CÍNTIA BORGES MARGI

UM MECANISMO PARA DISTRIBUIÇÃO SEGURA DE VÍDEO MPEG

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia.

Área de Concentração:
Sistemas Digitais

Orientadora:
Prof. Dra. Graça Bressan

SÃO PAULO
2000

Margi, Cíntia Borges
Um Mecanismo para Distribuição de Vídeo MPEG. São Paulo 2000.
126 p.

Dissertação (Mestrado) – Escola Politécnica da Universidade de São Paulo. Departamento de Computação e Sistemas Digitais.

1. Vídeo / distribuição 2. Segurança / vídeo I. Universidade de São Paulo. Escola Politécnica. Departamento de Computação e Sistemas Digitais. II. t

BC
Universidade de São Paulo
Biblioteca da Escola Politécnica
FD - 2743

CÍNTIA BORGES MARGI

UM MECANISMO PARA DISTRIBUIÇÃO SEGURA DE VÍDEO MPEG

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia.

SÃO PAULO
2000

A meus pais,

Michel e Cidália

A minhas avós (*in memorian*),

Florinda e Selma

"Valeu a pena? Tudo vale a pena
se a alma não é pequena.
Quem quere passar além do Bojador,
Tem que passar além da dor.
Deus ao mar o perigo e o abysmo deu,
mas nelle é que espelhou o céu."

Fernando Pessoa, em **Mar Portuguez**

AGRADEÇO

A Nossa Senhora, que por sua intercessão possibilitou enfrentar as dificuldades, e alcançar o final desta etapa.

A meus pais, Michel e Cidália, e meus irmãos, Flávia, Daniel e Raquel, pela paciência, compreensão e incentivo ao longo deste trabalho.

Aos professores Dra. Graça Bressan e Dr. Wilson Vicente Ruggiero, pela oportunidade de ingressar no mestrado na Escola Politécnica, pela orientação neste trabalho e pelas valiosas discussões.

A Oscar Dantas Vilcachagua, pelo apoio, pelas conversas e pela ajuda inestimável no desenvolvimento do protótipo.

A Regina Melo Silveira, que me ensinou muito do que é ser pesquisador, pelos debates sobre o assunto.

A todos os meus colegas e amigos, que de alguma forma contribuíram para a conclusão deste trabalho.

SUMÁRIO

Sumário 7

Lista de Figuras 12

Lista de Tabelas 14

Resumo 16

Abstract 17

Capítulo 1 Introdução *18*

1.1 Motivação 19

1.2 Objetivos 20

1.3 Estrutura Geral da Dissertação 22

Capítulo 2 Conceitos de Segurança *23*

2.1 Serviços de Segurança 24

2.2 Algoritmos de Criptografia 26

2.3 Criptografia Simétrica 26

2.3.1 DES (Data Encryption Standard) 27

2.3.2 3DES 33

2.3.3 IDEA (International Data Encryption Algorithm) 34

2.3.4 RC4 37

2.3.4.1 Funcionamento do RC4 37

2.3.5 Advanced Encryption Standard (Rijndael) 38

2.3.5.1 As iterações 39

2.3.5.2 A escala das chaves 42

2.4 Criptografia Assimétrica 43

2.4.1 RSA 44

2.5 Funções de Hash 45

2.5.1 MD-5 46

2.5.2 SHA-1 47

2.6 Esquemas de Distribuição de Chaves 48

2.6.1 PGP (Pretty Good Privacy) 49

2.6.2 Certificados Digitais 51

2.7 Protocolo SSL 52

2.7.1 SSL Record Protocol [Shostack 95] 53

2.7.2 SSL Handshake Protocol 53

2.7.3 Chaves no SSL 54

2.7.4 Protocolo de Aplicação SSL 55

2.8 Marca d'Água Digital 55

2.8.1 Protegendo Imagens Digitais através de Marca d'água [Voyatzis 99] ... 56

Capítulo 3 O Padrão de Compressão de Vídeo MPEG 58

3.1 Técnicas de Compressão 59

3.2 Sensibilidade do Olho Humano 61

3.3 Camada de Sistema **62**

3.4 Camada de Vídeo **62**

3.5 Processo de Compressão MPEG **65**

Capítulo 4 Mecanismos de Criptografia e Distribuição de Vídeo MPEG ***68***

4.1 Criptografia de Vídeo MPEG **69**

4.1.1 Criptografia Pura ou Simples 69

4.1.2 Criptografia Seletiva 70

4.1.2.1 SE_MPEG (Security Enhanced MPEG Player) 71

4.1.2.2 Aegis 73

4.1.2.3 SECMPEG 75

4.1.3 Algoritmo de Permutação em Zig-Zag 77

4.1.4 VEA (Video Encryption Algorithm) 78

4.1.5 Permutação Pura 79

4.1.6 Comparação Entre Os Mecanismos Descritos 79

4.2 Mecanismos de Distribuição de Vídeo **80**

4.2.1 Proteção de Vídeo sob Demanda 80

Capítulo 5 Mecanismo para a Distribuição Segura de Vídeo MPEG ***82***

5.1 Requisitos **83**

5.1.1 Controle de acesso 85

5.1.2 Reprodução de Material Autenticado e Íntegro 87

5.1.3 Níveis de Segurança 88

5.1.4 Confiabilidade do protocolo 89

5.2 Arquitetura do Sistema **89**

5.3 Especificação **90**

5.3.1 Protocolo de Acesso 90

5.3.2 O servidor de Vídeo: S/Server 96

5.3.3 Reprodução do Vídeo MPEG: S/Viewer 99

Capítulo 6 Projeto, Implementação e Resultados Obtidos ***102***

6.1 Projeto **103**

6.1.1 Interface do S/Viewer 103

6.1.2 Interface do S/Server 104

6.1.3 Máquina de Estados do S/Server 105

6.2 Definições para a Implementação **106**

6.2.1 Ambiente e Linguagem de Programação 107

6.2.2 Aplicativos e Bibliotecas 107

6.2.3 Múltiplas Conexões 108

6.2.4 Campo de Controle das Mensagens e Códigos de Erro 108

6.3 Detalhes da Implementação do Protótipo **109**

6.3.1 Compilação do Código MPEG 109

6.3.2 Criação da interface do Cliente 109

6.3.3 Infra-estrutura de Rede 111

6.3.4 Protocolo de Acesso 111

6.3.5 Identificação do Tipo de Vídeo 112

6.3.6 Criptografia de Vídeo 113

6.3.6.1 Criptografia Pura 113

6.4 Resultados Obtidos **114**

6.4.1 Comparação com SSL 114

6.4.2 Efeitos do Atraso 115

Capítulo 7 Considerações Finais e Trabalhos Futuros *118*

7.1 Motivação **119**

7.2 Objetivos **119**

7.3 Resultados Obtidos **120**

7.4 Trabalhos Complementares **121**

Referências Bibliográficas *122*

LISTA DE FIGURAS

FIGURA 2.1: CRIPTOGRAFIA SIMÉTRICA 26

FIGURA 2.2: ESTRUTURA GERAL DO ALGORITMO DES 28

FIGURA 2.3: ESTRUTURA DE UMA ITERAÇÃO 29

FIGURA 2.4: EXEMPLO DE UTILIZAÇÃO DAS TABELAS S-BOX 32

FIGURA 2.5: ESTRUTURA DO 3DES 33

FIGURA 2.6: ESTRUTURA GERAL DO ALGORITMO IDEA 35

FIGURA 2.7: BLOCO BÁSICO DE OPERAÇÃO DO IDEA 36

FIGURA 28: ESTRUTURA DE UMA ITERAÇÃO 36

FIGURA 2.9: TRANSFORMAÇÃO DE SAÍDA DO IDEA 37

FIGURA 2.10: ESTRUTURA DO ALGORITMO RIJNDAEL [JSAVARD] 42

FIGURA 2.11: CRIPTOGRAFIA ASSIMÉTRICA 44

FIGURA 2.12: ALGORITMO DO MD-5 47

FIGURA 2.13: CAMADA SSL 52

FIGURA 3.1: ESTRUTURA DA CAMADA DE VÍDEO MPEG 63

FIGURA 3. 2: CONSTITUIÇÃO DO MACROBLOCO MPEG 64

FIGURA 3. 3: INTERDEPENDÊNCIA DE QUADROS PARA UMA SEQÜÊNCIA MPEG 65

FIGURA 3.4: EXEMPLO DE QUANTIZAÇÃO 65

FIGURA 3.5:APLICAÇÃO EM ZIG-ZAG DA TÉCNICA DE SUPRESSÃO DE CARACTERES 66

FIGURA 4.1: CADEIA DE VÍDEO SECMPEG 76

FIGURA 4. 2: MÓDULOS DO SECMPEG 76

FIGURA 5.1: ARQUITETURA DO SISTEMA 89

FIGURA 5. 2: DIAGRAMA DE TEMPO DAS MENSAGENS TROCADAS ENTRE O CLIENTE E O SERVIDOR PARA ACESSO AO VÍDEO MPEG SEGURO 91

FIGURA 5.3: FORMATO DA MENSAGEM DE IDENTIFICAÇÃO DO CLIENTE 91

FIGURA 5. 4: CERTIFICADO DIGITAL 92

FIGURA 5.5: FORMATO DA MENSAGEM DE IDENTIFICAÇÃO DO SERVIDOR 93

FIGURA 5.6: FORMATO DA MENSAGEM DE REQUISIÇÃO DE LISTA DE VÍDEOS 93

FIGURA 5.7: FORMATO DA MENSAGEM COM A LISTA DE VÍDEOS 94

FIGURA 5. 8: FORMATO DA MENSAGEM DE REQUISIÇÃO DE VÍDEO 94

FIGURA 5. 9: FORMATO DA MENSAGEM DE ERRO 95

FIGURA 5.10: MENSAGEM DE RESET 95

FIGURA 5.11: FLUXOGRAMA DA CRIPTOGRAFIA E TRANSMISSÃO DO VÍDEO SEGURO 96

FIGURA 5.12: CABEÇALHO DO ARQUIVO MPEG SEGURO (VMS) 97

FIGURA 5.13: FLUXOGRAMA DA CRIPTOGRAFIA E TRANSMISSÃO DO VÍDEO SEGURO POR *STREAMING* 98

FIGURA 5.14: FLUXOGRAMA DO MECANISMO DE REPRODUÇÃO DE VÍDEO MPEG CRIPTOGRAFADO 100

FIGURA 6.1: MÁQUINA DE ESTADOS DO S/SERVER 105

FIGURA 6.2: ESTRUTURA DE PROCESSOS CONCORRENTES DO SERVIDOR 108

FIGURA 6.3: PROTÓTIPO DO S/VIEWER 110

LISTA DE TABELAS

TABELA 2.1: TABELA DA PERMUTAÇÃO INICIAL 29

TABELA 2.2: TABELA DA INVERSA DA PERMUTAÇÃO INICIAL 29

TABELA 2. 3: TABELA DA PERMUTAÇÃO/EXPANSÃO 30

TABELA 2. 4: TABELA DA PERMUTAÇÃO P 30

TABELA 2.5: TABELA DE SUBSTITUIÇÃO DOS S-BOX 31

TABELA 2.6: PERMUTAÇÕES PC1 32

TABELA 2.7: PERMUTAÇÕES PC2 32

TABELA 2.8: DESLOCAMENTOS DA CHAVE 33

TABELA 2.9: EXEMPLOS DAS OPERAÇÕES DO IDEA COM DOIS BITS 34

TABELA 2.10: S-BOX DO RIJNDAEL 40

TABELA 2.11: DESLOCAMENTO PARA BLOCOS DE 128 BITS 39

TABELA 2.12: DESLOCAMENTO PARA BLOCOS DE 192 BITS 40

TABELA 2.13: DESLOCAMENTO PARA BLOCOS DE 256 BITS 41

TABELA 2.14: MATRIZ DE MISTURA DE COLUNAS 41

TABELA 2.15: CONSTANTES DE ITERAÇÃO DE CÁLCULO DAS SUB-CHAVES 43

TABELA 3.1: GANHO OBTIDO COM COMPRESSÃO 61

TABELA 3.2: CÓDIGOS DE INÍCIO DE VÍDEOS MPEG 63

TABELA 4.1: ARQUIVOS UTILIZADOS NA AVALIAÇÃO DE DESEMPENHO 72

TABELA 4. 2: COMPARAÇÃO DA VELOCIDADE DOS *PLAYERS* MPEG CONVENCIONAL E SEGURO 73

TABELA 4. 3: DEGRADAÇÃO PERCENTUAL COM O SE_MPEG 73

TABELA 4. 4: TEMPO DE ENCRIPTAÇÃO DOS QUADROS DURANTE A TRANSMISSÃO 73

TABELA 4. 5: DESEMPENHO DO ALGORITMO DE PERMUTAÇÃO ZIG-ZAG 78

TABELA 4. 6: DESEMPENHO DO VEA 79

TABELA 4.7: COMPARAÇÃO DOS ALGORITMOS DE CRIPTOGRAFIA MPEG 80

TABELA 6.1: CAMPO DE CONTROLE DAS MENSAGENS 108

TABELA 6.2: CÓDIGOS DE ERRO 109

TABELA 6.3: VÍDEOS DE TESTE 116

TABELA 6.4: ATRASO INTRODUZIDO PELA CRIPTOGRAFIA DO VÍDEO 116

TABELA 6.5: ATRASO PERCENTUAL 117

RESUMO

Com a evolução das aplicações disponíveis na Internet, o uso de multimídia torna-se cada vez mais comum. Dentro deste contexto, o vídeo está sendo cada vez mais utilizado, principalmente com conteúdos valiosos, como é o caso do ensino a distância, às vezes com fins comerciais. Assim, a discussão dos aspectos de segurança envolvidos torna-se um assunto muito importante.

Este trabalho tem como objetivo levantar os principais aspectos envolvidos na distribuição segura de material multimídia, mais especificamente de vídeo MPEG, propor um mecanismo de distribuição e reprodução de vídeos MPEG que atenda a estes requisitos, e demonstrar a sua viabilidade.

Para identificar os requisitos de segurança necessários ao mecanismo proposto, este trabalho faz uma análise dos diversos métodos de criptografia para vídeos MPEG propostos em outros centros de pesquisa. A partir dos principais requisitos, um mecanismo para distribuição segura de vídeo MPEG é proposto. Este mecanismo consiste de um visualizador, o S/Viewer, de um servidor de vídeo, o S/Server, e de um protocolo para a comunicação entre ambos.

Após a especificação do mecanismo proposto, a sua viabilidade pode ser verificada através da implementação do protótipo.

ABSTRACT

With the evolution of the Internet applications, the use of multimedia is even more common. In this context, video is being much used, mainly with valuable contents, as distance learning, sometimes in a commercial environment. So, discussing security characteristics becomes an important issue.

This work has as goals to identify the main characteristics in secure multimedia distribution, mainly video MPEG, propose a mechanism for MPEG video distribution and reproduction that agree with these requirements, and demonstrate its viability.

In order to identify these security requirements, an analysis from several encryption methods developed in other research center is done. From these requirements, "A Secure Video Distribution Mechanism" is proposed. A player, S/Viewer, the server, S/Server, and the communication protocol between them compose this mechanism.

After the specification, its viability can be verified with the prototype implementation.

Capítulo 1

INTRODUÇÃO

Este capítulo trata das justificativas e motivações para o desenvolvimento deste trabalho. Também apresenta a estrutura geral desta dissertação.

1.1 MOTIVAÇÃO

Com o aumento da distribuição de material multimídia através da Internet, principalmente com conteúdos valiosos, como é o caso do ensino a distância, a discussão dos aspectos de segurança envolvidos torna-se um assunto muito importante. Esta questão fica mais interessante quando se leva em conta a distribuição de material multimídia com acesso controlado e a sua exibição em tempo real.

Além disso, sistemas de Vídeo sob Demanda também estão sendo explorados comercialmente, o que implica em preocupação adicional por parte de empresas, pressionando os grupos de pesquisa a desenvolverem mecanismos para proteção de direitos autorais.

Com a chegada da Internet 2 e de outros meios de transmissão de alta velocidade, como as redes de TV a cabo e redes sem fio, os usuários podem acessar vídeos de melhor qualidade [Orzessek 98]. Além disto, estas redes permitem a distribuição de material multimídia em maior escala e até comercialmente. Num ambiente comercial, o público é selecionado, sendo vital a confidencialidade, a proteção do direito de acesso e do direito de propriedade (autoral).

Outra aplicação de vídeo que vem se tornando bastante comum atualmente é a vídeo-conferência, devido às diferentes localizações dos funcionários de uma empresa pelo mundo. Este tipo de aplicação de vídeo tem requisitos fortíssimos quanto ao sigilo dos dados transmitidos [Kunkelmann 97].

Atualmente as implementações de vídeo distribuído (Vídeo sob Demanda ou vídeo-conferência) preocupam-se somente com o controle de acesso, existindo a necessidade

de definir mecanismos e implementar o sigilo e a proteção do direito autoral [Kunkelmann 97].

Além do Vídeo sob Demanda e da Vídeo-conferência, o *multicast* de vídeo (vários clientes recebem um mesmo vídeo de um servidor, sem que haja inúmeras cópias sendo transmitidas pela rede) também é uma aplicação cujo uso tem se intensificado. Da mesma forma que as aplicações citadas anteriormente, diversos problemas de segurança são encontrados quando da distribuição de vídeo através de *multicast*. [Hardjono 00] indica alguns problemas e possíveis direções para solucioná-los.

Apesar de muito estar sendo pesquisado e desenvolvido para tentar solucionar estas questões, ainda há muito o que fazer. Desta forma, este trabalho pode contribuir significativamente com a proposta deste "Mecanismo para Distribuição Segura de Vídeo MPEG".

1.2 OBJETIVOS

Discutir os aspectos de segurança envolvidos na distribuição de material multimídia, mais especificamente de vídeo, significa considerar algumas questões principais como: controle de acesso, integridade e sigilo. Estas questões estão interligadas, já que o sigilo torna-se relevante quando o acesso ao material é controlado, ou seja, somente usuários autorizados podem utilizá-lo.

Os textos, animações, desenhos e simulações são transmitidos através de SSL (*Secure Sockets Layer*), utilizando criptografia e certificados digitais. O uso de certificados digitais garante a autenticidade do servidor, e o uso de criptografia garante a

confidencialidade e a integridade das informações. Portanto, esta questão possui uma solução satisfatória.

Já o vídeo possui características bem diferentes das demais mídias. A sua distribuição é feita através de mecanismos próprios (*streaming*), que não contemplam aspectos de segurança.

Existem diversos padrões de compressão de vídeo, sendo que o padrão a ser utilizado neste trabalho é o padrão MPEG. Outros padrões de compressão de vídeo poderiam ter sido escolhidos ou até desenvolvidos, como ocorre em [Kunkelmann 98]. Porém, o padrão MPEG vem sendo recomendado por diversos fóruns e grupos de trabalho. O ATM Fórum recomenda que o padrão de compressão MPEG-2 seja utilizado em sistemas de Vídeo sob Demanda com redes ATM [Orzessek 98], da mesma forma que o DAVIC (Digital Audio-Visual Council) também recomenda este padrão para vídeo [Orzessek 98]. Da mesma forma, o IETF (Internet Engineering Task Force) também considera o padrão MPEG em suas recomendações e normas [Orzessek 98].

Este trabalho propõe-se a analisar os aspectos de segurança envolvidos, levantar os requisitos, especificar um mecanismo para distribuição segura de vídeo MPEG, implementar um protótipo e verificar a sua viabilidade.

O objetivo deste esquema é distribuir vídeos MPEG para diversas aplicações diferentes, como cursos online, seminários, ou suporte ao ensino presencial. Assim, os vídeos serão assistidos por diferentes usuários, implicando em vídeos de acesso público e de acesso restrito.

1.3 ESTRUTURA GERAL DA DISSERTAÇÃO

Esta dissertação está organizada de forma a primeiro prover ao leitor os conceitos necessários ao entendimento do "Mecanismo para a Distribuição Segura de Vídeo MPEG", para depois explicá-lo.

Assim, o capítulo 2 trata dos conceitos de segurança envolvidos neste trabalho, dos conceitos dos serviços de segurança e de diversos mecanismos de criptografia de dados.

O capítulo 3 explica as principais características do padrão de compressão MPEG, necessárias ao entendimento dos algoritmos existentes para a criptografia de vídeo MPEG, apresentados no capítulo 4 juntamente com outros mecanismos para vídeo.

O capítulo 5 trata dos requisitos e especificação do Mecanismo para a Distribuição Segura de Vídeo MPEG, principal objetivo deste trabalho, enquanto o capítulo 6 apresenta o projeto, implementação e relata os experimentos realizados, mostrando a viabilidade de sua implementação.

Já o capítulo 7 trata das conclusões deste trabalho, bem como da proposta de trabalhos futuros. As referências bibliográficas estão no capítulo 8.

Capítulo 2

CONCEITOS DE SEGURANÇA

Este capítulo trata dos conceitos dos serviços de segurança, de diversos mecanismos de criptografia de dados, trocas de chaves e marcas d'água.

Os aspectos de segurança a serem enfocados neste trabalho envolvem sigilo (ou confidencialidade), controle de acesso e integridade. O sigilo e a integridade são obtidos através de criptografia. A troca de certificados digitais permite garantir a autenticidade do servidor e do cliente envolvidos na transação. O controle de acesso pode ser implementado através de validação de senhas, ou através de identificação por certificados digitais.

2.1 Serviços de Segurança

Os serviços de segurança caracterizam os diferentes aspectos relevantes a segurança de um sistema de computadores. Estes serviços são: autenticidade, confidencialidade, integridade, disponibilidade, controle de acesso, não-repúdio e auditoria [Stallings 98].

Autenticidade: Requer que a origem ou o originador de uma mensagem seja corretamente identificado. A verificação da autenticidade é necessária após todo processo de identificação, seja de um usuário para um sistema, de um sistema para o usuário ou de um sistema para outro sistema.

Confidencialidade: Consiste em proteger a informação contra leitura ou cópia por alguém que não tenha sido explicitamente autorizado pelo proprietário daquela informação. A informação deve ser protegida qualquer que seja a mídia que a contenha: impressa, digital. Este tipo de segurança inclui não apenas a proteção da informação como um todo, mas também de partes da informação que podem ser utilizadas para inferir sobre o todo.

Integridade: Consiste em proteger a informação contra modificação sem a permissão explicita do proprietário daquela informação. A modificação inclui ações como escrita,

alteração de conteúdo, alteração de *status*, remoção, criação e o atraso de informações transmitidas. Deve-se considerar a proteção da informação nas suas mais variadas formas: armazenada em discos, fitas de *backup*, etc ...

Disponibilidade: Consiste em proteger os serviços prestados pelo sistema, de forma que não sejam degradados ou tornem-se indisponíveis sem autorização. Um sistema indisponível quando um usuário autorizado necessita dele pode resultar em perdas tão graves quanto as causadas pela remoção das informações daquele sistema.

