF e 16CIF são aproximadamente 4 e 16 vezes a resolução do CIF. O suporte do 4CIF e 16CIF significa que o codec poderia então competir com outras codificações de mais altas taxas de bits como os padrões MPEG. Testes atuais mostram que o H.263 tem um desempenho 1 a 2,5 melhor que o H.261. Isto significa que, dada uma qualidade de imagem, a taxa de bits H.261 é aproximadamente 2,4 vezes a gerada pelo H.263.

ISO/IEC MPEG (Motion Picture Expert Group)

O grupo da ISO/IEC MPEG foi estabelecido em 1988 para desenvolver padrões para representação codificada de vídeos, áudios associados, e suas combinações quando usados

para armazenamento e recuperação em *Digital Storage Media* (DSM). O conceito DSM inclui os dispositivos de armazenamento convencionais, como CD-ROMs, drivers de fita, disco rígidos e canais de telecomunicação (ISDN e LAN). MPEG usa a compressão interquadros (redundância temporal), obtendo taxas de compressão de até 200:1 pelo armazenamento apenas das diferenças entre quadros sucessivos. Especificações MPEG também incluem um algoritmo para compressão de áudio a taxas de 5:1 a 10:1.

Grupos de Trabalho MPEG

O grupo teve 3 itens de trabalho: codificação de vídeos e áudios associados sobre 1.5, 10 e 40 Mbps. Estes grupos são abreviados por MPEG-1, MPEG-2 e MPEG-3:

A intenção do MPEG-1 (normalizado em 1993) é a codificação de vídeo com qualidade VHS: 360x280 pixels com 30 quadros por seg. na taxa de 1.5 Mbps (taxa dos drivers de CD-ROM da época). Quando se fala codificação MPEG é MPEG-1 que se está referenciando.

MPEG-2 (normalizado em 1994) visa a codificação de vídeo com qualidade de televisão digital CCIR 601: 720x480 pixels com 30 quadros por seg. na taxa entre 2 a 10 Mbps.

MPEG-3 visa a codificação de vídeo com qualidade HDTV na taxa de 40 Mbps. Estes trabalhos foram interrompidos em julho 1992. Durante o processo de normalização, viu-se a necessidade da definição de codificação audiovisual a taxas de bits muito baixas, assim o surgiu o MPEG-4 (a ser aprovado em dezembro de 1999). A taxa de bits considerada aqui varia de 4,8 a 64 Kbps. As atividades do MPEG cobre mais que a compressão de vídeo, desde que a compressão do áudio associado e a sincronização audiovisual não podem ser independente da compressão do vídeo.

Partes do padrão MPEG

O padrão MPEG tem 3 partes principais:
MPEG-Vídeo: trata da compressão de sinais de vídeo;
MPEG-Áudio: trata da compressão de um sinal de áudio digital; e
MPEG-Sistemas: trata da sincronização e multiplexação de bitstreams de áudio e vídeo compactados.

Além disso, uma quarta parte chamada Conformidade especifica o procedimento para determinar as características dos bitstreams codificados e para testar a conformância com os requisitos identificados no Áudio, Vídeo e Sistemas.

Bibliografia

[Buford, 94] John F. K. Buford. Multimedia Systems. ACM Press - SIGGRAPH Series - New York, New York, 1994.
[Fluckiger, 95] François Fluckiger. Understanding Networked Multimedia: Applications and Technology. Prentice Hall International (UK) Limited, 1995.
[Furht, 94] B. Furht. Multimedia Systems: An Overview. IEEE Multimedia, Spring 1994, pp. 47-59.
[Huang, ?] T. S. Huang, S. M. Kang, J. Stroming, MPEG-4 Project, Universidade de Illinois, EUA. http://uivlsi.csl.uiuc.edu/ stroming/mpeg4/
[Lu, 96] Guojun Lu. Communication and Computing for Distributed Multimedia Systems. Artech House Inc., 1996.
[Kuo, 98] Franklin Kuo, Wolfgang Efflesberg, J.J. Garcia-Luna-Aceves. Multimedia Communications: Protocols and Applications. Prentice Hall PTR, 1998.
[Patel, 93] K. Patel, B.C. Smith, L.A. Rowe. Performance of a Software MPEG Video Decoder. Proc. ACM Multimedia'93m ACM Press, New York, 1993, pp. 75-82.
[Raghavan, 98] V. Raghavan, S.K. Tripathi. Networked Multimedia Systems: Concepts, Architecture, and Design. Prentice Hall, 1998.
[Tanenbaum, 97] A. Tanenbaum. Computer Networks. Third Edition, Prentice Hall, 1997.
[Vidal, 97] P.C.S. Vidal. Evolução do Padrão MPEG.
http://www.gta.ufrj.br/~vidal/mpeg/mpeg.html
www.withoutprice.com/ptl_navegacod.asp?tit=3887
http://www.viaweb.com.br/ssl_pgp.htm
www.archive.com.br/pgp.html