Controle de Acesso: Consiste na capacidade de se permitir ou negar acesso aos serviços e recursos oferecidos pelo sistema. Acessos desconhecidos ou feitos por pessoas não autorizadas podem significar a necessidade de uma verificação de todos os recursos envolvidos em busca de possíveis estragos que possam ter sido causados ao sistema, mesmo que aparentemente nada tenha ocorrido.

Não-repúdio: Requer que o originador de uma mensagem (ou ação) não possa negar futuramente o envio da mensagem (ou a realização da ação). Do mesmo modo, o receptor de uma mensagem (ou ação) não deve ser capaz de negar o recebimento da mensagem (ou ação).

Auditoria: Consiste na capacidade de verificação das atividades do sistema e determinação do que foi feito, por quem, quando e o que foi afetado. A auditoria aplica-se não apenas à verificação de atividades de usuários não autorizados, mas também dos usuários autorizados, que podem cometer erros ou executar ações maliciosas no sistema.

Nesta dissertação, os serviços de segurança mais relevantes para a especificação e implementação do S/Viewer são: a confidencialidade, a autenticidade e a integridade.

2.2 ALGORITMOS DE CRIPTOGRAFIA

Criptografia é a arte e ciência que estuda como manter informações seguras. Consiste na técnica de transformar um texto claro em um texto cifrado, ou ininteligível, para que este possa ser transmitido ou armazenado. O uso de criptografia implementa, além da confidencialidade, a autenticidade e a verificação da integridade.

Existem dois tipos principais de criptografia: a simétrica (ou convencional) e a assimétrica. A criptografia simétrica utiliza uma única chave no processo de criptografia, e esta chave deve ser mantida em segredo. A criptografia assimétrica utiliza duas chaves, matematicamente relacionadas, sendo uma delas para encriptar e outra para decriptar. Uma das chaves é mantida em segredo, e a outra é divulgada.

2.3 CRIPTOGRAFIA SIMÉTRICA

Na criptografia simétrica, a mesma chave é utilizada para encriptar e decriptar uma mensagem. Conforme observa-se na Figura 2.1, que ilustra o processo de criptografia simétrica, somente o remetente e o destinatário devem conhecer a chave secreta, pois esta é a única maneira de obter a confidencialidade.

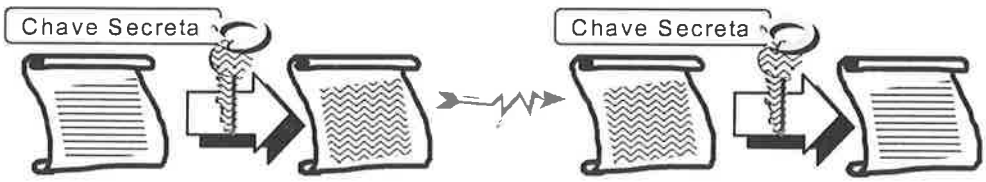

Figura 2.1: Criptografia Simétrica

Os principais algoritmos de Chaves Simétricas são:

- DES (*Data Encryption Standard*) e 3DES

- IDEA (*International Data Encryption Algorithm*)
- RC2 e RC4
- AES (*Advanced Encryption Standard*)

Estes algoritmos utilizam chaves com comprimento de 56 a 256 bits.

O maior problema dos algoritmos de criptografia simétrica é a distribuição da chave secreta. Como transmiti-la se o canal de comunicação existente não é seguro? Assim, para que seja possível utilizar este tipo de algoritmo é necessário transmitir a chave secreta por outro meio seguro! Em compensação, estes algoritmos de criptografia são mais eficientes para encriptar grandes massas de dados.

2.3.1 DES (Data Encryption Standard)

O DES (*Data Encryption Standard*) é um algoritmo adotado como padrão em 1977 pelo *National Institute of Standards and Technologies* (NIST). O algoritmo foi originalmente proposto em 1971 pela IBM, com o nome de LUCIFER. Em 1973, o NIST fez uma requisição de submissão de proposta de algoritmos de criptografia, e a IBM submeteu o LUCIFER. Após algumas modificações, como a diminuição do tamanho da chave de 128 para 56 bits, este algoritmo tornou-se o DES.

Este algoritmo [Stallings 98] baseia-se na permutação e substituição dos bits da mensagem a ser encriptada. Este modo de funcionamento do DES permite que seja implementado eficientemente através de *hardware*.

O algoritmo é composto por uma permutação inicial, 16 iterações, seguida por uma troca entre os 32 bits mais significativos e os 32 menos significativos, e uma permutação inversa da inicial. O DES utiliza uma chave de 56 bits, sendo que os bits múltiplos de oito são desprezados quando se entra uma chave de 64 bits. A mensagem a

ser criptografada é dividida em blocos de 64 bits, que são, então, encriptados ou decriptados.

A estrutura geral do algoritmo do DES é mostrada Figura 2.2, sendo cada um dos módulos principais de funcionamento do DES detalhados em seguida.

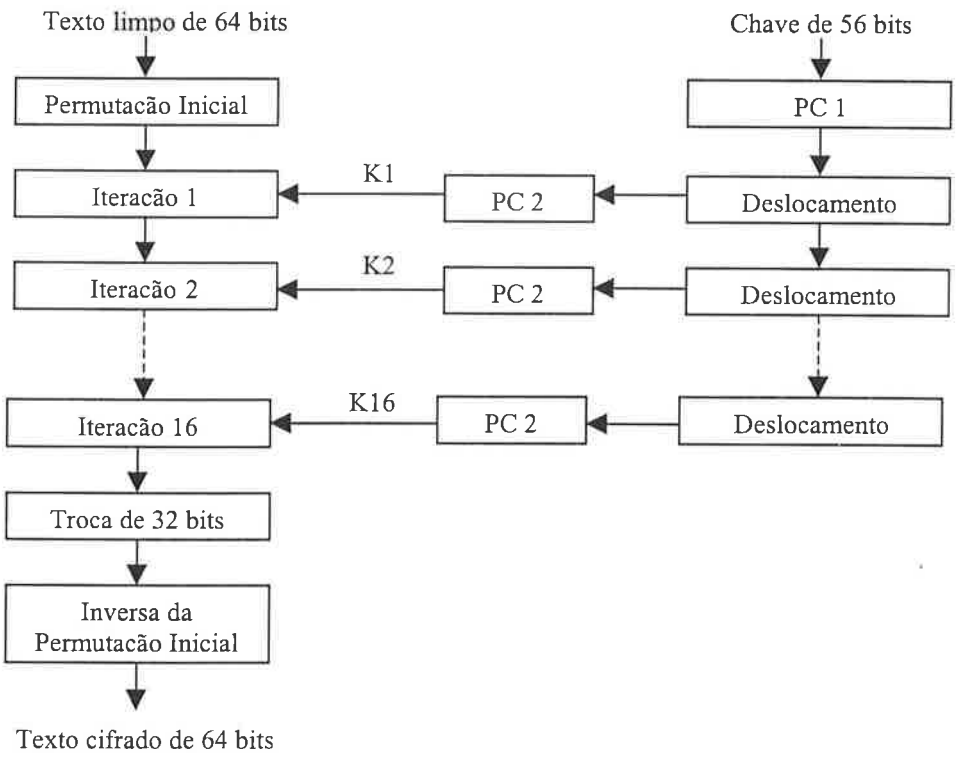

Figura 2.2: Estrutura Geral do Algoritmo DES

A Permutação Inicial e a Inversa da Permutação Inicial são realizadas de acordo com as Tabelas de Permutação, que fazem parte do algoritmo do DES. A Tabela 2.1 mostra a Permutação Inicial, e a Tabela 2.2 contém a Inversa da Permutação Inicial, ou Permutação Final.

Na permutação inicial (IP), o primeiro bit é trocado pelo qüinquagésimo oitavo, o segundo pelo qüinquagésimo, e assim até que o sexagésimo quarto é trocado pelo sétimo.

Tabela 2.1: Tabela da Permutação Inicial

Permutação Inicial (IP)															
58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

A permutação final é a inversa da permutação inicial, ou seja, se na permutação inicial o primeiro bit é trocado pelo qüinquagésimo oitavo, na permutação final o qüinquagésimo oitavo bit é trocado pelo primeiro.

Tabela 2.2: Tabela da Inversa da Permutação Inicial

Permutação Final (IP^{-1})															
40	8	48	16	56	24	64	32	39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30	37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28	35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26	33	1	41	9	49	17	57	25

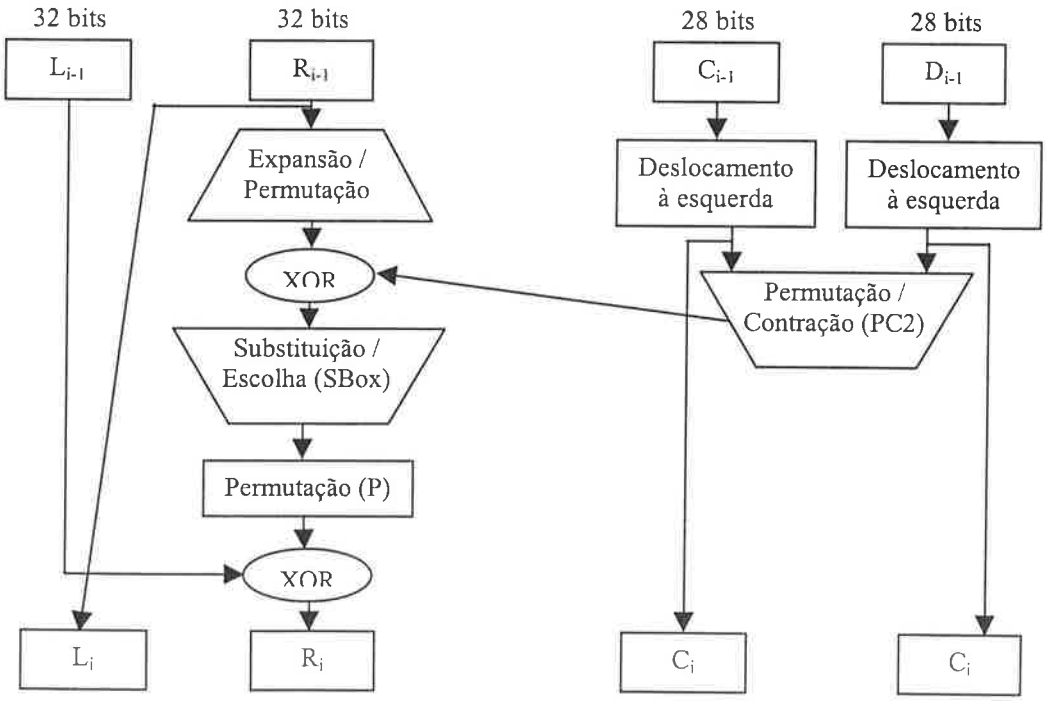

Figura 2.3: Estrutura de uma Iteração

Uma iteração é composta de diversas permutações, utilizando as tabelas de substituição S-Box e Ou-Exclusivos. A Figura 2.3 mostra como estas técnicas são empregadas em uma iteração.

A parte esquerda da Figura 2.3 é relativa aos dados, enquanto a parte direita relaciona-se com a chave de criptografia.

A Permutação/Expansão e a Permutação que são utilizadas na iteração obedecem respectivamente a Tabela 2. 3 e a Tabela 2. 4.

Tabela 2. 3: Tabela da Permutação/Expansão

Expansão / Permutação (E)											
32	1	2	3	4	5	4	5	6	7	8	9
8	9	10	11	12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21	22	23	24	25
24	25	26	27	28	29	28	29	30	31	32	1

Na Permutação / Expansão alguns bits são repetidos durante a troca de bits para que se aumente de 32 para 48 bits.

Tabela 2. 4: Tabela da Permutação P

Permutação P							
16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

A substituição que é feita no bloco "Substituição / Escolha (S-Box)" é feita de acordo com a Tabela 2.5. Para efetuar a substituição, os 48 bits são agrupados seis a seis, sendo que o primeiro grupo utiliza S[1], o segundo S[2], e assim por diante.

O primeiro e o sexto bits são utilizados para selecionar a linha, enquanto os outros quatro bits do meio selecionam a coluna. O número obtido na tabela irá substituir os seis bits de entrada.

Tabela 2.5: Tabela de Substituição dos S-Box

Substituição / Escolha (S-Box)															
S[1]															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S[2]															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S[3]															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S[4]															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S[5]															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S[6]															
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S[7]															
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S[8]															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Por exemplo, se o número 011011 compõe o primeiro grupo de seis bits, utiliza-se o número 01 (1) para selecionar a linha e o número 1101 (13) para determinar a coluna na tabela S[1], obtendo o número 5 como resultado, conforme observa-se na Figura 2.4.

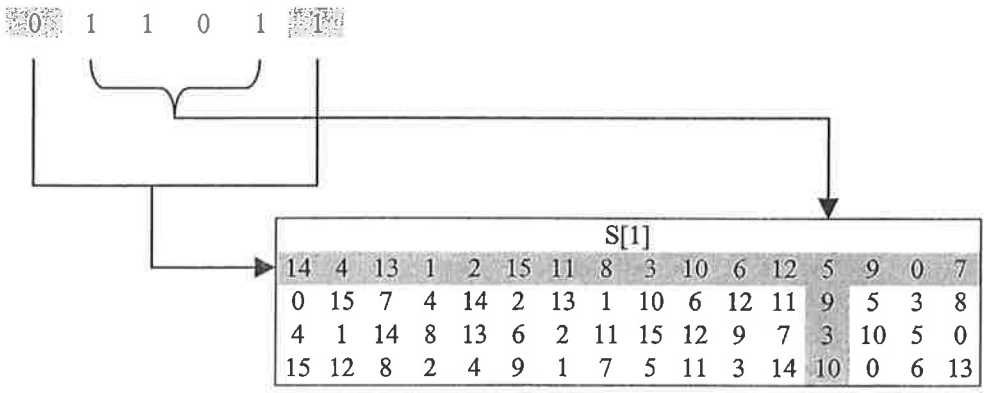

Figura 2.4: Exemplo de Utilização das tabelas S-Box

A Figura 2.3 mostra que a chave de criptografia também sofre deslocamentos, permutações e contrações. A permutação inicial da chave (Permutação Escolha) PC1 é apresentada na Tabela 2.6.

Tabela 2.6: Permutações PC1

Permutação Escolha 1 (PC-1)													
57	49	41	33	25	17	9	1	58	50	42	34	26	18
10	2	59	51	43	35	27	19	11	3	60	52	44	36
63	55	47	39	31	23	15	7	62	54	46	38	30	22
14	6	61	53	45	37	29	21	13	5	28	20	12	4

Com o detalhamento da estrutura de uma iteração, percebe-se que ocorre a Permutação/Contração (Permutação Escolha) PC2 em cada iteração. Esta permutação / contração obedece a Tabela 2.7, sendo que dos 56 bits da chave, apenas 48 são utilizados.

Tabela 2.7: Permutações PC2

Permutação Escolha 2 (PC-2)											
14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

A Tabela 2.8 mostra como é determinado o número de bits da chave que serão deslocados circularmente, ou rotacionados, em cada iteração.

Tabela 2.8: Deslocamentos da Chave

Deslocamento à esquerda da Chave																
Iteração	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits rotacionados	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

O algoritmo de decriptografia do DES é igual ao algoritmo de criptografia, porém as chaves são calculadas na ordem inversa.

2.3.2 3DES

O 3DES é uma forma de aumentar o tamanho da chave de criptografia do algoritmo DES. Para tanto, aplica-se o algoritmo DES três vezes, alternadamente, encriptando (**E_1**), decriptando (**D**) e encriptando (**E_2**).

Esta implementação pode utilizar-se de duas chaves distintas K_1 e K_2, conforme a Figura 2.5. Como são utilizadas duas chaves de 56 bits, o resultado é equivalente a uma chave de 112 bits.

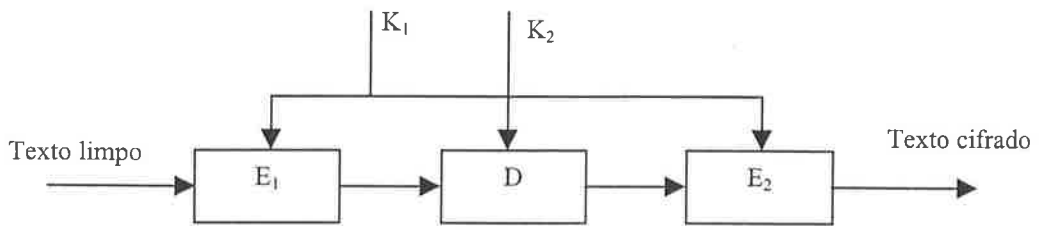

Figura 2.5: Estrutura do 3DES

O algoritmo 3DES também pode utilizar-se de três chaves de criptografia diferentes, sendo a primeira para **E_1**, a segunda para **D** e a terceira para **E_2**. Desta forma a chave de criptografia é equivalente a 168 bits.

2.3.3 *IDEA (International Data Encryption Algorithm)*

Este algoritmo foi desenvolvido no Instituto de Tecnologia Suíço por Xuejia Lai e James Massey. Um dos objetivos da implementação deste algoritmo era substituir o algoritmo DES.

O IDEA [Stallings 98] utiliza uma chave de 128 bits e processa o texto a ser criptografado em blocos de 64 bits.

Tabela 2.9: Exemplos das operações do IDEA com dois bits

X		Y		X + Y		X * Y		X ⊕ Y	
0	00	0	00	0	00	1	01	0	00
0	00	1	01	1	01	0	00	1	01
0	00	2	10	2	10	3	11	2	10
0	00	3	11	3	11	2	10	3	11
1	01	0	00	1	01	0	00	1	01
1	01	1	01	2	10	1	01	0	00
1	01	2	10	3	11	2	10	3	11
1	01	3	11	0	00	3	11	2	10
2	10	0	00	2	10	3	11	2	10
2	10	1	01	3	11	2	10	3	11
2	10	2	10	0	00	0	00	0	00
2	10	3	11	1	01	1	01	1	01
3	11	0	00	3	11	2	10	3	11
3	11	1	01	0	00	3	11	2	10
3	11	2	10	1	01	1	01	1	01
3	11	3	11	2	10	0	00	0	00

O algoritmo utiliza três operações básicas, cujos exemplos estão na Tabela 2.9:

- ou-exclusivo bit a bit (⊕);
- soma de números inteiros sem sinal módulo 2^{16} (65.536) ($\boxplus$);
- multiplicação de números inteiros sem sinal módulo $2^{16} + 1$ (65.537) ($\boxed{*}$).

Estas operações não apresentam propriedades associativas ou distributivas. A Tabela 2.9 apresenta um exemplo destas operações para dois bits.

A estrutura geral do algoritmo do IDEA é apresentada na Figura 2.6. O algoritmo é constituído de oito iterações e de um bloco com transformações de saída. A figura também mostra o bloco que gera as chaves parciais, usadas em cada uma das iterações.

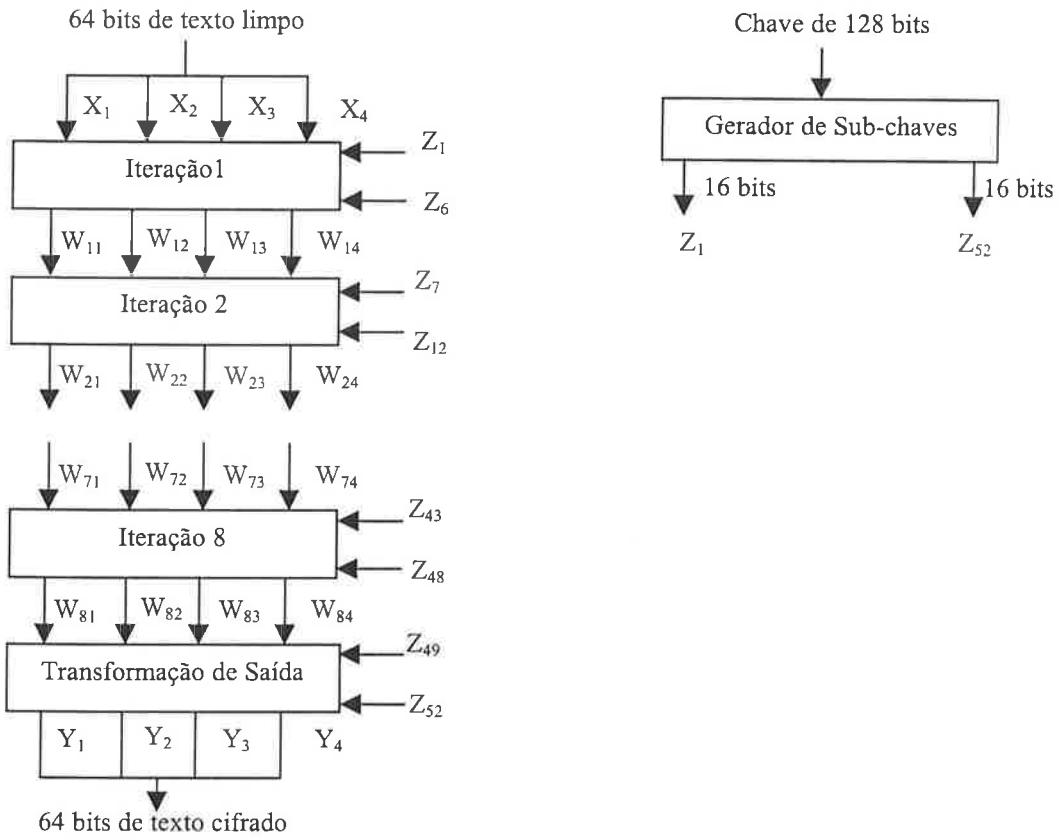

Figura 2.6: Estrutura Geral do algoritmo IDEA

Na Figura 2.6 observa-se que 52 sub-chaves de 16 bits são geradas a partir de uma chave de 128 bits. As primeiras oito sub-chaves (Z_1 a Z_8) são obtidas diretamente da chave original (128 bits), sendo Z_1 os 16 bits mais significativos, Z_2 os próximos 16 bits, e assim por diante até Z_8 com os 16 bits menos significativos. Então, um deslocamento circular à esquerda de 25 posições é aplicado à chave de 128 bits, e as próximas oito sub-chaves são geradas. Este procedimento se repete até que se obtenham as 52 sub-chaves.

Os blocos de iteração do IDEA, bem como o bloco de transformação de saída utiliza as três operações já citadas. O bloco operacional básico do IDEA, que garante a difusão dos bits a serem cifrados, é apresentado na Figura 2.7.

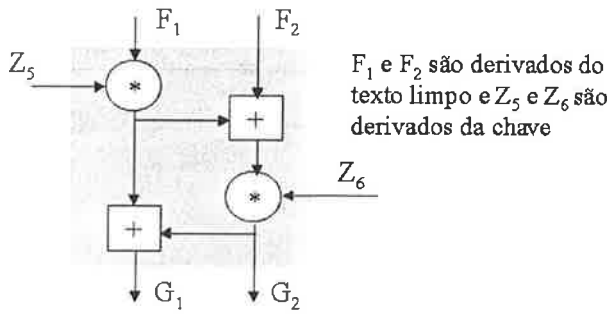

Figura 2.7: Bloco básico de operação do IDEA

O bloco que representa as operações realizadas em uma iteração é composto pelo bloco básico de operação e por blocos da operação ou-exclusivo. A Figura 28 ilustra este bloco para a primeira iteração.

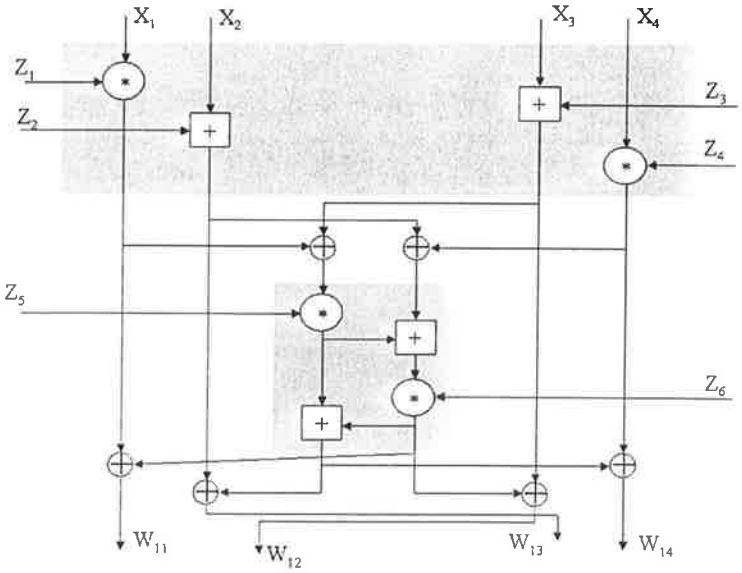

Figura 28: Estrutura de uma iteração

A transformação de saída, que é aplicada após a oitava iteração, é apresentada na Figura 2.9.

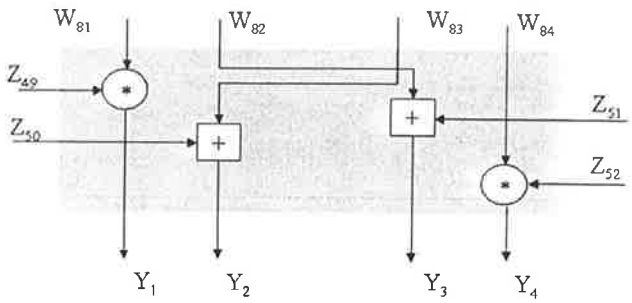

Figura 2.9: Transformação de Saída do IDEA

O processo de decriptografia do IDEA é similar a encriptação, porém as sub-chaves são calculadas de forma diferente da encriptação, de modo a corresponderem às sub-chaves necessárias a decriptografia.

2.3.4 RC4

O RC2 foi o primeiro algoritmo da família RC, desenvolvido por Ronald Rivest e mantido como segredo da RSA Data Security. Em 1996, uma publicação anônima na USENET revelou o algoritmo. A implementação do RC2 vendida pela RSA permite o uso de chaves de tamanho variável entre 1 e 2048 bits.

O RC4 permite que a chave criptográfica possa variar de tamanho entre 1 e 2048 bits, enquanto que o RC5 permite o uso de um tamanho de chave qualquer definida pelo usuário.

2.3.4.1 Funcionamento do RC4

Este algoritmo de criptografia possui tamanho de chave variável e opera no modo cadeia contínua (stream). A sua operação é baseada num conjunto de 8 x 8 S-box (S_0,

$S_1, \ldots, S_{255}$). As suas entradas são permutações dos números 0 à 255 e a permutação é função da chave de tamanho variável.

Os passos de funcionamento do algoritmo são explicados a seguir. Inicialmente considere dois contadores i e j, com valores iniciais iguais a zero. Para cada byte a ser codificado faça:

- $i = (i + 1) \bmod 256$;
- $j = (j + S_i) \bmod 256$;
- Troque S_i por S_j;
- $t = (S_i + S_j) \bmod 256$;
- $k = S_t$.

O byte **k** é utilizado para codificar o byte do texto limpo através de uma operação de Ou–Exclusivo.

Para obter a tabela de substituição é necessário inicializar o conjunto de S-box da seguinte forma: $S_0 = 0$, $S_1 = 1$, ... ,$S_{255} = 255$. Em seguida, atribui-se a um outro vetor de 256 valores de um byte cada ($K_0, K_1, \ldots, K_{255}$) os bits da chave de tamanho variável, repetindo-se a chave até completar os 256 bytes. Depois executa-se os seguintes passos:

- $j = 0$;
- Para $i = 0$ até 255:
    - $j = (j + S_i + K_i) \bmod 256$
    - Troque Si por Sj.

2.3.5 Advanced Encryption Standard (Rijndael)

Este algoritmo foi desenvolvido, e submetido em 2000 ao NIST no concurso de seleção para o padrão AES (*Advanced Encryption Standard*), tendo sido escolhido.

O algoritmo de cifragem por blocos Rijndael é projetado para utilizar operações com conjunto de bytes, permitindo que o tamanho do bloco e da chave sejam escolhidos entre 128, 192 ou 256 bits [Jsavard] [Rijmen]. Este algoritmo é relativamente simples.

O algoritmo possui um número variável de iterações. Sem contar uma iteração extra no final da cifragem com um passo omitido, o número de iterações é:

- 9 se tanto o bloco como a chave possuírem 128 bits de comprimento.
- 11 se o bloco ou a chave possuírem 192 bits de comprimento, e se nenhum deles for maior que isso.
- 13 se o bloco ou a chave possuírem 256 bits.

Para encriptar um bloco de dados, primeiro efetua-se um passo de "Add Round Key" (fazendo um XOR da sub-chave com o bloco), as iterações regulares (que são descritas a seguir), e a iteração final, com o passo de mistura de coluna omitido.

2.3.5.1 As iterações

Cada iteração consta de quatro passos. O primeiro passo é o "**Byte Sub**", onde cada byte é substituído de acordo com uma S-box (Tabela 2.11).

O segundo passo é o Deslocamento de linha (**Shift Row**). Considerando que o bloco é composto de 1 a 16 bytes, estes bytes são organizados em um retângulo, e deslocados conforme a Tabela 2.10.

Tabela 2.10: Deslocamento para blocos de 128 bits

De				para			
1	5	9	13	1	5	9	13
2	6	10	14	6	10	14	2
3	7	11	15	11	15	3	7
4	8	12	16	16	4	8	12

Tabela 2.11: S-box do Rijndael

99	124	119	123	242	107	111	197
48	1	103	43	254	215	171	118
202	130	201	125	250	89	71	240
173	212	162	175	156	164	114	192
183	253	147	38	54	63	247	204
52	165	229	241	113	216	49	21
4	199	35	195	24	150	5	154
7	18	128	226	235	39	178	117
9	131	44	26	27	110	90	160
82	59	214	179	41	227	47	132
83	209	0	237	32	252	177	91
106	203	190	57	74	76	88	207
208	239	170	251	67	77	51	133
69	249	2	127	80	60	159	168
81	163	64	143	146	157	56	245
188	182	218	33	16	255	243	210
205	12	19	236	95	151	68	23
196	167	126	61	100	93	25	115
96	129	79	220	34	42	144	136
70	238	184	20	222	94	11	219
224	50	58	10	73	6	36	92
194	211	172	98	145	149	228	121
231	200	55	109	141	213	78	169
108	86	244	234	101	122	174	8
186	120	37	46	28	166	180	198
232	221	116	31	75	189	139	138
112	62	181	102	72	3	246	14
97	53	87	185	134	193	29	158
225	248	152	17	105	217	142	148
155	30	135	233	206	85	40	223
140	161	137	13	191	230	66	104
65	153	45	15	176	84	187	22

Os blocos de tamanho 192 ou 256 bits são deslocados conforme a Tabela 2.12 e Tabela 2.13, respectivamente:

Tabela 2.12: Deslocamento para blocos de 192 bits

De						para					
1	5	9	13	17	21	1	5	9	13	17	21
2	6	10	14	18	22	6	10	14	18	22	2
3	7	11	15	19	23	11	15	19	23	3	7
4	8	12	16	20	24	16	20	24	4	8	12

Tabela 2.13: Deslocamento para blocos de 256 bits

de								para							
1	5	9	13	17	21	25	29	1	5	9	13	17	21	25	29
2	6	10	14	18	22	26	30	6	10	14	18	22	26	30	2
3	7	11	15	19	23	27	31	15	19	23	27	31	3	7	11
4	8	12	16	20	24	28	32	20	24	28	32	4	8	12	16

Observe na Tabela 2.13 que no caso do bloco de 256 bits, as linhas são deslocadas para os lugares 1, 3 e 4 à esquerda, ao invés de 1, 2 e 3 como nos outros dois tamanhos de bloco.

O terceiro passo é a mistura de colunas (**Mix Column**). Para isso é feita uma multiplicação de matrizes, onde cada coluna é multiplicada pela matriz da Tabela 2.14.

Tabela 2.14: Matriz de Mistura de Colunas

2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

Contudo a multiplicação é feita em $GF(2^8)$. Isto significa que os bytes sendo multiplicados são tratados como polinômios e não como números. Assim, o byte "multiplicado" por 3 é o byte que sofreu XOR com o byte deslocado um bit a esquerda. Se o resultado tem mais de oito bits, os bits extras não são simplesmente descartados, ao invés disso são cancelados através de XOR da *string* binária de 9 bits "100011011" com o resultado (deslocado a direita se necessário). Esta *string* suporta o polinômio gerador (*generating polynomial*) de uma particular $GF(2^8)$. Uma técnica similar é usada em CRC (*Cyclic Redundancy Check*).

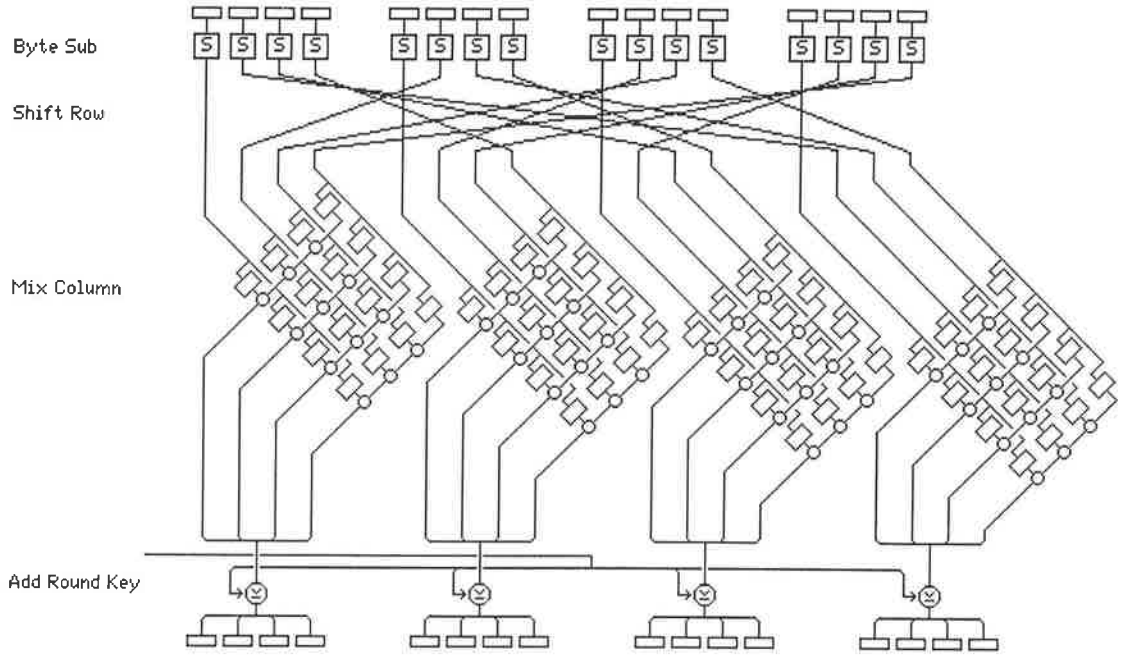

Figura 2.10: Estrutura do Algoritmo Rijndael [Jsavard]

O passo final é a Adição da Chave (**Add Round Key**). Neste passo, é feito simplesmente um XOR com a sub-chave da iteração corrente.

A iteração final extra omite o passo de Mistura de Colunas, mas as outras iterações são as mesmas de uma iteração regular.

A estrutura geral do algoritmo é apresentada na Figura 2.10.

2.3.5.2 A escala das chaves

Para chaves de tamanho 128 e 192 bits, o material da sub-chave, que consiste das chaves de todas as iterações em ordem, é composto da chave original, seguida de extensões. As extensões tem o tamanho da chave original, consistindo de 4 palavras de 1 byte, sendo que cada palavra é o resultado do XOR da palavra de 4 bytes precedentes e a palavra correspondente na extensão anterior ou uma função dela. Para a primeira extensão, a palavra é primeiro rotacionada de um byte à esquerda, depois os seus bytes

são transformados usando a S-box do passo **Byte Sub**,e então efetua-se o XOR entre a constante dependente da iteração e o primeiro byte.

As constantes de iteração são apresentadas na Tabela 2.15.

Tabela 2.15: Constantes de Iteração de Cálculo das sub-chaves

1	2	4	8	16	32	64	128
27	54	108	216	123	246	247	245
241	249	233	201	...			

Estas constantes em binário são as sucessivas potências de 2, na representação $GF(2^8)$. Para chaves de 256 bits, em adição, a S-box do passo **Byte Sub** sozinha é aplicada à palavra da extensão precedente para a quinta palavra na extensão em adição.

2.4 CRIPTOGRAFIA ASSIMÉTRICA

Em 1976, um artigo publicado por Diffie e Hellman [Diffie 76] propunha uma nova abordagem para sistemas de criptografia, e desafiava os outros criptologistas a desenvolverem um algoritmo criptográfico que atendesse a estes requisitos propostos para um sistema de chave pública. A primeira proposta foi publicada em 1978 por Rivest, Shamir e Adleman [Rivest 78].

Os algoritmos de chave pública, ou de criptografia assimétrica, têm como principal característica o uso de funções com elevada complexidade computacional e o uso de chaves de 512 a 1024 bits. O principal algoritmo de Chave Pública é o RSA (Rivest, Shamir e Adleman).

Na criptografia assimétrica, uma chave é utilizada para encriptar e a outra para decriptar uma mensagem. Conforme observa-se na Figura 2.11, que ilustra o processo de

criptografia assimétrica, não existe o problema de manutenção do segredo da chave, pois uma chave é mantida em segredo (chave privada) e a outra é divulgada (chave pública).

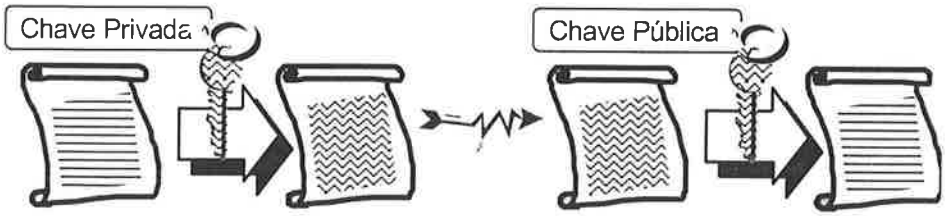

Figura 2.11: Criptografia Assimétrica

As chaves pública e privada são matematicamente relacionadas, sendo que qualquer uma delas pode ser utilizada para encriptar a mensagem, e somente a outra poderá ser utilizada para decriptar a mensagem. Quando uma mensagem é criptografada com a chave privada, a autenticidade da mensagem é garantida. Quando o objetivo a ser alcançado é a confidencialidade, a mensagem deve ser encriptada com a chave pública do destinatário.

2.4.1 RSA

O algoritmo RSA foi desenvolvido por Ron Rivest, Adi Shamir e Len Adleman em 1977 no MIT, e foi publicado em 1978 [Rivest 78].

Este algoritmo apoia-se na dificuldade de fatorar números primos extensos para determinar as chaves pública e privada. As chaves são compostas por um par de números, sendo {e, n} a chave pública e {d, n} a chave privada.

Para obter o par de chaves {e, n} e {d, n} segue-se o seguinte algoritmo:

- Seleciona-se dois números primos grandes (com mais de cem dígitos) **p** e **q**;

- Calcula-se $\mathbf{n} = \mathbf{p} * \mathbf{q}$;
- Calcula-se $\phi(\mathbf{n}) = (\mathbf{p} - 1)*(\mathbf{q} - 1)$;
- Seleciona-se um inteiro **e**, tal que MDC $(\phi(\mathbf{n}), \mathbf{e}) = 1$ e $1 < \mathbf{e} < \phi(\mathbf{n})$;
- Calcula-se $\mathbf{d} = \mathbf{e}^{-1} \bmod \phi(\mathbf{n})$;
- Obtém-se a chave pública $K_U = \{\mathbf{e}, \mathbf{n}\}$ e a chave privada $K_R = \{\mathbf{d}, \mathbf{n}\}$.

Uma vez obtidas as chaves pública e privada, é possível encriptar e decriptar mensagens. Os mecanismos de encriptação (E) e decriptação (D) são apresentados a seguir.

Sendo M o texto limpo, obtém-se o texto criptografado C fazendo:

$$C = E(M) = M^e \bmod n$$

E para obter o texto limpo, tempo o texto criptografado C, é necessário fazer:

$$M = P(C) = C^d \bmod n$$

2.5 Funções de Hash

As funções de hash são mecanismos utilizados para verificar a integridade de uma mensagem. Estas funções são unidirecionais, possuem saída de tamanho fixo (não importa o tamanho da entrada) e devem ser resistentes a colisão (mais de uma entrada podem gerar a mesma saída, mas dada uma saída deve ser impossível encontrar duas entradas que sejam capazes de gerá-la).

Dois algoritmos para cálculo de hash serão abordados: o MD-5 e o SHA-1.

2.5.1 MD-5

O MD-5 foi desenvolvido por Ron Rivest no MIT, e é padronizado segundo a RFC 1321.

Este algoritmo [Stallings 98] produz uma saída de 128 bits a partir de uma entrada de tamanho arbitrário. Esta entrada é processada em blocos de 512 bits. A lógica de funcionamento do MD-5, ilustrada na Figura 2.12, pode ser dividida em cinco passos:

1) **Acrescentar bits de enchimento**: toda mensagem terá bits acrescentados, sendo o tamanho final modulo 512 congruente a 448. Assim, uma mensagem pode ter de 1 a 512 bits de enchimento, que consiste de um bit com valor 1 seguido de quantos bits 0 forem necessários.
2) **Acrescentar tamanho da mensagem**: o tamanho da mensagem antes do enchimento é acrescentado no final da mensagem utilizando uma representação de 64 bits. Se o tamanho da mensagem for superior a 2^{64}, então os 64 bits de mais baixa ordem são utilizados (tamanho do arquivo módulo 64).
3) **Inicialização do *buffer* MD**: um *buffer* de 128 bits é utilizado para manter os resultados intermediários e final da função de hash. Este *buffer* pode ser representado como quatro registradores de 32 bits, que são inicializados com o seguinte valor (em hexadecimal):

   A = 01 23 45 67
   B = 89 AB CD EF
   C = FE DC BA 98
   D = 76 54 32 10

4) **Processamento da mensagem em blocos de 512 bits**: o centro do algoritmo é uma função de compressão **H** que consiste de quatro iterações, que possuem estruturas similares mas utilizam funções lógicas primitivas diferentes.

5) **Saída**: depois de processar todos os blocos de 512 bits, a saída da última iteração fornece o hash de 128 bits.

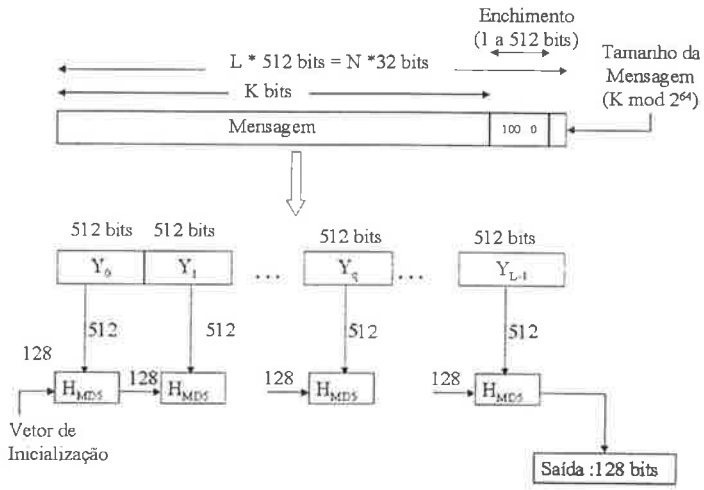

Figura 2.12: Algoritmo do MD-5

2.5.2 SHA-1

O SHA (*Secure Hash Algorithm*) foi desenvolvido pelo *NIST (National Institute of Standards and Technology)* e publicado como norma em 1993. O algoritmo é baseado no MD-4.

O algoritmo aceita como entrada uma mensagem com menos de 2^{64} bits e gera uma saída de 160 bits. A entrada é processada em blocos de 512 bits.

O funcionamento do SHA segue as mesma estrutura do MD-5 [Stallings 98], apresentada na Figura 2.12, porém os *buffers* e a saída possuem tamanho de 160 bits. A sua lógica de funcionamento pode ser dividida em cinco passos:

1) **Acrescentar bits de enchimento**: toda mensagem terá bits acrescentados, sendo o tamanho final modulo 512 congruente a 448. Assim, uma mensagem pode ter

de 1 a 512 bits de enchimento, que consiste de um bit com valor 1 seguido de quantos bits 0 forem necessários.

2) **Acrescentar tamanho da mensagem**: um bloco de 64 bits é acrescentado à mensagem, sendo tratado como um inteiro sem sinal de 64-bit. Este bloco contém o tamanho original da mensagem (antes do enchimento).

3) **Inicialização do *buffer* MD**: um *buffer* de 160 bits é utilizado para manter os resultados intermediários e final da função de hash. Este *buffer* pode ser representado como cinco registradores de 32 bits, que são inicializados com o seguinte valor (em hexadecimal):

A = 67 45 23 01
B = EF CD AB 89
C = 98 BA DC FE
D = 10 32 54 76
E = C3 D2 E1 F0

4) **Processamento da mensagem em blocos de 512 bits**: o centro do algoritmo é uma função de compressão que consiste de quatro iterações, que possuem 20 passos cada. Estas iterações são similares, mas utilizam uma estrutura lógica diferente.

5) **Saída**: depois de processar todos os blocos de 512 bits, a saída da última iteração fornece o hash de 160 bits.

2.6 Esquemas de Distribuição de Chaves

Conforme discutido anteriormente, os algoritmos de criptografia simétrica têm como maior dificuldade no seu uso a distribuição das chaves secretas. Estas devem ser distribuídas através de um canal de comunicação seguro.

Quando se utiliza um algoritmo de chave assimétrica, não é necessário utilizar um canal seguro para transmitir a chave pública. Porém, como garantir que uma chave pública realmente pertence alguém? Ou seja, como garantir a autenticidade de uma chave pública?

Com o intuito de minimizar estas dificuldades alguns esquemas de distribuição de chaves foram propostos, sendo cada um deles com objetivos específicos. Um exemplo que será discutido é o PGP (*Pretty Good Privacy*), cujo principal objetivo é a transferência de emails e aplicações de armazenamento de arquivos. Outro mecanismo que será discutido são os certificados digitais, utilizados para garantir a autenticidade de uma chave pública.

2.6.1 PGP (Pretty Good Privacy)

O PGP (*Pretty Good Privacy*) foi proposto por Phil Zimmerman. Este mecanismo provê serviços de confidencialidade e autenticação que podem ser utilizados em email e em aplicações de armazenamento de arquivos. Para obter estes serviços de segurança, o PGP utiliza criptografia simétrica e assimétrica, e funções de hash.

Como algoritmo de criptografia simétrica [Stallings 98], o PGP pode utilizar o IDEA, 3DES ou o CAST-128. Para criptografia assimétrica utiliza o RSA, o DSS ou Diffie-Hellman. A função de hash utilizada é o SHA-1. O PGP também pode efetuar a compressão de dados para a transmissão ou armazenamento. Neste caso o algoritmo utilizado é o ZIP.

O algoritmo PGP executa as seguintes operações para enviar uma mensagem [Stallings 98]:

1. calcula o hash da mensagem (160 bits) com o algoritmo SHA-1 [Stallings 98];
2. encripta assimetricamente (algoritmo RSA ou DSS) o hash da mensagem utilizando a chave privada do remetente, obtendo a assinatura da mensagem;
3. concatena a mensagem e a assinatura;
4. efetua a compressão da mensagem + assinatura, utilizando o ZIP [Stallings 98];
5. gera uma chave de sessão de 128 bits, que será utilizada apenas para aquela mensagem;
6. encripta simetricamente (algoritmos IDEA, CAST-128 ou 3DES) com a chave de sessão o resultado da compressão da mensagem + assinatura;
7. encripta assimetricamente (algoritmo RSA) a chave de sessão com a chave pública do destinatário;
8. concatena o resultado da criptografia simétrica (mensagem + assinatura) com a chave de sessão encriptada assimetricamente.

Observando as operações realizadas, tem-se que a autenticidade de uma mensagem é obtida através da assinatura digital, ou seja, através da criptografia assimétrica do hash da mensagem (passos 1 e 2). A compressão da mensagem é realizada com o algoritmo ZIP (passo 4). Obtém-se a confidencialidade da mensagem através da criptografia simétrica da mesma, com a chave de sessão gerada exclusivamente para aquela mensagem (passos 5 e 6).

Quando uma mensagem é recebida, o PGP efetua os seguintes passos:

1. separa a mensagem recebida em duas partes: a chave de sessão encriptada assimetricamente e a mensagem encriptada simetricamente;
2. utilizando a chave privada do destinatário, decriptografa a chave de sessão;

3. com a chave de sessão, decriptografa a mensagem comprimida;
4. descomprime o resultado do passo anterior, obtendo a mensagem e a assinatura;
5. utiliza a chave pública do remetente para decriptografar a assinatura, obtendo o hash da mensagem enviada (160 bits);
6. calcula o hash da mensagem recebida (160 bits);
7. compara o hash recebido com o hash calculado para verificar a integridade da mensagem.

A chave de sessão é obtida com o próprio algoritmo do IDEA ou CAST-128 no modo *"Cipher Feedback"*. Utiliza-se como chave um número randômico de 128 bits, e dois blocos de 64 bits como entrada. O resultado desta criptografia é a chave de sessão. Este mecanismo de obtenção da chave de sessão obedece a padronização ANSI X12.17.

2.6.2 Certificados Digitais

Os certificados digitais [Feghhi 99] garantem a autenticidade de uma chave pública. Atualmente, o padrão de certificados utilizado é o ITU-T X.509. As informações que constam deste padrão são [Tremblett 99]:

- **Versão** do formato do certificado.
- **Número serial**: é único e controlado pela Autoridade de Certificação (CA).
- **Identificação do algoritmo** utilizado para assinar o certificado.
- **Emissor** com informações sobre a CA.
- **Período de validade** inicial e final.
- **Sujeito** com informações do usuário.
- **Informação sobre a chave pública**.

- **Assinatura** da CA cobrindo todo o certificado.

2.7 PROTOCOLO SSL

O protocolo SSL (Secure Sockets Layer) foi desenvolvido pela Netscape Communications e submetido ao IETF como Internet *Draft* em fevereiro de 1995. A partir da versão três deste protocolo, foi desenvolvido o protocolo TLS [Dierks 99], cujo objetivo é tornar-se o padrão atual.

Este protocolo é implementado nos sockets, protegendo dados de qualquer aplicação (por exemplo, http, ftp, telnet) que seja construída utilizando-o, ignorando detalhes de implementação das mesmas [Shostack 95]. O HTTP utilizado junto com SSL é chamado HTTPS, e faz uso da porta 443.

SSL provê encriptação de uma sessão; autenticação de um servidor e, opcionalmente, do cliente; e autenticação da mensagem. O SSL é composto por duas camadas (Figura 2.13), a primeira é a *SSL Record Protocol*, e a outra é composta pelo *Handshake*, *Alert* e pela Aplicação, que utilizam a camada anterior [Stallings 98].

O SSL supõe que existe uma estrutura de certificação digital para a autenticação do servidor, não provê renegociação de chaves de sessão, e utiliza um protocolo confiável de camada de transporte.

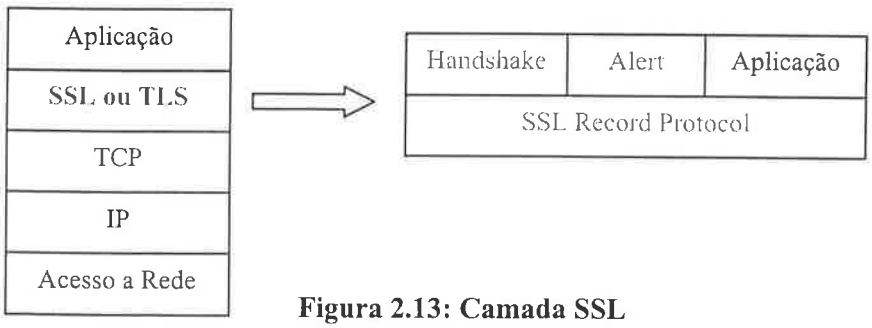

Figura 2.13: Camada SSL

Os algoritmos de criptografia simétrica suportados pelo SSL incluem: RC2 e RC4, ambos com 128 e 40 bits, DES, 3DES e IDEA.

Os itens a seguir discutem as características do protocolo SSL.

2.7.1 SSL Record Protocol [Shostack 95]

Todas as mensagens do protocolo são transmitidas em registros de até 32.767 bytes, sendo que cada mensagem contém 2 ou 3 bytes de cabeçalho. O cabeçalho inclui uma função de segurança de escape (*security escape function*), um indicador de existência de enchimento (*padding*), o tamanho da mensagem, e o tamanho do enchimento. Se houver enchimento, o cabeçalho possui 3 bytes, caso contrário, 2 bytes.

Junto com o registro são enviados três componentes: MAC-DATA, dados atuais e dados de enchimento.

MAC-DATA é o resultado do hash de uma chave, dos dados, do enchimento e do número de seqüência. A escolha do tipo de cifragem implica na escolha da função de hash. A chave utilizada é a chave de escrita do remetente, que é a mesma chave de leitura do destinatário. Se não for definido o tipo de cifragem não há MAC-DATA ou enchimento.

2.7.2 SSL Handshake Protocol

Quando uma estação tenta utilizar uma conexão SSL, ocorre o procedimento de estabelecimento de conexão SSL (*handshake*).

A primeira mensagem enviada é o CLIENT-HELLO, que contém um desafio e o sistema criptográfico que suporta ou deseja. O desafio é um número gerado aleatoriamente pelo cliente, para que, em etapa posterior, ele possa certificar-se de que o

servidor com que foi estabelecida a conexão SSL é o mesmo a quem foi solicitada. O servidor responde com um SERVER-HELLO, que possui um identificador de conexão (*connection id*), o seu certificado digital e o sistema criptográfico que suporta. O cliente irá escolher qual a criptografia usada.

O cliente verifica a chave pública do servidor, e responde com CLIENT-MASTER-KEY, onde está a chave gerada aleatoriamente total ou parcialmente encriptada com a chave pública do servidor. O servidor irá decriptar esta mensagem, e obter a sua chave de escrita (conforme se discute na próxima seção). Então o cliente envia um CLIENT-FINISHED, que inclui o identificador da conexão encriptado com a chave de escrita do cliente.O servidor então envia o SERVER-VERIFY, contendo o desafio recebido do cliente encriptado com a chave de escrita do servidor. Quando o servidor envia o desafio recebido do cliente, ele está demonstrando que recebeu o pedido de conexão daquele cliente e permitindo a verificação da chave de escrita do servidor pelo cliente.

O SSL também permite que seja efetuada a autenticação do cliente. Para isso, o servidor deve enviar um REQUEST-CERTIFICATE, que contém um outro desafio. O cliente responde com um CLIENT-CERTIFICATE, incluindo o tipo de certificado do cliente, o certificado e outros dados. Então o servidor envia a mensagem de SERVER-FINISH.

2.7.3 Chaves no SSL

Existem várias chaves que são utilizadas no protocolo SSL: a chave pública do servidor, uma chave mestra (*master key*), a chave de leitura do cliente (ou de escrita do servidor) e a chave de escrita do cliente (ou chave de leitura do servidor).

As chaves de leitura e escrita do cliente são derivadas da chave mestra, através de um hash seguro da mesma, um caractere ordinal, o desafio e o identificador da conexão. A

chave mestra é enviada encriptada com a chave pública do servidor, sendo reutilizada por várias sessões para calcular as novas chaves de leitura e escrita do cliente.

Dois ou mais conjuntos de hash são gerados, de acordo com o número de bits necessários para a chave. O algoritmo de hash utilizado normalmente é o MD5.

2.7.4 Protocolo de Aplicação SSL

Uma vez terminada a fase de estabelecimento da conexão, inicia-se o protocolo da aplicação.

Uma sessão SSL é terminada quando uma conexão TCP é finalizada.

2.8 MARCA D'ÁGUA DIGITAL

A marca d'água digital evoluiu da antiga técnica de estenografia. Estenografia pode ser definida como esconder mensagens em um texto claro, de forma que fica escondida em uma grande quantidade de dados sem ser necessária criptografia [Bush 99].

Marca d'Água Digital embute informações em dados digitais (o transportador) de maneira secreta e não suspeita. Esta informação embutida é a marca d'água, que deve ser robusta contra distorções, que não degradem significativamente a percepção do sinal transportador [Bush 99].

Uma importante diferença entre estenografia e marca d'água está relacionada com o objetivo do oponente. No primeiro caso, tenta-se revelar a informação que a mensagem carrega; enquanto no segundo, tenta-se remover a marca d'água de modo a violar o direito de cópia, ou fundir a mesma marca d'água depois de alterar a informação, para forjar autenticidade [Voyatzis 99].

2.8.1 Protegendo Imagens Digitais através de Marca d'água [Voyatzis 99]

Para entender a maneira como marcas d'água digitais protegem os direitos autorais, considere o esquema de distribuição através da rede, onde tanto um usuário legítimo como um pirata podem receber dados, e onde tanto um servidor legítimo como um pirata podem distribuir dados. Usuários que não retransmitem dados não violam o direito de cópia, porém piratas violam quando reproduzem ou retransmitem os dados ilegalmente.

Um esquema de proteção deve permitir um método confiável para a busca eficiente de cópias na rede, que originalmente pertenceram a ele.Este esquema também deve prover indicações legais de propriedade.

A marca d'água (ou selo de direitos autorais ou selo invisível) é associada a um número de identificação privado único, chamado chave de marca d'água (*watermark key*). Cada provedor possui uma chave única no esquema de proteção, que consiste das seguintes ações:

- O detentor dos direitos autorais altera dados originais da imagem para introduzir a marca, produzindo uma imagem marcada;
- O detentor dos direitos autorais pode detectar a marca d'água usando a chave única;
- O provedor registra a marca d'água com uma autoridade confiável e garante a sua unicidade;
- O provedor registra as imagens marcadas com uma autoridade confiável, para estabelecer a propriedade das mesmas.

Assim, os provedores podem checar possíveis violações monitorando o tráfego das imagens na rede. O fato de ter registrado as imagens permite demonstrar a propriedade em caso de ambigüidade, caso tenha ocorrido um ataque à marca d'água.

Capítulo 3

O Padrão de Compressão de Vídeo MPEG

Este capítulo explica o padrão de compressão MPEG, discutindo os princípios fundamentais para a compressão de vídeo, e como esta funciona.

O comitê MPEG (*Moving Picture Experts Group*) foi criado pela ISO (*International Organization for Standardization*), em 1988, com o intuito de criar um padrão para armazenamento e transmissão de vídeo, a partir de um consenso entre representantes de diferentes setores (telecomunicações, computação, cinema e TV, universidades) [LeGall 91].

O padrão MPEG-1, criado em 1991 e publicado sob número de referência ISO11172, foi desenvolvido para armazenar sinais digitais de áudio e vídeo colorido com qualidade VCR (Vídeo Cassete Records) em CD-ROM. A taxa de transmissão é de 1,5 Mbps, sendo o canal de áudio de 200 à 250 Kbps e o canal de vídeo de 1,15 à 1,2 Mbps.

O padrão MPEG-2 [Mitchel 96] apresenta as mesmas características do padrão MPEG-1, com algumas melhorias que permitem a sua operação em ambientes suscetíveis a erros. MPEG-2 mantém a mesma sintaxe de vídeo, porém acrescenta algumas extensões.

O padrão MPEG trata separadamente vídeo e áudio, especificando como estes sinais são associados e sincronizados, possuindo assim três partes: a camada de sistema, a camada de vídeo e a camada de áudio. Estas camadas serão discutidas em itens posteriores.

Como o volume de dados de vídeo é muito grande (1 segundo de vídeo na resolução de 640 x 480 resulta em 27 MB), torna-se necessário utilizar técnicas de compressão [Silveira 98].

3.1 Técnicas de Compressão

Existem dois tipos principais de compressão: a compressão livre de perdas e a compressão com perdas. A informação submetida a um processo de compressão livre de perdas não sofre nenhuma alteração em relação à informação original após a sua

descompressão. No caso da compressão com perdas (ou compressão irreversível), a informação obtida após a descompressão é diferente da informação original.

A codificação MPEG é baseada na percepção do olho humano, então é possível utilizar a compressão irreversível sem que o observador note qualquer diferença.

As técnicas de compressão, segundo [Silveira 98], podem ser classificadas em:

- Codificação por entropia;
- Codificação dependente da fonte.

Na codificação por entropia a natureza dos dados a serem comprimidos não é considerada, como ocorre na Supressão de Caracteres Repetidos (usada no padrão MPEG) e na Codificação Estatística.

No caso da codificação dependente da fonte, a compressão aproveita as características específicas do tipo de informação. Por exemplo a Compressão por Transformadas, ou a Compressão Diferencial ou a Quantização Diferencial.

A compressão de vídeo consiste em eliminar as informações redundantes (correlatas). Estas correlações podem aparecer de duas formas:

- correlação espacial
- correlação temporal.

A correlação espacial é observada em uma mesma imagem, ou seja, são as informações redundantes, como por exemplo a cor de fundo de uma cena. Para eliminar a correlação espacial, o padrão MPEG utiliza a Transformada Discreta de Coseno (DCT), seguida da quantização dos coeficientes obtidos.

Já a correlação temporal é observada em dois quadros consecutivos; por exemplo a primeira cena mostra uma sala com móveis e uma pessoa, enquanto na segunda cena aparece a mesma sala, porém a pessoa mudou de lugar. Para eliminar a correlação

temporal, o padrão MPEG utiliza o processo chamado de Compensação de Movimento, que é o emprego da técnica DPCM, codificando apenas as diferenças encontradas entre os quadros.

A Tabela 3.1 ilustra o ganho obtido utilizando técnicas de compressão de vídeo para uma resolução de 640 x 480 [Silveira 98].

Tabela 3.1: Ganho obtido com compressão

	1 segundo	1 minuto	1 hora
1 : 1	27 Mb	1,6 Gb	97 Gb
3: 1	9 Mb	540 Mb	32 Gb
100 : 1	270 Kb	16 Mb	97 Mb

Os algoritmos de compressão permitem definir o grau de compressão desejado.

3.2 Sensibilidade do Olho Humano

A resposta do olho humano em relação à sensibilidade luminosa varia com a luminância e a crominância. A luminância é a resposta cumulativa dos olhos a todos os comprimentos de onda contidos em uma fonte luminosa; seus componentes são o contraste (impressão de branco ou preto) e o brilho (impressão de escuro ou brilhante). A crominância é a resposta dos olhos às cores, ou seja, a cada um dos comprimentos de onda específicos. O olho humano é mais sensível ao comprimento de onda correspondente ao verde, seguido do vermelho e então o azul, sendo mais sensível a luminância que a crominância [Silveira 98] [Mitchel 96].

De acordo com as características do olho humano, os três sinais de cores utilizados pelos sistemas de vídeo (RGB ou CMY) são convertidos em dois sinais de crominância e um sinal de luminância. O padrão MPEG-1 define um formato para sub-amostragem

(SIF – *Standard Interchange Format*) que determina o número de amostras por linha e o número de linhas por quadro para a luminância e para a crominância:

- Luminância: 352 amostras por linha; 240 linhas por quadro para o NTSC e 288 linhas por quadro para o PAL/SECAM.
- Crominância: 176 amostras por linha; 120 linhas por quadro para o NTSC e 144 linhas por quadro para o PAL/SECAM.

O uso do SIF permite uma redução de 50% na taxa de transmissão sem degradação da qualidade da imagem.

3.3 CAMADA DE SISTEMA

A camada de sistema MPEG é responsável pelas informações de tempo para a reprodução do vídeo MPEG-1. Ou seja, especifica como os fluxos de informação de áudio e de vídeo devem ser multiplexados em um único canal de transmissão ou no local de armazenamento. Além disso, deve sincronizar estes dois fluxos de dados.

3.4 CAMADA DE VÍDEO

A camada de vídeo MPEG é dividida em seis camadas [Mitchel 96]: Camada de Seqüência de Vídeo, Camada de Grupos de Imagens (GOP), Camada de Imagem, Camada de *slice*, Camada de Macroblocos e Camada de Blocos, conforme observa-se na Figura 3.1.

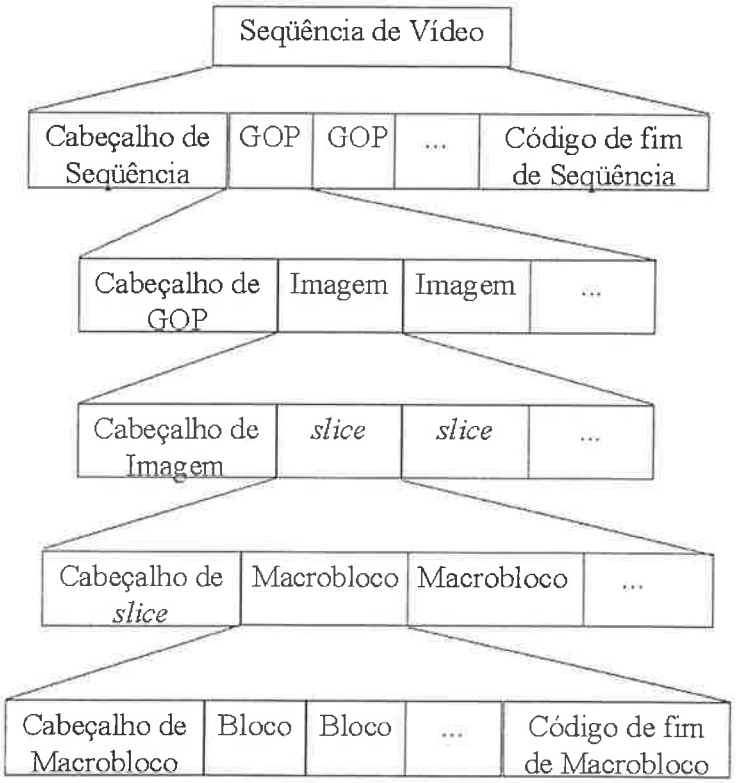

Figura 3.1: Estrutura da Camada de Vídeo MPEG

Cada uma destas camadas é identificada pelo seu cabeçalho, cujos valores podem ser observados na Tabela 3.2.

Tabela 3.2: Códigos de início de vídeos MPEG

Nome do Código de Início	Valor em Hexadecimal
Extension_start code	000001B5
Group_start_code	000001B8
Picture_start_code	00000100
Reservado	000001B0
Reservado	000001B1
Reservado	000001B6
Sequence_end_code	000001B7
Sequence_error_code	000001B4
Sequence_header_code	000001B3
Slice_start_code 1	00000101
...	...
Slice_start_code 175	000001AF

Em MPEG-1 a imagem é dividida em blocos de 16 x 16 amostras para luminância, e blocos de 8 x 8 amostras para cada sinal de crominância. Um macrobloco é composto

por um bloco de luminância (4 x (8 x 8) amostras) e dois blocos de crominância (1x (8 x 8) + 1x (8 x 8) amostras). O vetor de movimento indica a translação espacial de um bloco para o outro, sendo utilizado na Compensação de Movimento para eliminar a correlação temporal.

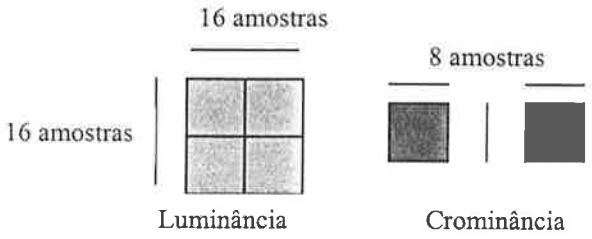

Figura 3. 2: Constituição do Macrobloco MPEG

As cadeias de vídeo podem ter três tipos de quadros:

- quadro I (*intra-frame*): é um quadro codificado somente com informações da imagem, não dependendo de qualquer quadro passado ou futuro;
- quadro P (*forward predicted frame*): este quadro é codificado relativamente ao quadro de referência precedente mais próximo (quadro I ou quadro P);
- quadro B (*bi-directional predicted frame*): sua codificação é feita relativa ao quadro de referência precedente mais próximo (quadros I ou P), ou ao quadro de referência sucessivo mais próximo, ou a ambos.

Uma seqüência típica de quadros MPEG é apresentada na Figura 3. 3 , onde a dependência entre os quadros I, P e B pode ser observada. Note que se um quadro I não é decodificado corretamente, todos os quadros seguintes apresentarão erros, até a decodificação do próximo quadro I.

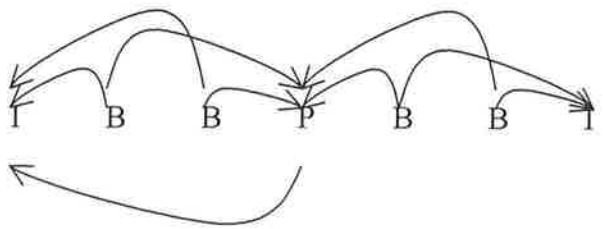

Figura 3. 3: Interdependência de Quadros para uma Seqüência MPEG

3.5 Processo de Compressão MPEG

O processo de compressão MPEG segue os seguintes passos:

- processo de identificação dos quadros;
- preparação dos blocos de dados;
- codificação: transformada discreta de coseno (DTC), quantização, supressão de seqüências repetidas (aplicada em zig-zag) e codificação de Huffman.

Coeficiente DCT c(i j)

168	45	7	3	2	2	1	1
67	32	3	3	2	1	1	1
12	5	5	5	2	1	2	1
5	5	2	2	1	1	1	1
3	2	2	1	1	1	1	1
2	2	2	1	1	1	1	0
1	1	1	1	2	1	1	0
1	2	1	1	1	1	0	0

Tabela de Quantização Q(i j)

1	1	1	1	1	4	8	16
1	1	1	1	4	4	8	16
1	1	1	2	4	4	8	16
2	8	8	8	8	16	16	16
4	8	8	8	8	16	16	32
4	8	8	8	16	16	16	32
4	8	8	8	16	16	32	32
8	8	8	16	16	32	32	64

Coeficiente Quantizado q(i j)

168	45	7	3	2	0	0	0
67	20	3	3	2	0	0	0
12	5	5	2	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

$q(i\ j) = c(i\ j) / Q\ (i\ j)$

Figura 3.4: Exemplo de Quantização

Com a Transformada Discreta de Coseno (DCT), os blocos são transformados do domínio espacial para o domínio das freqüências, o que irá facilitar a compressão dos dados. Inicialmente, cada bloco é composto por 64 valores que representam a amplitude do sinal amostrado para o componente considerado. Após a transformada DCT, estes 64 valores representam as freqüências dos sinais amostrados.

A quantização consiste em dividir cada um dos coeficientes obtidos através da Transformada Discreta de Coseno por valores da Tabela de Quantização, de modo a obter a Tabela de Coeficientes Quantizados. Os valores da Tabela de Quantização variam de 1 a 255, e determinam quais coeficientes serão descartados após a quantização. Quanto maior o valor do coeficiente, maior será a taxa de compressão, o que reduz a fidelidade da imagem resultante.

Após a quantização muitos coeficientes são nulos, conforme observa-se na Figura 3.4. Este fato permite o emprego da técnica de supressão de caracteres repetidos ou nulos, que é aplicada em zig-zag para aumentar a probabilidade de encontrar seqüências repetidas, conforme observa-se na Figura 3.5. A Técnica de Supressão de Caracteres Repetidos ou Nulos consiste em, detectar as seqüências repetidas ou nulas, e substituí-las por um código seguido do número de vezes que o caracter ocorreu.

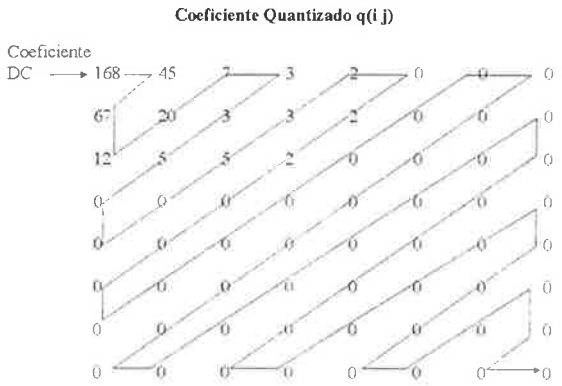

Figura 3.5:Aplicação em Zig-zag da técnica de supressão de caracteres

O próximo passo é a aplicação da codificação de Huffman. Código de Huffman ou Compressão Estatística consiste em identificar padrões comuns e, a partir da freqüência de ocorrência de cada padrão, criar uma palavra código para este. Estas palavras-código fazem parte de um dicionário construído para a codificação.

Capítulo 4

MECANISMOS DE CRIPTOGRAFIA E DISTRIBUIÇÃO DE VÍDEO MPEG

Este capítulo discute os trabalhos correlatos realizados em outros centros de pesquisa, sob o aspecto da criptografia de vídeo e de distribuição.

Diversos trabalhos vêm sendo realizados em outros centros de pesquisa no sentido de desenvolver mecanismos de segurança para a transmissão de vídeo comprimido no padrão MPEG, mais especificamente MPEG-1. Estes trabalhos abordam de maneira diferente a implementação destes mecanismos, de modo que apresentam vantagens e desvantagens em relação aos demais. Dentre os mecanismos de criptografia de vídeo, podem ser citados: Criptografia Simples, Criptografia Seletiva, Algoritmo de Permutação Zig-Zag e VEA (*Video Encryption Algorithm*).

Os mecanismos citados tratam apenas da criptografia do vídeo MPEG, não abordando a camada de áudio. Para a criptografia do áudio seria necessário estudar e especificar outros mecanismos que tratem da camada de áudio MPEG.

Uma vez abordadas as características destes trabalhos, é feita uma comparação dos resultados obtidos com cada um dos codificadores / decodificadores propostos, ou seja, verifica-se a confidencialidade obtida em cada um deles, a velocidade de criptografia e se existe variação do tamanho dos arquivos.

Além destes trabalhos sobre a criptografia de vídeo, outros trabalhos consideram situações específicas de distribuição, como é o caso do vídeo sob demanda. O artigo de Griwodz [Griwodz 98] apresenta uma solução para ambientes de vídeo sob demanda em ambientes comerciais, porém não confidenciais, que também será discutida neste capítulo.

4.1 Criptografia de Vídeo MPEG

4.1.1 Criptografia Pura ou Simples

No mecanismo de criptografia pura (*Naive Encryption*), os vídeos MPEG são tratados como dados, ou seja, não são consideradas as características da codificação MPEG. O

arquivo MPEG é criptografado utilizando um algoritmo de criptografia convencional, como o DES ou IDEA, e então o arquivo MPEG é enviado. Após receber o arquivo MPEG, este é decriptografado e o arquivo obtido pode ser assistido. Observe que o arquivo MPEG estará desprotegido (decriptografado) no disco do usuário.

Este mecanismo proporciona um nível de segurança alto durante a transmissão, já que a dificuldade em quebrar o algoritmo é aquela apresentada ao tentar quebrar o DES ou o IDEA. Não altera o tamanho do arquivo após a criptografia, mas é muito lento [Qiao 98], além de ocorrer aumento no atraso durante a decodificação da cadeia de vídeo [Spanos 96], tornando inviável utilizá-lo em aplicações de tempo real.

4.1.2 Criptografia Seletiva

A criptografia seletiva procura utilizar as características das cadeias de vídeo MPEG para diminuir a quantidade de informações criptografadas. Os quadros I são aqueles que carregam mais informações, enquanto os quadros P e B representam variações da imagem em quadros I adjacentes. Assim, se os quadros I forem criptografados será difícil compreender o conteúdo de uma cadeia de vídeo. Alguns mecanismos também permitem criptografar os quadros I e P, ou todos os quadros (I, P e B).

Porém, quando criptografa-se somente os quadros I e executa-se o arquivo em *player* MPEG convencional que suporte erros, ainda é possível perceber o conteúdo do vídeo [Agi 96]. Uma solução proposta [Agi 96] é aumentar a freqüência dos quadros I, mas isto diminui a compressão, aumentando o tamanho do arquivo.

Dentre os diversos trabalhos que implementam este tipo de mecanismo podem ser citados: SE_MPEG, Aegis e SECMPEG.

4.1.2.1 SE_MPEG (Security Enhanced MPEG Player)

Este projeto [Li 96] propõe um esquema de criptografia e decriptografia em tempo real de cadeias de vídeo. Para tanto, os autores propõem um *player* MPEG seguro, que implementa proteção através de um esquema de criptografia baseado nas seqüências de codificação MPEG. A criptografia pode ser feita somente nos quadros I, nos quadros I e P, ou em todos os quadros I, P e B. Um aumento na proteção implica em maior *overhead*, pois mais criptografia é feita.

O codificador e o decodificador de vídeo MPEG propostos são baseados em um codificador MPEG feito em Berkeley, e no PGP proposto por Zimmerman, sendo os quadros criptografados com o algoritmo de criptografia convencional IDEA e a chave distribuída utilizando criptografia assimétrica (RSA).

Os vídeos MPEG são encriptados quadro a quadro, já que os quadros são unidades independentes em um arquivo MPEG. Como o PGP trabalha com arquivos completos, foram implementadas funções para que fosse possível trabalhar com conjuntos de quadros (*buffered frames*). Estas funções criptografam e decriptografam o conteúdo de um *buffer*, tanto utilizando o algoritmo IDEA como utilizando o algoritmo RSA. Também foi implementada uma função que gera a chave de sessão.

Para encriptar o arquivo MPEG quadro a quadro, é necessário analisar o arquivo e identificar todos os cabeçalhos, e criptografar somente os quadros, mantendo as seqüências de início e fim de código do arquivo MPEG. Assim, é possível analisar um arquivo MPEG e identificar os seus cabeçalhos mesmo com os quadros encriptados.

Para melhorar o nível de proteção obtido com a criptografia, o codificador possibilita o uso de uma hierarquia de proteção. Esta hierarquia de proteção é obtida quando escolhe-se entre criptografar somente quadros I, quadros I e P, ou os quadros I, P e B.

Entretanto, a quantidade de informação possível de se recuperar a partir dos quadros P e B sem o quadro I associado depende do vídeo codificado. O padrão de seqüências MPEG que será utilizado na codificação não depende do conteúdo do vídeo. Se as transições de cena estiverem em quadros P ou B, os macroblocos destes quadros serão codificados como os macroblocos dos quadros I, contendo informação suficiente para que seja possível inferir o conteúdo do vídeo se somente os quadros I forem codificados. No filme *Star Wars*, as transições ocorrem em média a cada sete segundos [Li 96]. Se fossem criptografados somente os quadros I neste filme, seria possível decodificar diversos quadros a cada sete segundos. Por isso os outros esquemas da hierarquia de proteção são necessários, apesar de aumentar o *overhead* da decriptografia.

Tabela 4.1: Arquivos utilizados na avaliação de desempenho

Nome do Arquivo	Tamanho do Arquivo (bytes)	Número de Quadros	Tamanho do Quadro
renat.mpg	555.297	351	160 x 120
orincsa.mpg	1.708.105	850	320 x 249

Foram feitos diversos testes de desempenho do SE_MPEG, comparando-o com o codificador original [Li 96]. Estes testes foram feitos em uma Sun SPARC Station 10, rodando Solaris 2.4 com 32MB de memória. Os arquivos MPEG estavam armazenados localmente no disco. Foram utilizados dois arquivos diferentes com quadros de vídeo de tamanhos diferentes, conforme observa-se na Tabela 4.1.

A Tabela 4. 2 compara o desempenho obtido com o *player* MPEG convencional e com o SE_MPEG para dois arquivos citados. Estes dados representam a média de dez execuções.

Tabela 4. 2: Comparação da velocidade dos *players* MPEG convencional e seguro

Nome do Arquivo	Não encriptado	Quadros I, P e B encriptados	Quadros I e P encriptados	Quadros I encriptados
Renat.mpg	29.87 fps	23.10 fps	24.10 fps	25.28 fps
Orincsa.mpg	12.36 fps	10.68 fps	10.87 fps	11.03 fps

A Tabela 4. 3 apresenta estes resultados em termos de degradação da performance.

Tabela 4. 3: Degradação Percentual com o SE_MPEG

Nome do Arquivo	Quadros I, P e B encriptados	Quadros I e P encriptados	Quadros I encriptados
renat.mpg	22.6%	19.3%	15.4%
orincsa.mpg	13.6%	12.1%	10.7%

Este projeto considera ainda quando a criptografia deve ser efetuada, se o arquivo deve ser encriptado antes da transmissão, ou durante a transmissão. No caso da encriptação do arquivo todo antes da transmissão, ocorre uma latência inicial devido à criptografia. Fazendo a criptografia durante a transmissão esta latência é removida, mas um tempo extra de CPU é necessário para enviar o quadro pela rede. A Tabela 4. 4 mostra o tempo necessário para encriptar quadros durante a transmissão.

Tabela 4. 4: Tempo de encriptação dos Quadros durante a transmissão

Tamanho do Quadro	Quadro B (ms)	Quadro P (ms)	Quadro I (ms)
160 x 120	2 - 4	12 – 15	18 – 22
320 x 240	2 - 4	12 – 15	36 – 44

4.1.2.2 Aegis

O Aegis [Spanos 95], [Spanos 96] propõe um esquema de criptografia onde somente os quadros I de todos os grupos de quadros em uma cadeia de vídeo MPEG são encriptados. A escolha de encriptar somente os quadros I foi feita baseada no fato de que os quadros I carregam grande quantidade de informações, enquanto os quadros P e B representam variações da imagem em quadros I adjacentes.

Além de encriptar os quadros I, Aegis também encripta o cabeçalho de seqüência de vídeo. O cabeçalho de seqüência de vídeo contém os parâmetros de inicialização da decodificação, como altura e largura do quadro, taxa de quadros, taxa de bits e tamanho do *buffer*. Encriptar o cabeçalho dissimula a identidade de uma cadeia MPEG, fazendo com que esta torne-se irreconhecível. O código de seqüência final também é encriptado no Aegis, dificultando ainda mais o reconhecimento de uma cadeia MPEG.

Este esquema de criptografia considera que o nível de segurança obtido é alto, pois o intruso terá dois desafios:

- o de identificar a cadeia MPEG, e consequentemente os grupos de quadros, quadros e os quadros I encriptados;
- o de decriptar os quadros I identificados.

Aegis atua como um filtro de cadeias MPEG entre o codificador / decodificador e o meio de transmissão. Foi decidido utilizar a estrutura de um filtro, ao invés de integrar o mecanismo de segurança ao codificador / decodificador, para manter independência entre a implementação do mecanismo de segurança e a padronização MPEG (MPEG-1 e 2 mantém o conceito de grupo de quadros).

Devido à estrutura de camadas da padronização MPEG e aos cabeçalhos destas camadas, é possível identificar os segmentos de interesse dentro da seqüência de vídeo. Um problema que aparece quando parte da cadeia MPEG é encriptada é como identificar as partes encriptadas durante o processo de decriptografia. Para facilitar esta identificação, o processo de criptografia do Aegis acrescenta *flag-bytes*, usando a técnica de *bit-stuffing*. Estes *flag-bytes*, que são mantidos em segredo, são inseridos tanto no começo como no final de uma seção encriptada. A técnica de *bit-stuffing*

considera se o *flag-byte* aparece ou não na seção encriptada, e em caso positivo acrescenta um digito binário **0** seguido dos sete bits que correspondem ao *flag*.

No processo de criptografia, Aegis utiliza o algoritmo DES no modo CBC (Cipher Block Chaining Mode). Este modo do DES é considerado o mais seguro, e é o menos sensível a erros de bits.

Devido às diferenças nos cabeçalhos, um *player* MPEG convencional não é capaz de decodificar corretamente uma seqüência de vídeo codificada pelo Aegis, sendo que as imagens de fundo aparecem borradas e sem nitidez [Spanos 96].

Segundo uma simulação feita pelos autores para o mecanismo, o desempenho do Aegis é bastante próximo ao de um sistema sem criptografia, já que este é capaz de manter constante o atraso devido à criptografia. Um sistema com criptografia completa não mantém o atraso constante, pois demora a processar as informações de entrada, acumulando atrasos. Apesar de os atrasos obtidos com Aegis serem muito próximos daqueles obtidos com um *player* convencional, o nível de segurança é aceitável, mas não é adequado para aplicações sensíveis [Agi 96], já que com players com suporte a erros ainda é possível identificar a imagem criptografada.

4.1.2.3 SECMPEG

Este projeto [Meyer 95] propõe uma variação do padrão MPEG para a transmissão segura de vídeo, que incorpora criptografia seletiva e informações adicionais no cabeçalho.

O SEC_MPEG implementa quatro níveis de segurança:

- 1º nível: encripta todos os cabeçalhos;
- 2º nível: encripta todos os cabeçalhos mais os coeficientes DC e os termos AC dos blocos I;

- 3° nível: encripta os quadros I e os blocos I dos quadros P e B;
- 4° nível: encripta todos os campos.

O SECMPEG utiliza uma cadeia de vídeo com informações adicionais de segurança, conforme observa-se na Figura 4.1.

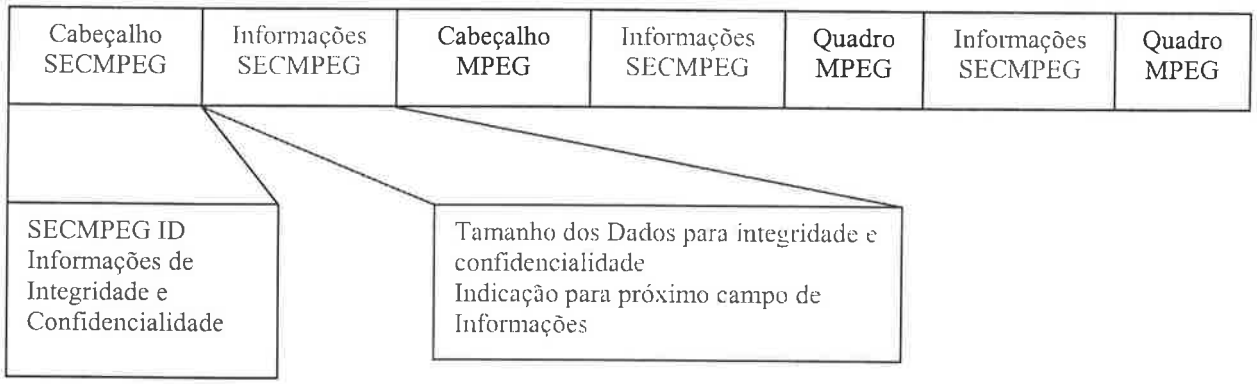

Figura 4.1: Cadeia de Vídeo SECMPEG

SECMPEG utiliza o algoritmo DES para a criptografia das informações. Para a verificação da integridade do conteúdo é feito um cálculo de CRC. A

Figura 4. 2 apresenta a estrutura do SECMPEG, bem como os módulos existentes.

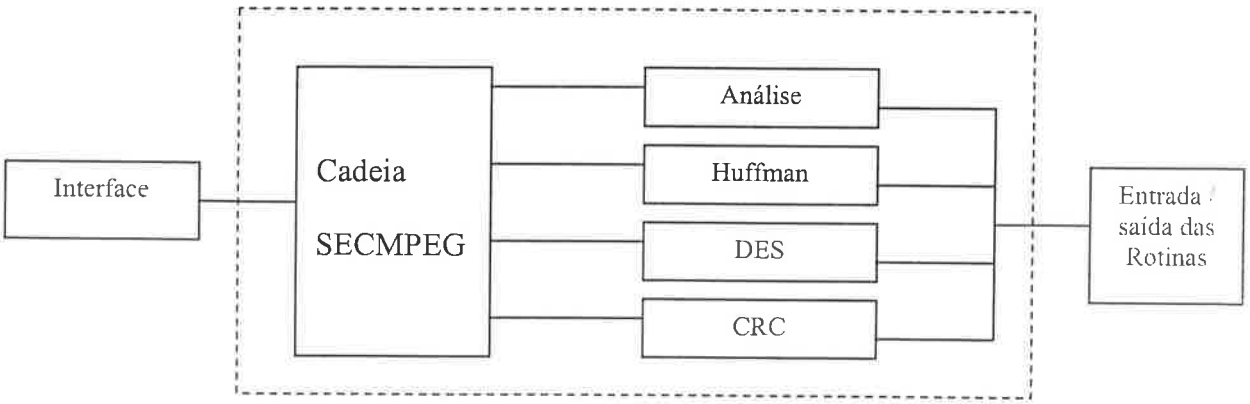

Figura 4. 2: Módulos do SECMPEG

4.1.3 Algoritmo de Permutação em Zig-Zag

O mecanismo proposto associa a criptografia à compressão da imagem e do vídeo (JPEG e MPEG) [Tang 96]. Este mecanismo de criptografia utiliza uma lista randômica de permutação para fazer o mapeamento dos blocos 8 x 8 no vetor 1 x 64, ao invés de fazê-lo em zig-zag (que é utilizado pelo padrão MPEG).

A partir de quatro experimentos com a ordem dos coeficientes DC e AC no vetor 1 x 64 concluiu-se que: a posição do coeficiente DC é importante; a imagem ainda é compreensível se o coeficiente DC for zero e os coeficientes AC forem permutados em zig-zag; o último coeficiente AC pode ser mudado para zero através da matriz de quantização sem prejuízo à qualidade da imagem.

Outro mecanismo estudado é a divisão do coeficiente DC ($d_0d_1d_2d_3d_4d_5d_6d_7$) em duas partes com quatro bits, sendo uma delas colocada no lugar do coeficiente DC ($d_0d_1d_2d_3$) e outra no lugar do último coeficiente AC ($d_4d_5d_6d_7$). Assim, a codificação / criptografia utiliza este mecanismo, e em seguida aplica a lista de permutação ao vetor 1 x 64, ao invés da permutação em zig-zag.

Este algoritmo aumenta consideravelmente o tamanho das cadeias de vídeo, já que, quando altera-se a ordem do vetor 1x64, perde-se capacidade de compressão, uma vez que esta é maximizada quando aplica-se a lista de permutação em zig-zag, o que aumenta o número de símbolos repetidos para aplicação do Código de Huffman.

Este mecanismo de criptografia é vulnerável ao tipo de ataque de texto limpo conhecido. Por este motivo são realizadas algumas modificações: aplica-se uma função

de criptografia ao coeficiente DC, e são geradas duas listas de permutação, que são aplicadas segundo um sorteio (*flip coin*).

A Tabela 4. 5 mostra o desempenho do algoritmo para dois vídeos: flower.mpg e tennis.mpg.

Tabela 4. 5: Desempenho do Algoritmo de Permutação Zig-Zag

Tempo para codificação	Algoritmo Original	Algoritmo de Permutação Zig-Zag
Vídeo		
flower.mpg	37.985 seg	37.969 seg
tennis.mpg	14.213 seg	14.403 seg

4.1.4 VEA (Video Encryption Algorithm)

O algoritmo *Video Encryption Algorithm* (VEA) utiliza o comportamento estatístico do vídeo comprimido [Qiao 97]. A análise estatística feita com as cadeias de vídeo MPEG trata as cadeias de vídeo como *bytes*. A primeira observação feita é que a freqüência de ocorrência dos valores destes *bytes* (0 a 255) é praticamente a mesma para qualquer valor do *byte*. Analisando esta distribuição para meio *byte*, em qualquer posição da cadeia, não ocorre nenhuma alteração na distribuição de freqüência. Ainda observa-se que diferentes cadeias MPEG possuem o mesmo comportamento.

Outro estudo realizado é relacionado a freqüência de ocorrência de digramas (pares de números adjacentes). Esta análise divide o quadro I em porções, e então verifica-se o número de ocorrências do par de maior freqüência na porção. Se um destes pares se repetir, então um digrama se repetiu. Observou-se que não há nenhum padrão de *byte* repetido com porções de 1/16 de um quadro I. Esta informação é relevante para o desenvolvimento do algoritmo VEA.

O algoritmo VEA assume que a porção do quadro I tem a seguinte forma: $a_1a_2...a_{2n-1}a_{2n}$. Separa-se os *bytes* pares dos *bytes* ímpares, obtendo duas novas cadeias (lista par e lista

ímpar). Aplica-se a função Ou-exclusivo entre as listas par e ímpar, obtendo-se $c_1c_2...c_n$. Escolhe uma função de criptografia E, e aplica-se a lista par. O texto criptografado é $c_1c_2...c_n$ E $(a_2a_4...a_{2n})$.

Os autores fazem uma comparação entre o Aegis e o VEA, sendo que o VEA proporciona um ganho de 47% em relação à criptografia com o IDEA no tempo total de criptografia [Qiao 97]. Segundo [Qiao 97], parte deste aumento no tempo total com o uso do IDEA deve-se ao fato de ter sido utilizada a implementação do PGP. Observe os valores obtidos na Tabela 4. 6.

Tabela 4. 6: Desempenho do VEA

MPEG Vídeo	Tempo total - VEA	Tempo por quadro - VEA	Ganho: VEA x IDEA	Tempo total - IDEA	Tempo por quadro – IDEA
Bus.mpg	3.9779 s	26.88 ms	48.07%	7.6609 s	51.76 ms
Flower.mpg	3.8208 s	25.82 ms	47.40%	7.2645 s	49.08 ms
Klara1.mpg	29.7195 s	29.84 ms	48.65%	57.8817 s	58.11 ms
Space.mpg	5.0372 s	7.79 ms	47.56%	9.6061 s	14.85 ms
Twister.mpg	38.3258 s	31.01 ms	47.26%	72.6637 s	58.79 ms

4.1.5 Permutação Pura

Os resultados estatísticos que permitiram o desenvolvimento do VEA, também validam o uso da Permutação Simples. A permutação simples embaralha os *bytes* das cadeias por permutação. A cardinalidade da chave de permutação depende do nível de segurança desejado, podendo variar de 64 números até 1/8 de um quadro I [Qiao 98].

4.1.6 Comparação Entre Os Mecanismos Descritos

Alguns dos mecanismos de criptografia descritos podem alterar o tamanho da cadeia de vídeo MPEG, como o de Permutação em Zig-Zag. O nível de segurança de cada um dos

mecanismos é diferente, além do tempo necessário para a criptografia [Qiao 98] (ou velocidade de criptografia).

Assim, pode-se comparar estes algoritmos segundo três parâmetros: velocidade de criptografia, nível de segurança e tamanho das cadeias de vídeo. A Tabela 4.7 mostra os resultados desta comparação obtidos por [Qiao 98].

Tabela 4.7: Comparação dos Algoritmos de Criptografia MPEG

Algoritmo	Nível de Segurança	Velocidade	Tamanho das Cadeias
Criptografia Pura	Alto	Lento	Sem alterações
Criptografia Seletiva	Moderado	Rápido	Aumenta
Permutação Zig-Zag	Muito baixo	Muito rápido	Aumenta muito
VEA	Alto	Rápido	Sem alterações
Permutação Pura	Baixo	Super rápido	Sem alterações

4.2 Mecanismos de Distribuição de Vídeo

Considerando os requisitos de proteção de direito autoral para a distribuição de vídeo, alguns trabalhos vem sendo desenvolvidos. A maior parte destes trabalhos preocupa-se em determinar um modelo de marca d'água, que possa identificar a autoria. Porém outras soluções também são encontradas.

4.2.1 Proteção de Vídeo sob Demanda

[Griwodz 98] apresenta uma maneira de proteger a distribuição de Vídeo sob Demanda em redes de WAN, utilizando armazenamento do vídeo em servidores distribuídos pela rede (*caching*) e *prefetching* para distribuir a maior parte dos dados de vídeo. Parte dos dados do vídeo original é corrompida, e transmitida à baixa taxa num canal seguro

ponto-a-ponto para o cliente. Desta forma, assegura-se que a reconstrução e a revenda de vídeos pelo roubo de dados não é possível.

São apresentadas duas maneiras de combinar esta proposta com meios de identificar revendedores de vídeo. A primeira faz uso de marcas d'água específicas para cada cliente, que são inseridas durante a recombinação dos dados recebidos do servidor de intermediário (*cache*) com aqueles recebidos pela conexão ponto-a-ponto segura (parte *unicast*). O outro modo apresentado mantém alguns bits errados para posterior identificação.

Em conjunto com uma destas duas maneiras de impedir a revenda, o esquema de criptografia parcial provê mecanismos de proteção de direito autoral suficientes para diversas aplicações que requeiram distribuição em massa de conteúdo não confidencial e comercial.

Capítulo 5

MECANISMO PARA A DISTRIBUIÇÃO SEGURA DE VÍDEO MPEG

Este capítulo apresenta os requisitos e a especificação do mecanismo proposto nesta dissertação.

Nos capítulos anteriores foram descritos os aspectos de segurança envolvidos, o padrão de compressão MPEG e os principais esquemas de criptografia existentes. Após discutir estes aspectos é possível especificar um mecanismo para a distribuição segura de vídeo. Assim, este capítulo trata da especificação do Mecanismo para a Distribuição Segura de Vídeo MPEG proposto neste trabalho.

O presente capítulo está dividido em duas partes: os requisitos considerados para o mecanismo, e a especificação, de acordo com os requisitos apresentados.

A especificação deste mecanismo resulta no **S/Viewer**, *player* MPEG seguro, na definição do esquema de codificação / criptografia MPEG para o servidor de vídeo, o **S/Server**, e no protocolo de acesso ao vídeo.

5.1 REQUISITOS

Baseado nos estudos anteriores é possível levantar quais as características importantes para um mecanismo para distribuição segura de vídeo MPEG. Dentre estas características deve-se considerar quais os serviços de segurança que devem ser atendidos, quais os tipos de vídeos que devem ser executados, diferentes requisitos de segurança para os vídeos armazenados no servidor, como garantir a autenticidade e integridade do material a ser reproduzido no visualizador, como garantir a segurança do protocolo de comunicação.

Os serviços de segurança relevantes para este mecanismo são: autenticidade, integridade, controle de acesso e confidencialidade.

Garantindo o serviço de autenticidade, o cliente irá executar somente vídeos recebidos de servidores conhecidos e confiáveis. O usuário irá selecionar um servidor conhecido,

e o programa visualizador irá verificar a identidade do mesmo através de certificados digitais. Da mesma forma, o servidor só irá transmitir um vídeo para um usuário devidamente identificado e cadastrado.

A integridade dos vídeos deve ser sempre verificada, de modo a garantir que a sua reprodução não contenha erros de transmissão, ou tenha sido alterado indevidamente.

O controle de acesso é necessário quando se criam diferentes níveis de segurança para o conteúdo armazenado no servidor. Assim, um vídeo pode ser de segurança baixa, e pode ser de acesso livre, enquanto outro pode ser de acesso restrito devido à importância do seu conteúdo. Além disso, para controlar a distribuição do vídeo garantindo o direito autoral, é necessário manter um registro de quais foram as cópias distribuídas e para quem.

Dado a existência de diversos níveis de segurança para os vídeos armazenados, alguns destes podem requerer que seja garantido o sigilo do seu conteúdo. Assim, torna-se necessário utilizar-se de técnicas que garantam a confidencialidade.

Associando o controle de acesso aos diversos níveis de segurança existentes no servidor é possível garantir o direito autoral. Assim, caso se perceba uma distribuição indevida do material originalmente armazenado neste servidor, é possível identificar os prováveis culpados.

Outro aspecto que deve ser considerado é a confiabilidade do protocolo de comunicação entre o servidor e o *player*. Este não pode ser vulnerável a ataques conhecidos, possuindo a máquina de estados bem definida, sem falhas, tratando todos os possíveis erros.

Desta forma, pode-se organizar os requisitos do Mecanismo de Distribuição Segura de Vídeo MPEG em quatro aspectos principais:

- Controle de Acesso;
- Reprodução de Material Autenticado e Íntegro;
- Diversos Níveis de Segurança;
- Confiabilidade do Protocolo.

Estes aspectos principais são discutidos nos itens a seguir.

5.1.1 Controle de acesso

Os vídeos armazenados em um servidor não possuem um mesmo fim, ou seja, diferentes usuários em diferentes situações irão acessar este conteúdo. Por exemplo, no contexto de ensino on-line, um seminário pode ser assistido por qualquer usuário, enquanto um vídeo pertencente a um curso só deve ser visto por um aluno matriculado no mesmo. Outra situação possível é aquela em que somente um pequeno grupo de usuários possa reproduzir determinados vídeos, como vídeos de reuniões de um projeto. No caso de uma vídeo-locadora, as permissões de acesso devem ser definidas dinamicamente, pois o usuário só poderá assistir ao vídeo selecionado após efetuar o pagamento. Assim, os vídeos disponíveis no servidor possuem diferentes requisitos de segurança, já que estes podem ser vídeos de cursos on-line, seminários, vídeos desenvolvidos para projetos, registros de reuniões, conteúdo de uma vídeo-locadora.

Devido a estes diferentes requisitos de segurança é necessário implementar controle de acesso, onde o usuário é identificado e os seus privilégios são verificados. Os privilégios de um usuário estão relacionados aos vídeos que o usuário tem permissão para reproduzir.

Inicialmente podem-se dividir os vídeos armazenados no servidor em dois grupos: vídeos de acesso público e de acesso restrito.

Os vídeos de acesso público podem ser reproduzidos por qualquer usuário, fazendo parte dos privilégios de todos os usuários que acessem o servidor de vídeo. Apesar de poderem ser reproduzidos por qualquer cliente, os vídeos de acesso público só podem ser reproduzidos após a identificação completa do usuário.

Já os vídeos de acesso restrito, como é o caso daqueles pertencentes a cursos on-line ou a uma vídeo-locadora, possuem associados a eles uma lista com os usuários que tem a autorização para reproduzi-los.

Uma vez identificado o usuário através de seu certificado digital, os seus privilégios são determinados checando as listas de permissão dos vídeos. Esta busca resulta nos vídeos de acesso restrito que o usuário pode reproduzir e dos vídeos de acesso público disponíveis no servidor.

A lista de permissão de acesso de um vídeo é criada quando da inserção do mesmo no servidor de vídeo, podendo ser alterada posteriormente pelo responsável pelo vídeo. Os usuários são identificados na lista de permissão de vídeo pelo número de série do certificado e seu nome completo. A identificação dos privilégios é possível já que estes parâmetros são obtidos do próprio certificado digital do cliente.

O processo de cadastro de permissões no servidor de vídeo pode ser automatizado para facilitar o gerenciamento. Assim, quando um aluno matricula-se em um curso, o seu certificado digital é solicitado, e os dados necessários são adicionados aos privilégios dos vídeos pertencentes ao curso.

5.1.2 Reprodução de Material Autenticado e Íntegro

Os serviços de segurança de autenticidade e integridade são essenciais para o correto funcionamento do mecanismo proposto. Assim, o visualizador S/Viewer deve garantir estes serviços.

Para iniciar a reprodução de um vídeo MPEG seguro, o S/Viewer deverá solicitar o certificado digital do servidor de vídeo, garantindo desta forma a autenticidade do material a ser reproduzido. Ou seja, deve-se verificar se o nome do servidor contatado é o mesmo identificado no certificado digital. Da mesma forma, o servidor deve identificar corretamente cada um dos seus clientes, através de certificados digitais, garantindo a autenticidade destes.

Além disto, o *player* também deve verificar a integridade do vídeo a ser reproduzido, garantindo que somente material sem erros ou alterações seja reproduzido. A integridade do vídeo pode ser checada através da verificação do hash do mesmo, que está encriptado (assinatura digital). O uso de assinatura digital garante não só a integridade do vídeo, como também a sua autenticidade.

O método para verificação da integridade do vídeo irá diferir em função do modo como é transmitido. Quando se utiliza rede de alta velocidade, o arquivo de vídeo pode ser transmitido completo para posterior reprodução, e então o hash pode ser calculado para todo o arquivo. Caso utilize-se uma rede de velocidade mais baixa, como a Internet, é necessário utilizar técnica de *streaming* de vídeo, o que implica em verificar a integridade de cada seqüência de vídeo recebida. Assim, o método de verificação da integridade depende do modo de transmissão a ser utilizado.

5.1.3 Níveis de Segurança

Conforme discutido nos itens anteriores, os vídeos armazenados no servidor destinam-se a diferentes aplicações, e conseqüentemente possuem diferentes requisitos de segurança. Devido a esta característica, é necessário definir diversos níveis de segurança para os vídeos.

Estes diversos níveis de segurança podem ser obtidos com diferentes métodos de criptografia. Por exemplo, o VEA possui nível de segurança diferente da Criptografia Seletiva [Qiao 98]. Outra maneira de implementar este requisito é efetuar a criptografia no momento da requisição do vídeo, criando uma chave de criptografia do vídeo para cada usuário.

Resumindo, os níveis de criptografia podem ser obtidos através de dois modos:

- Utilizando diferentes esquemas de codificação / criptografia MPEG;
- Através do uso de diferentes chaves de criptografia do vídeo para diferentes clientes.

Conforme descrito no capítulo 4, os diversos esquemas de criptografia MPEG possuem diferentes níveis de segurança e diferentes velocidades de criptografia, conforme se observa na Tabela 4.8 [Qiao 98]. Assim, variando os algoritmos de criptografia é possível obter diferentes níveis de segurança para que haja um compromisso adequado com a velocidade de decodificação.

Para um dado esquema de criptografia MPEG escolhido, ainda é possível utilizar diferentes chaves de criptografia. Assim, cada cliente pode reproduzir o vídeo MPEG seguro solicitado utilizando uma chave diferente, ao invés de todos os clientes utilizarem a mesma chave para o mesmo vídeo. Este modo proporciona um bom nível de segurança.

5.1.4 Confiabilidade do protocolo

O protocolo de comunicação entre o cliente e o servidor deve ter sua máquina de estados bem definida, evitando inconsistências, estados sem saída, transições indesejadas. Também deve tratar todos os possíveis erros, de modo a não deixar vulnerabilidades no mecanismo proposto.

A troca de mensagens não pode estar sujeita a ataques conhecidos, como os *reply attacks* (ataques onde mensagens válidas são capturadas, e utilizadas posteriormente pelo oponente para obter dados do sistema). Por isso, mecanismos para impedir o uso posterior de mensagens válidas também devem ser implementados.

5.2 ARQUITETURA DO SISTEMA

A arquitetura do sistema de vídeo proposto é bastante simples, conforme observa-se na Figura 5.1.

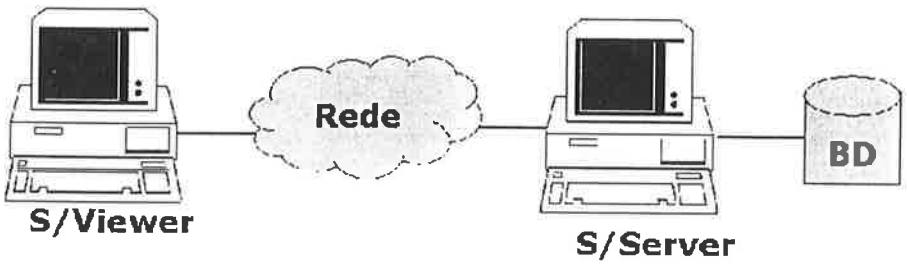

Figura 5.1: Arquitetura do Sistema

O servidor de vídeo, S/Server, possui uma base de dados associada, onde estão cadastrados os vídeos com as informações necessárias (nível de segurança e usuários que terão acesso aos vídeos), e os usuários com as suas permissões.

O S/Viewer permite ao usuário assistir vídeos MPEG padrão ou seguros (vms - vídeo MPEG seguro), além de poder escolher entre assistir um vídeo gravado localmente ou acessar um servidor de vídeo.

O cliente se comunica com o servidor através do protocolo TCP, utilizando uma porta entre as disponíveis. A implementação do servidor suporta múltiplas conexões, através da criação de novos processos.

5.3 Especificação

A especificação do mecanismo de distribuição segura de vídeo MPEG consta de três partes: do protocolo de acesso, do servidor de vídeo (**S/Server**) e do *player* (**S/Viewer**). Os próximos itens tratam desta especificação.

5.3.1 Protocolo de Acesso

O objetivo deste item é especificar o protocolo de acesso ao vídeo MPEG seguro, bem como todas as transações envolvidas, mensagens trocadas e seus formatos.

Para acessar um vídeo disponível no **S/Server** (servidor de vídeo), o **S/Viewer** (o *player* MPEG seguro proposto) irá estabelecer uma conexão TCP com o mesmo. Através desta conexão TCP, serão trocadas as mensagens do protocolo de comunicação entre o servidor e o cliente, e o arquivo de vídeo será transmitido.

A comunicação entre o cliente e o servidor deve seguir algumas etapas:

- Identificação;
- Requisição de lista de vídeos (opcional se o usuário já souber o vídeo);
- Requisição do vídeo.

A partir destas três etapas é possível definir quais as mensagens trocadas entre o cliente e o servidor. A Figura 5. 2 ilustra as mensagens trocadas durante o processo de acesso ao vídeo.

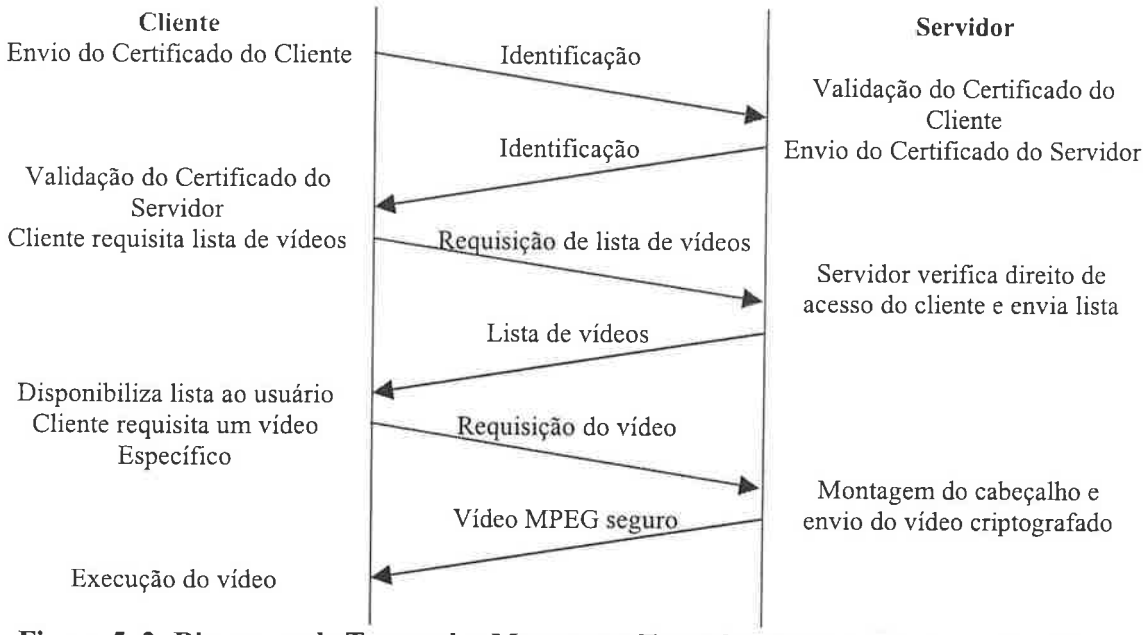

Figura 5. 2: Diagrama de Tempo das Mensagens Trocadas entre o Cliente e o Servidor para Acesso ao Vídeo MPEG Seguro

A mensagem de **Identificação do Cliente**, que é a primeira mensagem que o cliente envia, contém um campo de controle, o seu certificado digital (Figura 5. 4) e um número, que associa informação temporal e um outro número aleatório, encriptado com a sua chave privada ($E_{KRC}[n^o]$), conforme observa-se na Figura 5.3.

Controle	Certificado Digital	$E_{KR}[n^o]$

Figura 5.3: Formato da Mensagem de Identificação do Cliente

O campo de controle é utilizado para identificar o tipo da mensagem, neste caso uma mensagem de identificação.

This Certificate belongs to:	This Certificate was issued by:
Cintia Borges Margi	LARC - CA
cbmargi@larc.usp.br	Laboratorio de Arquitetura e Redes
Departamento de Projetos	de Computadores
LARC - PCS - EPUSP	LARC - PCS - EPUSP
Sao Paulo, Sao Paulo, BR	Sao Paulo, Sao Paulo, BR

Serial Number: B9:39:FC:00:00:00:04
This Certificate is valid from Wed Jan 27, 1999 to Thu Jan 27, 2000
Certificate Fingerprint: 60:15:9F:50:76:D7:36:A7:4C:69:F1:6F:86:EE:8D:DF

Figura 5. 4: Certificado Digital

O número que associa informação temporal e um outro número aleatório é usado para que o servidor possa rejeitar requisições antigas que se atrasaram no trajeto, ou então para evitar que um impostor copie a mensagem de identificação, e posteriormente faça-se passar por um cliente legítimo (ataques do tipo *replay*). Assim, esta informação temporal estabelece um tempo de vida para a mensagem de identificação. Na composição deste número utiliza-se também um número aleatório para dificultar a falsificação deste valor.

Ao receber a mensagem de **Identificação do Cliente**, o servidor irá verificar o certificado digital do cliente, e decriptar o número recebido com a chave pública do cliente. O número é decomposto, e a informação temporal é verificada. Se a mensagem ainda estiver dentro do seu tempo de vida, o servidor envia o seu certificado digital, o horário recebido do cliente encriptado com a chave pública do mesmo, e o número (que associa informação temporal e um outro número aleatório) encriptado com a sua chave privada em uma mensagem de **Identificação do Servidor** (Figura 5.5).

Controle	Certificado Digital	E_{KU}[nº recebido]	E_{KR}[nº]

Figura 5.5: Formato da Mensagem de Identificação do Servidor

Ao receber a mensagem de **Identificação do Servidor**, o cliente faz três verificações:

1- irá verificar se o certificado digital recebido corresponde ao servidor contatado;

2- se o número que recebeu encriptado com a sua chave pública é o mesmo que enviou;

3- se o número com informação temporal assinado pelo servidor indica que a mensagem ainda está dentro do seu tempo de vida.

Caso as três verificações sejam bem sucedidas, o cliente terá a garantia de estar acessando o servidor desejado, e assim poderá continuar efetuando a requisição dos vídeos disponíveis (Requisição de Lista de Vídeos), ou então solicitar um vídeo específico (Requisição de Vídeo). Observe que no caso de um curso a distância, o próprio curso indicará o nome do vídeo a ser reproduzido, fazendo assim uma requisição direta do vídeo.

No caso da **Requisição de Lista de Vídeos** disponíveis (Figura 5.6) identificada pelo campo de controle, o servidor irá verificar quais os privilégios do cliente e irá responder com a lista de vídeos. A mensagem de **Requisição de Lista de Vídeos** também possui a informação de tempo de vida da mensagem, indicada pelo número (que associa a informação temporal ao número aleatório) assinado pelo cliente.

Controle	E_{KR}[nº]

Figura 5.6: Formato da Mensagem de Requisição de Lista de Vídeos

A mensagem de **Lista de Vídeos**, com a qual o servidor responde a requisição do cliente, contém um campo de controle, a lista de vídeos e a informação temporal (Figura 5.7). A lista de vídeos contém os nomes dos vídeos que o cliente portador do certificado digital apresentado pode executar.

Controle	Vídeo 1 Vídeo 2 Vídeo 3 Vídeo 4 Vídeo 5 Vídeo 6	$E_{KR}[n^o]$

Figura 5.7: Formato da Mensagem com a Lista de Vídeos

Após receber a mensagem **Lista de Vídeos**, o S/Viewer mostra ao usuário a lista de vídeos disponíveis. O usuário seleciona o vídeo desejado, e então a **Requisição de Vídeo** é feita ao servidor. Esta mensagem é composta pelo campo de controle, pelo nome do vídeo e pela informação temporal (Figura 5. 8).

Controle	Nome do Vídeo	$E_{KR}[n^o]$

Figura 5. 8: Formato da Mensagem de Requisição de Vídeo

Quando o servidor recebe uma **Requisição Vídeo**, este verifica se o usuário possui permissão de acesso ao mesmo. Em caso afirmativo, inicia o processo de transmissão do vídeo MPEG seguro. Caso contrário, envia uma mensagem de erro informando que o acesso ao vídeo não foi permitido.

Para indicar ao cliente algum erro, o servidor faz uso da Mensagem de Erro (Figura 5. 9), que possui um campo destinado ao código do erro que ocorreu.

Controle	Código do Erro	$E_{KR}[n^o]$

Figura 5. 9: Formato da Mensagem de Erro

Tanto o servidor como o cliente, podem fazer uso da mensagem de *reset* (Figura 5.10), caso seja necessário terminar a conexão estabelecida.

Controle	$E_{KR}[n^o]$

Figura 5.10: Mensagem de RESET

Todas as mensagens trocadas entre o servidor e o cliente possuem um campo com a informação temporal (número composto pela informação temporal e um outro número aleatório), assinado digitalmente pelo remetente. Desta forma garante-se a autenticidade das mensagens, evita-se que mensagens atrasadas sejam respondidas e, também, evita-se ataques do tipo *reply*.

É importante registrar estas transações de controle de acesso e troca de certificados em um arquivo de log do servidor de vídeo. Com estas informações é possível identificar quais são os usuários do sistema e quem acessa os vídeos disponíveis, permitindo uma posterior auditoria, cobrança, ou até mesmo uma simples verificação dos vídeos mais acessados.

5.3.2 O servidor de Vídeo: S/Server

O servidor de vídeo é responsável pela criptografia e pela transmissão do vídeo MPEG. Assim, é necessário identificar as suas características e o seu modo de funcionamento, o que é descrito neste item.

Após o processo de identificação e seleção do vídeo, o servidor recebe a mensagem de **Requisição do Vídeo**. Para enviar o vídeo ao cliente, o servidor deve cumprir as seguintes etapas, conforme ilustrado na Figura 5.11:

- criptografar o vídeo MPEG, ou localizar um arquivo previamente encriptado;
- montar o cabeçalho MPEG seguro;
- montar o arquivo a ser transmitido;
- transmitir o arquivo.

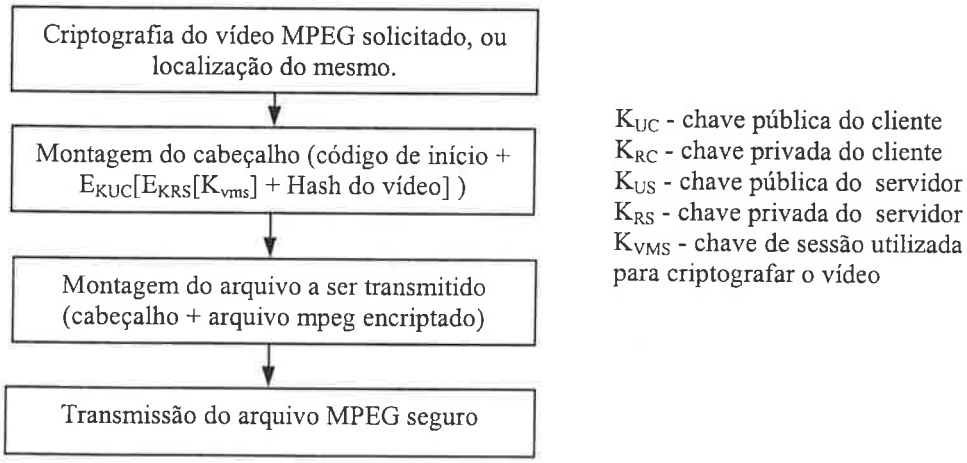

Figura 5.11: Fluxograma da Criptografia e Transmissão do Vídeo Seguro

A criptografia do vídeo MPEG é feita utilizando um dos esquemas apresentados anteriormente no Capítulo 4, conforme o nível de segurança cadastrado para o vídeo no servidor:

- escolhe-se o algoritmo correspondente ao nível de segurança cadastrado no servidor para o vídeo em questão.
- em geral, utiliza-se o mais eficiente, tomando como base a comparação mostrada no Capítulo 4, que implica na utilização do VEA [Qiao 98].

A criptografia do arquivo MPEG pode ser realizada antes da transmissão do mesmo, ou somente quando o usuário solicita o vídeo (criptografia *on the fly*). Caso a criptografia seja anterior à requisição, esta será atendida mais rapidamente, porém o nível de segurança é menor já que diversos usuários receberão vídeos MPEG seguros criptografados com a mesma chave. A criptografia no instante da requisição permite que diferentes usuários recebam arquivos MPEG seguros com diferentes chaves de criptografia.

O cabeçalho MPEG seguro é necessário para que o visualizador (**S/Viewer**) possa reproduzir o vídeo MPEG. Este cabeçalho é composto pelo código de início de seqüência de vídeo VMS (vídeo MPEG seguro), pela chave de criptografia do vídeo VMS (K_{VMS}) e pelo *hash* do vídeo VMS.

Para garantir a confidencialidade da chave de criptografia do vídeo (K_{VMS}) e do *hash* do vídeo VMS, estas informações são criptografadas com a chave pública do usuário (K_{UC}). Além disto, para garantir a autenticidade da chave de criptografia do vídeo (K_{VMS}), esta é criptografada com a chave privada do servidor (K_{RS}). Assim o cabeçalho fica conforme observa-se na Figura 5.12.

Código de início	$E_{KUC}[E_{KRS}[K_{vms}]$ + *Hash* do vídeo]

Figura 5.12: Cabeçalho do Arquivo MPEG seguro (VMS)

Uma vez montado o cabeçalho, acrescenta-se ao início do arquivo MPEG encriptado, e inicia-se a transmissão do vídeo. A transmissão do arquivo de vídeo pode ocorrer de duas formas: o arquivo completo ou por *streaming*.

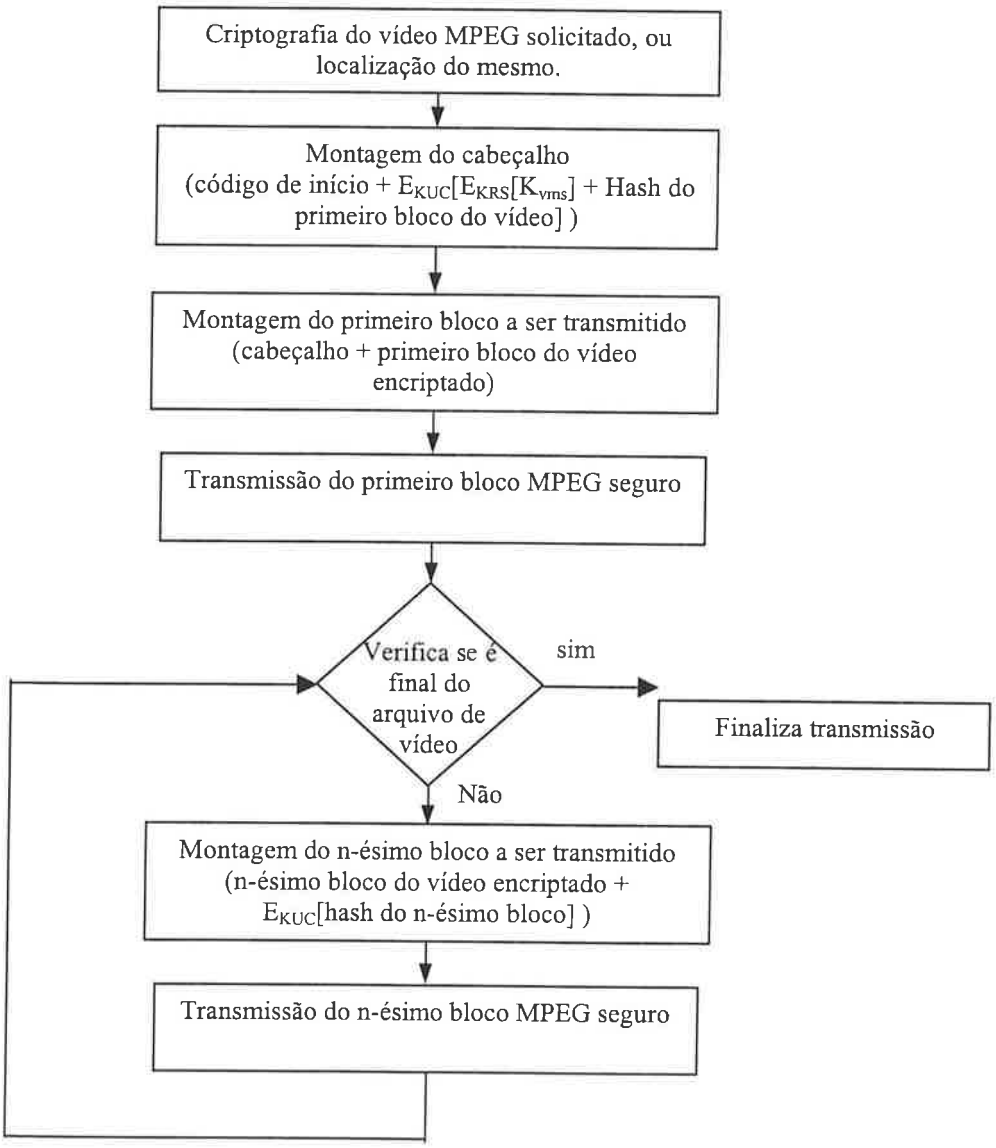

Figura 5.13: Fluxograma da Criptografia e Transmissão do Vídeo Seguro por *streaming*

Quando a infra-estrutura de rede utilizada é de alta velocidade, o arquivo de vídeo pode ser transmitido integralmente, para posterior reprodução. Assim, segue-se exatamente o fluxograma mostrado na Figura 5.11.

Se o vídeo estiver sendo transmitido na Internet, a transmissão deve ser feita utilizando técnicas de *streaming* de vídeo. O RTSP (*Real Time Streaming Protocol*) utilizado para transmissão de vídeo por *streaming* define como controlar a distribuição do vídeo, e não como este será transmitido [Schulzrinne 98]. Assim, o fluxograma do Mecanismo deve ser alterado, de modo que a verificação de integridade ocorra a cada trecho do vídeo transmitido. A Figura 5.13 ilustra a modificação necessária.

5.3.3 Reprodução do Vídeo MPEG: S/Viewer

Reproduzir um vídeo MPEG criptografado significa decriptografá-lo e decodificá-lo simultaneamente e em tempo real. Ou seja, o vídeo não fica disponível ao usuário decriptografado.

O *player* seguro deve ser capaz de reproduzir vídeos MPEG padrão e vídeos MPEG encriptados. Para identificar o tipo de vídeo a ser reproduzido é necessário determinar um código de início para o vídeo MPEG encriptado, uma vez que todo vídeo MPEG padrão é identificado pelo seu código de início de seqüência.

Assim, o primeiro passo para a reprodução de um vídeo é identificar o tipo de MPEG: padrão ou criptografado. Se for um arquivo MPEG padrão, a reprodução é executada normalmente. Caso seja um vídeo MPEG criptografado, são necessários os outros passos.

Se o vídeo a ser reproduzido for um MPEG criptografado, é necessário desmontar o cabeçalho e decriptografar as informações criptografadas utilizando para tanto a chave

privada do cliente ($D_{KRC}[E_{KRS}[K_{VMS}]$ + *Hash* do vídeo], resultando em $E_{KRS}[K_{VMS}]$ e no *Hash* do vídeo).

Em seguida, calcula-se o hash do arquivo MPEG criptografado e compara-se com o hash recebido. Caso ocorra algum erro, a reprodução do arquivo é terminada.

Esta descrição considera que o hash do vídeo é calculado para todo o arquivo MPEG, assim é necessário ter recebido o arquivo todo para iniciar a sua reprodução. Quando considera-se distribuição de vídeo por *streaming*, torna-se necessário alterar o método de cálculo do hash do arquivo MPEG todo para parte do arquivo MPEG.

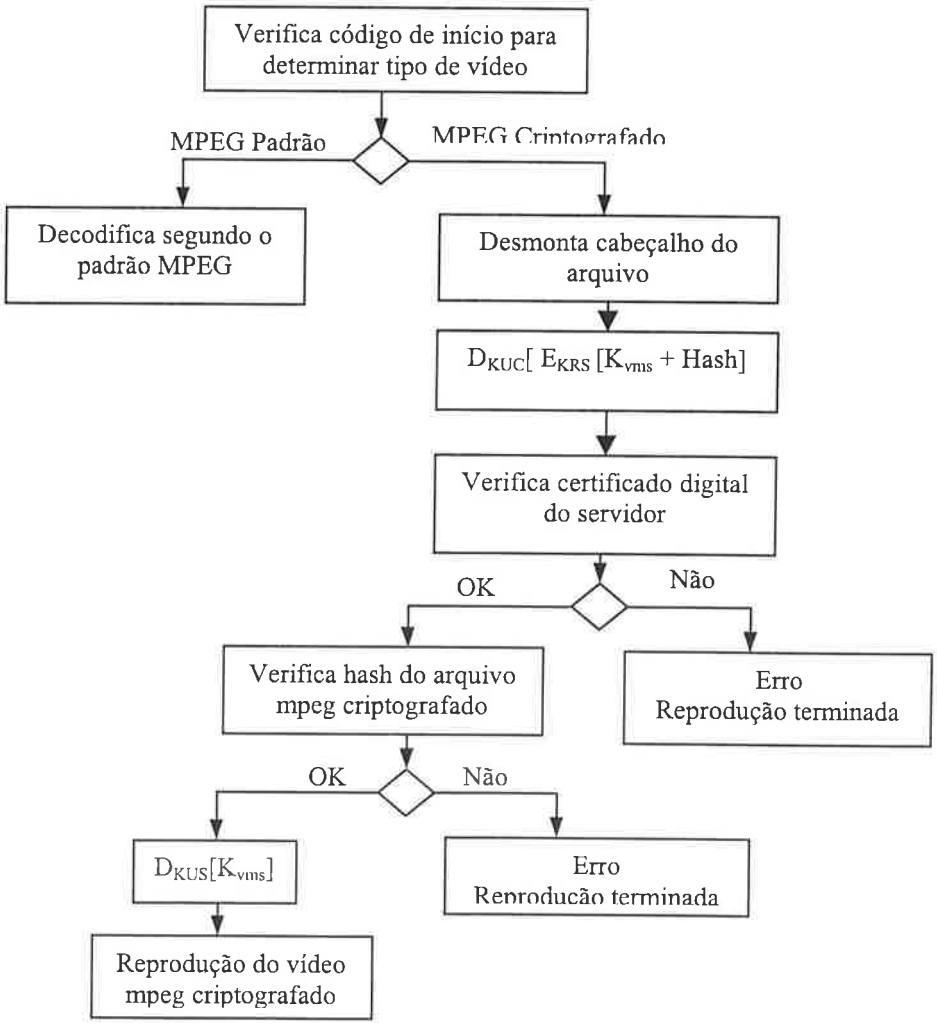

Figura 5.14: Fluxograma do Mecanismo de reprodução de vídeo MPEG criptografado

O próximo passo é obter a chave K_{VMS} utilizando a chave pública do servidor (através de $D_{US}[K_{VMS}]$) para, então, iniciar a reprodução do vídeo. A Figura 5.14 ilustra o processo de reprodução de um vídeo MPEG pelo *player* proposto.

A decriptografia e reprodução do vídeo variam conforme o tipo de criptografia escolhido. Assim, além de identificar se o vídeo a ser reproduzido é MPEG normal ou encriptado, é necessário identificar qual o tipo de criptografia empregado no código de início do vídeo.

Capítulo 6

PROJETO, IMPLEMENTAÇÃO E RESULTADOS OBTIDOS

Este capítulo apresenta os aspectos de projeto e implementação do mecanismo proposto, bem como as dificuldades encontradas e os resultados obtidos.

6.1 PROJETO

Neste item são discutidos diversos aspectos do projeto do S/Vierwer e do S/Server, como os requisitos para a interface, a máquina de estados, entre outros.

6.1.1 Interface do S/Viewer

A interface do S/Viewer deve ser uma interface gráfica e amigável. Através dela, o usuário deve ter acesso a todas as suas funções.

O S/Viewer permite ao usuário assistir vídeos MPEG padrão ou seguros (VMS). O *player* é capaz de diferenciar estes dois tipos de arquivos pelo código de início do mesmo durante a reprodução. Observe que o mecanismo de distribuição de vídeo pode entregar um arquivo de vídeo comum.

O usuário também pode escolher entre assistir um vídeo gravado localmente ou acessar um servidor de vídeo para selecionar o filme desejado. Se o usuário optar por acessar o servidor de vídeo seguro, o S/Server, deve selecionar o certificado digital a ser usado, e depois indicar o nome do servidor a ser conectado.

Uma vez escolhido o vídeo, o usuário deve utilizar a opção de "Play" para assisti-lo.

O S/Viewer também deve indicar ao usuário a necessidade de requisitar outro certificado digital quando a data de expiração estiver próxima. O acesso à chave privada do usuário, que deve ser armazenada encriptada, deve ser controlado por uma senha, de modo a impedir que um usuário possa se passar por outro.

Uma funcionalidade interessante para o S/Viewer é permitir ao usuário fazer uma lista dos vídeos mais interessantes e em que servidor está armazenado, ou seja, um caderno de endereços eletrônico.

Além de todas esses requisitos para a interface do visualizador, a implementação deve ser feita de maneira a impedir que um usuário mal intencionado possa fazer Engenharia Reversa.

6.1.2 Interface do S/Server

A interface do S/server deve permitir que as tarefas de manutenção e administração do sistema sejam executadas de maneira fácil e segura.

O servidor de vídeo seguro deve permitir ao administrador do mesmo:

- instalar um certificado digital para o servidor;
- remover um certificado digital para o servidor;
- cadastrar vídeos, determinando o seu nível de segurança (ou seja, escolher o tipo de criptografia empregada) e definindo os grupos de usuários que terão acesso aos vídeos;
- alterar o nível de segurança de vídeos já cadastrados;
- alterar os grupos de usuários que têm acesso a vídeos já cadastrados;
- remover vídeos do sistema;
- cadastrar usuários no sistema, definindo a quais;
- alterar informações de um usuário, e até grupos a que pertence;
- remover usuários do sistema.

Estas funções podem ser realizadas de duas maneiras: através de linha de comando, ou através da interface gráfica. No caso de linha de comando, os devidos parâmetros devem ser indicados. Caso utilize-se a interface gráfica, os parâmetros necessários serão solicitados para cada uma das funções a serem realizadas.

A interface do S/Server também deve permitir a visualização dos arquivos de registro

das transações executadas com o servidor.

O administrador do sistema deve ser notificado quando for necessário requisitar outro certificado digital, devido à próxima expiração daquele instalado.

6.1.3 Máquina de Estados do S/Server

A partir das mensagens trocadas no protocolo de acesso entre o S/Viewer e o S/Server, é possível desenhar a máquina de estados do servidor. Esta máquina de estados é simples, conforme observa-se na Figura 6.1.

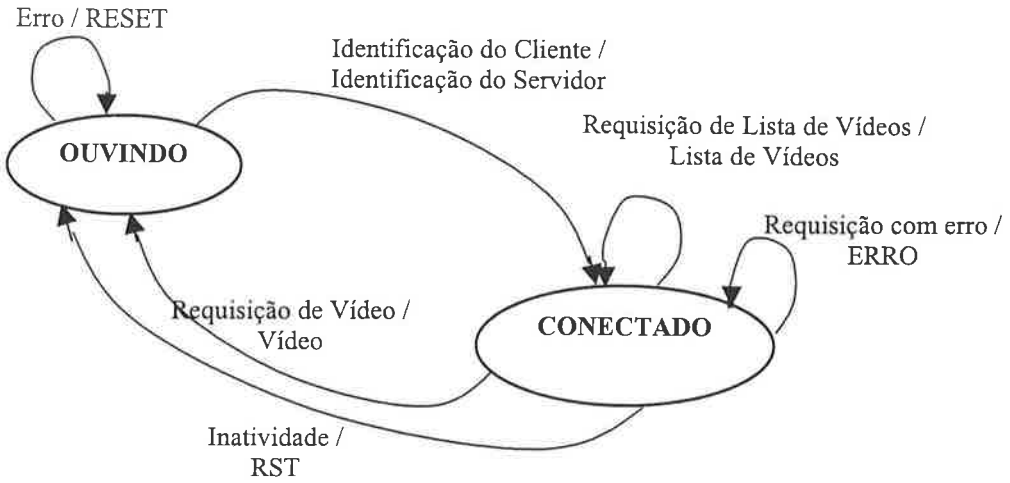

Figura 6.1: Máquina de Estados do S/Server

Se o servidor está no estado OUVINDO, a única maneira de mudar de estado é recebendo uma mensagem de identificação do cliente correta. Ou seja, a mensagem de identificação deve conter um certificado para o cliente e as informações temporais válidas, o que faz com que o servidor responda com a sua mensagem de identificação e mude para o estado CONECTADO. Em caso de erro no certificado do cliente ou no

tempo de vida da mensagem, o servidor envia uma mensagem do tipo *reset*, invalidando a tentativa de estabelecer uma conexão com o cliente.

Uma vez no estado CONECTADO, o servidor pode receber dois tipos de mensagens: de requisição de lista de vídeos e de requisição de vídeo.

Se o servidor recebe uma Requisição de Lista de Vídeos, ele irá verificar quais os privilégios do usuário, ou seja, quais vídeos pode assistir, e então enviar a Lista de Vídeos.

O outro tipo de mensagem que o servidor pode receber é uma Requisição de Vídeo. Quando recebe uma mensagem deste tipo, o servidor verifica se aquele usuário tem o direito de assistir o vídeo solicitado, e em caso afirmativo, envia o vídeo solicitado, mudando em seguida para o estado OUVINDO. Caso contrário, envia uma mensagem de tipo erro indicando que o usuário não pode assistir àquele vídeo, e continua no estado CONECTADO.

Caso o servidor perceba inatividade na comunicação estabelecida com o cliente, ele envia uma mensagem de RST e muda para o estado OUVINDO. Este conceito de inatividade deve considerar que o usuário pode estar selecionando o vídeo a ser assistido, tendo valores diferentes dos tempos de inatividade de outros protocolos.

6.2 Definições para a Implementação

Uma vez especificado o "Mecanismo de Distribuição Segura de Vídeo MPEG" e identificados os requisitos da implementação, é possível implementar o protótipo. Assim, os próximos itens tratam dos aspectos de implementação do S/Viewer e do S/Server, discutindo os problemas enfrentados e mostrando as interfaces resultantes.

6.2.1 Ambiente e Linguagem de Programação

O servidor e o *player* MPEG seguro propostos foram desenvolvidos a partir do software implementado na parte 5 do padrão ISO/IEC 13818-5 e 11172-5 (codificador e decodificador MPEG 1 e 2). Como estes programas foram desenvolvidos em linguagem C, esta é a linguagem adotada para o desenvolvimento do *player* e do servidor seguro.

O decodificador MPEG utilizado gera diversos formatos de saídas (*display* gráfico X11, YUV, SIF, TGA, PPM), sendo que a execução do vídeo em um display gráfico só é possível no ambiente Unix [MPEG 94]. Assim, este Mecanismo de Distribuição Segura de Vídeo será implementado no ambiente Unix (Linux Red Hat 6.1 com Kernel 2.20-12).

6.2.2 Aplicativos e Bibliotecas

Para facilitar o desenvolvimento da Interface Gráfica do S/Viewer, utilizou-se o aplicativo Glade [Glade]. Este aplicativo permite que sejam montadas janelas com botões e menus de maneira amigável. É possível utilizar a interface padrão GTk, ou a Gnome para gerar as janelas.

Uma vez montadas todas as janelas necessárias à interface, o aplicativo Glade gera o código correspondente às janelas em uma linguagem de programação escolhida. Esta linguagem pode ser C, C++ ou Perl. Para o S/Viewer utilizou-se a linguagem C.

Tanto o S/Viewer como o S/Server manipulam certificados digitais, utilizam funções de Hash e diversos algoritmos de criptografia padronizados. A biblioteca OpenSSL [OpenSSL] implementa os padrões de diversos algoritmos e certificados digitais, possui código aberto e documentação disponível. Assim, utilizá-la no desenvolvimento do S/Server e do S/Viewer tornou-se uma boa opção.

6.2.3 Múltiplas Conexões

O servidor de vídeo S/Server deve suportar múltiplas conexões simultâneas com usuários. Isto foi implementado criando um servidor concorrente com diversos processos, ou seja, para cada nova conexão é criado um novo processo (Figura 6.2) [Comer 93].

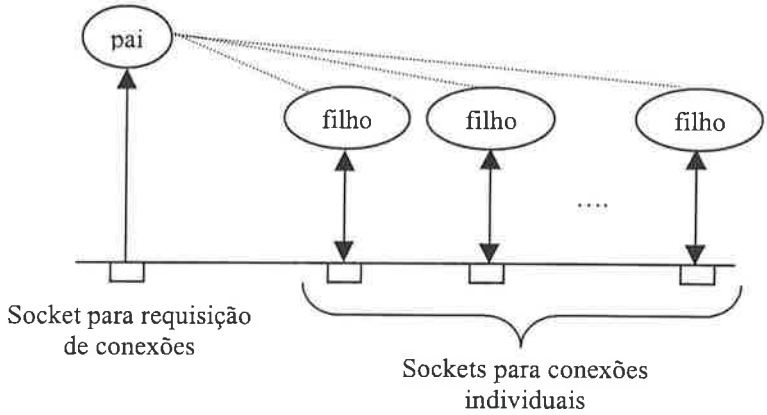

Figura 6.2: Estrutura de Processos Concorrentes do servidor

O processo principal (pai) fica aguardando novas conexões. Quando uma conexão é solicitada, um novo socket TCP e um processo filho são criados para atender a esta requisição. Desta forma, o processo pai continua recebendo pedidos de conexão.

6.2.4 Campo de Controle das Mensagens e Códigos de Erro

Os campos de controle para as mensagens do protocolo entre o S/Server e o S/Viewer são apresentados na Tabela 6.1.

Tabela 6.1: Campo de Controle das Mensagens

Mensagem	Identificação	Controle
Identificação do Cliente e do Servidor	IDENT	13
Requisição de Lista de Vídeos	REQ_LISTA	14
Lista de Vídeos Disponíveis	LISTA_VIDEO	15
Requisição de Vídeo	REQ_VIDEO	16
Vídeo	VIDEO	17
Reset	RST	99
Erro	ERRO	88

Quando ocorre um erro, envia-se uma mensagem de erro indicando qual o problema identificado. Os códigos de erro utilizados são apresentados na Tabela 6.2.

Tabela 6.2: Códigos de Erro

Tipo de Erro	Código de Erro
Certificado do Cliente Inválido	11
Certificado do Servidor Inválido	12
Vídeo Requisitado sem Permissão	21

6.3 Detalhes da Implementação do Protótipo

A implementação do protótipo foi mais complicada do que o esperado, devido a alguns problemas encontrados com a utilização de bibliotecas. A seguir, estão descritos os passos da implementação do protótipo.

6.3.1 Compilação do Código MPEG

O primeiro passo na implementação do protótipo foi compilar o código do codificador e do decodificador MPEG utilizados. Inicialmente, o código foi compilado em ambiente Windows NT, porém não era possível visualizar a imagem do vídeo, pois a implementação disponível deste formato existe somente para ambiente Unix. Para compilar o código fonte em ambiente Linux, foi necessário acrescentar algumas bibliotecas de desenvolvimento e gráficas ao ambiente Linux previamente instalado (configuração mínima de workstation).

6.3.2 Criação da interface do Cliente

O próximo passo foi a criação da interface gráfica para o cliente, S/Viewer. Para esta implementação foi utilizado o aplicativo Glade [Glade].

O protótipo do visualizador de vídeo MPEG, S/Viewer, possui uma tela principal (Figura 6.3) de onde é possível realizar as seguintes tarefas:

- abrir um vídeo local;
- acessar um servidor de vídeo e selecionar um arquivo;
- reproduzir um vídeo;
- selecionar um certificado digital.

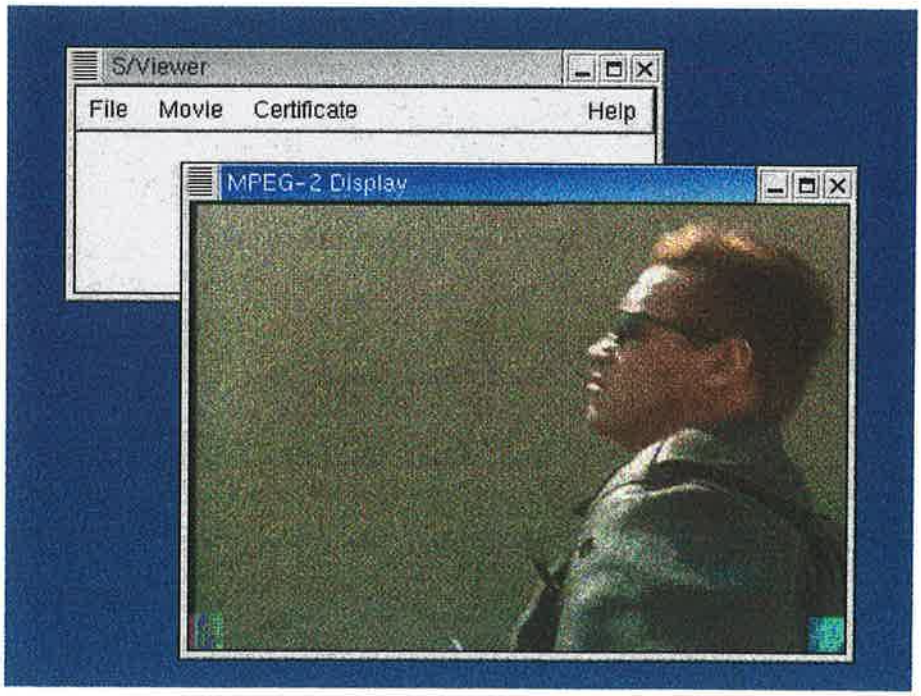

Figura 6.3: Protótipo do S/Viewer

No menu **File**, escolhe-se entre três opções:

- **Open**: que carrega um vídeo armazenado localmente;
- **Open on Server**: que carrega um vídeo a partir de um servidor de vídeo, utilizando o mecanismo implementado;
- **Quit**: que sai do programa.

No menu **Movie**, apenas a opção Play está implementada. Esta opção executa o arquivo de vídeo carregado pela opção **Open (on server)**.

No menu **Certificate** é possível selecionar o certificado digital do usuário.

6.3.3 Infra-estrutura de Rede

Após implementar a interface do cliente, a etapa seguinte foi implementar a infra-estrutura de comunicação entre o cliente e o servidor. A implementação utiliza-se de sockets TCP para a comunicação entre o servidor e o cliente.

O servidor espera conexões na porta 5193, quando recebe uma solicitação cria um processo filho que irá estabelecer a conexão com o cliente (vide item 2.3 deste capítulo).

A implementação inicial da infra-estrutura de comunicação para o cliente foi feita independentemente da interface anteriormente desenvolvida. Assim, a etapa seguinte foi unir estes dois programas.

6.3.4 Protocolo de Acesso

Com a estrutura de rede do cliente e do servidor funcionando, além da interface gráfica do primeiro, o próximo passo é implementar o protocolo de acesso entre o cliente e o servidor de acordo com o "Mecanismo de Distribuição Segura de Vídeo MPEG".

Esta etapa foi a mais complicada, onde foram encontrados os maiores problemas de desenvolvimento, e onde se gastou mais tempo.

Dentro do protocolo de acesso, é necessário fazer uso da assinatura digital do remetente, ou seja, encriptam-se dados com a chave privada. A chave privada deve estar armazenada encriptada com uma chave gerada a partir de uma senha fornecida pelo usuário. Apesar de existirem funções para realizar esta operação na biblioteca

OpenSSL, não foi possível implementá-la, devido à documentação deficiente e a falta de exemplos do uso da função. Antes de optar por não implementar esta funcionalidade, gastou-se um bom tempo tentando realizá-la. Desta forma, na implementação do protótipo, a chave privada é armazenada como texto limpo.

O passo seguinte foi efetuar a montagem da mensagem. Na especificação não se considera um campo para o tamanho da mensagem, porém a ausência deste torna inviável a implementação.

As primeiras mensagens transmitidas com dados assinados digitalmente não eram aceitas pelo servidor, devido à impossibilidade de decriptografá-las com a chave pública do cliente. Isto ocorreu por diversas razões, dentre as quais: perda de dados quando da recepção no socket TCP; envio do tamanho errado da mensagem. Uma vez alterados o esquema de manipulação de mensagens e da montagem do campo de tamanho, as mensagens foram aceitas e o protocolo pôde ser implementado.

6.3.5 Identificação do Tipo de Vídeo

Uma vez implementado o protocolo de acesso entre o cliente e o servidor, o próximo passo da implementação é a identificação do tipo de vídeo a ser reproduzido pelo S/Viewer.

A reprodução de um vídeo MPEG inicia-se identificando o código de início de seqüência de vídeo (0x000001B3), por isso é necessário que o S/Viewer saiba diferenciar esta seqüência do código de início de um vídeo MPEG seguro (VMS), que foi definido como sendo 0x00000200.

Quando um usuário escolhe a opção **Play** na interface gráfica do S/Viewer, a primeira tarefa a ser executada é a identificação do código de início do vídeo. Caso seja

identificado que o vídeo a ser executado é padrão, a reprodução ocorre normalmente. Se for identificado que o vídeo é seguro, o usuário é notificado e então inicia-se o procedimento de reprodução de vídeo MPEG seguro, conforme o fluxograma apresentado no capítulo 5.

6.3.6 Criptografia de Vídeo

Diversas são as possibilidades de implementação de criptografia de vídeo, conforme discutido no capítulo 4, como criptografia pura, seletiva, ou VEA.

Para o protótipo, não serão implementadas todas as opções de criptografia discutidas anteriormente. Apenas um tipo de criptografia foi implementado de modo a demonstrar os efeitos do uso de criptografia na transmissão / reprodução de arquivos de vídeo.

6.3.6.1 Criptografia Pura

A criptografia pura é a mais simples de ser implementada, e por isso foi escolhida para ser implementada no protótipo. Como algoritmo de criptografia simétrica foi escolhido o 3DES no modo CBC (Cipher Block Chain) [Stallings 98].

A implementação inicia-se com a criptografia somente durante a transmissão do vídeo. Desta forma, o arquivo de vídeo é encriptado quando solicitado ao S/Server, antes da transmissão do mesmo, e decriptado antes da execução pelo S/Viewer, já estando armazenado no cliente. Esta implementação já permite que sejam verificados os atrasos introduzidos pela criptografia antes da transmissão e da reprodução do vídeo.

6.4 Resultados Obtidos

Com a implementação do protótipo do MPEG *player* e do servidor de vídeo seguro, foi possível verificar a viabilidade da implementação do "Mecanismo de Distribuição Segura de Vídeo MPEG".

Porém, uma vez especificado e implementado o mecanismo, como assegurar que a sua utilização soluciona problemas de outros protocolos, ou apresenta uma melhor solução?

Um outro protocolo que poderia ser utilizado para distribuição de vídeo, bem como de outros materiais multimídia (imagens, texto), é o SSL [Shostack 95], ou TLS [Dierks 99]. O item 6.4.1 faz uma comparação do SSL com o mecanismo proposto neste trabalho.

Outro aspecto a ser discutido após a implementação é sobre o atraso inserido pelo uso de criptografia quando da transmissão e reprodução de um arquivo de vídeo. O item 6.4.2 trata deste tópico.

6.4.1 Comparação com SSL

O SSL provê autenticação do cliente e do servidor através de certificados digitais, assim como o mecanismo proposto neste trabalho. Ambos calculam o *hash* da mensagem transmitida permitindo a verificação da integridade. Porém, o "Mecanismo de Distribuição Segura de Vídeo MPEG" (MDSVM) provê alguns serviços, que geram diferenças significativas entre ambos:

- a criptografia adequada para o vídeo;
- a confidencialidade garantida mesmo quando o arquivo de vídeo está armazenado no cliente;

- possibilidade de identificar distribuições indevidas, que violem direitos autorais.

O SSL provê apenas a criptografia convencional para dados (DES, IDEA, RC4, entre outros), enquanto o MDSVM possui a capacidade de utilizar diversas técnicas de criptografia específicas para vídeo MPEG. Assim, o sigilo do vídeo é obtido mais eficientemente no MDSVM.

O MDSVM garante a confidencialidade do arquivo de vídeo, mesmo quando está armazenado na máquina do usuário, pois a decriptografia só pode ser realizada no S/Viewer. No caso do SSL, o sigilo é garantido apenas durante a transmissão e, uma vez na máquina do usuário, o arquivo não está encriptado. Estas diferenças ocorrem porque o MDSVM atua na camada de aplicação, e o SSL abaixo dela.

Como para obter a chave para abertura do vídeo encriptado é necessária a chave privada do cliente, é possível identificar para qual usuário válido o arquivo de vídeo foi entregue. E se alguma cópia ilegal for encontrada, é possível identificar quem a distribuiu, através da chave pública e dos arquivos de registro do S/Server.

Por estas razões, conclui-se que o "Mecanismo para Distribuição Segura de Vídeo MPEG", além de ter a implementação viável, apresenta uma boa solução para os problemas encontrados na distribuição de vídeo.

6.4.2 Efeitos do Atraso

Os arquivos de vídeo seguros encriptados *on the fly* (quando são requisitados) introduzem um atraso na transmissão de vídeo. Da mesma forma, a decriptografia do vídeo para a sua execução também gera atraso.

Os arquivos de vídeo utilizados para teste do protótipo do "Mecanismo de Distribuição Segura de Vídeo MPEG" são aqueles recomendados como referência pelo comitê MPEG. O nome dos vídeos, sua duração e tamanho são observados na Tabela 6.3.

Tabela 6.3: Vídeos de Teste

Vídeo	Tamanho (bytes)	Duração (seg)
bike.mpg	642.590	3,801
flowers.mpg	2.819.836	4,389
siegel.mpg	2.078.802	3,053
tennis.mpg	1.246.001	3,429

Com a implementação da criptografia de vídeo pura (descrita no item 6.3.6.1), foi possível quantificar o atraso introduzido. A Tabela 6.4 mostra os atrasos obtidos com o uso da criptografia simples, algoritmo 3DES-CBC.

O computador utilizado para efetuar os testes é um Intel Pentium III 650Mhz, com 128 MB de memória RAM, e Linux 7.0 instalado. O servidor e o cliente foram executados simultaneamente no computador referido.

Tabela 6.4: Atraso Introduzido pela Criptografia do Vídeo

Vídeo	Atraso na Transmissão (ms)	Atraso na Reprodução (ms)
bike.mpg	327	136
flowers.mpg	1.647	1.088
siegel.mpg	1.248	44
tennis.mpg	725	263

Conhecendo a duração dos arquivos de vídeo usados para teste do Mecanismo de Distribuição Segura de Vídeo MPEG, é possível obter qual a porcentagem do atraso na transmissão devido à montagem e criptografia do arquivo de vídeo seguro, e também na reprodução do vídeo devido à desmontagem e decriptografia do vídeo. A Tabela 6.5 indica estes resultados, bem como a porcentagem do atraso global inserido pelo uso de criptografia *on the fly* no servidor e no cliente.

Tabela 6.5: Atraso Percentual

Vídeo	Atraso na Transmissão	Atraso na Reprodução	Atraso geral
bike.mpg	8,60%	3,58%	12,18%
flowers.mpg	37,53%	24,79%	62,31%
siegel.mpg	40,88%	14,41%	55,29%
tennis.mpg	21,14%	7,67%	28,81%

A criptografia do vídeo introduziu um atraso maior do que a decriptografia, devido às operações que o servidor executa no processo. É possível que a própria implementação do algoritmo de criptografia e decriptografia utilizado possua diferentes tempos de processamento.

O atraso inserido mostrou-se alto, porém conforme descrito por [Qiao 98] e apresentado no capítulo 4, a criptografia pura é o processo mais lento.

É importante identificar o tempo que o usuário tolera esperar pelo arquivo de vídeo, já que com a criptografia pura a espera pode representar até 60% da duração do vídeo. Uma vez conhecida a tolerância do usuário, é necessário definir qual o melhor mecanismo de criptografia de vídeo para cada aplicação, minimizando o atraso no início da reprodução do vídeo.

Capítulo 7

Considerações Finais e Trabalhos Futuros

Este capítulo apresenta as considerações sobre o trabalho desenvolvido, bem como indicações de outros trabalhos relacionados que podem ser desenvolvidos.

7.1 Motivação

A distribuição de material multimídia pela Internet, principalmente de vídeo, tem aumentado consideravelmente. Na maior parte das vezes, este conteúdo não possui mecanismo algum de proteção de direitos autorais ou sigilo. Diversos trabalhos estão sendo desenvolvidos para solucionar a questão da proteção de direitos autorais e de sigilo. Técnicas de marcas d'água possibilitam a garantia dos direitos autorais [Bush 99], sendo que várias pesquisas estão sendo realizadas. O problema do sigilo é resolvido com técnicas de criptografia de vídeo, como descrito no Capítulo 4. Considerando estes dois aspectos, este trabalho tem como objetivo propor uma solução para o problema do sigilo, através do uso de várias técnicas de criptografia de vídeo, e garantir a autenticidade das partes envolvidas no processo de acesso ao vídeo, de modo que a distribuição do vídeo possa ser controlada, minimizando problemas com o direito de autoria.

7.2 Objetivos

Os objetivos apresentados para este trabalho são:

- analisar os aspectos de segurança envolvidos na distribuição de vídeo;
- levantar os requisitos de um mecanismo para distribuição segura de vídeo MPEG;
- especificar um mecanismo para distribuição segura de vídeo MPEG;
- implementar um protótipo deste mecanismo;
- verificar a viabilidade do mecanismo proposto.

Conforme é possível observar nos capítulos anteriores, os objetivos propostos foram cumpridos, e o mecanismo proposto pode ser utilizado para minimizar ou eliminar os problemas citados na distribuição de vídeo.

Assim, especificar e implementar um protótipo de "Um Mecanismo para Distribuição Segura de Vídeo MPEG" permitiu verificar a sua viabilidade, e como os problemas encontrados na distribuição de vídeo podem ser minimizados, ou até eliminados.

7.3 Resultados Obtidos

Uma questão que pode ser formulada é por que não utilizar um protocolo já padronizado, como o SSL, para a distribuição de vídeo. Tanto o SSL como o mecanismo proposto provêm a autenticação do servidor e do cliente (opcional no SSL) através de certificados digitais, e calculam o *hash* da mensagem transmitida permitindo a verificação da integridade. Porém, o "Mecanismo de Distribuição Segura de Vídeo MPEG" provê alguns serviços, que geram diferenças significativas entre ambos:

- a criptografia adequada para o vídeo;
- a confidencialidade garantida mesmo quando o arquivo de vídeo está armazenado no cliente;
- possibilidade de identificar distribuições indevidas, que violem direitos autorais.

Estes fatos demonstram a validade da proposta do trabalho, dado que os protocolos existentes ainda não contemplam todos os aspectos de segurança necessários.

Outro resultado obtido no trabalho mostra como é importante definir qual o custo aceito pelo usuário do sistema de distribuição segura de vídeo, ou seja, quanto tempo o usuário tolera esperar pelo arquivo de vídeo, já que com a criptografia pura a espera pode

representar até 60% da duração do vídeo. Além de definir a tolerância do usuário, é necessário definir qual o melhor mecanismo de criptografia de vídeo para cada aplicação, minimizando o atraso no início na reprodução do vídeo.

7.4 TRABALHOS COMPLEMENTARES

Apesar de ter cumprido os objetivos propostos, este trabalho não esgota todas as possibilidades de pesquisa neste campo, ainda sendo necessários desenvolver trabalhos de pesquisa em:

- Comparação quantitativa dos algoritmos de criptografia de vídeo;
- Verificação da viabilidade do uso destes algoritmos de criptografia de vídeo para *streaming*;
- Um mecanismo para distribuição segura de áudio;
- Marca d'água para áudio;
- Marca d'água para vídeo.

Os tópicos apresentados indicam alguns trabalhos relacionados, que podem ser desenvolvidos de maneira complementar ao Mecanismo de Distribuição Segura de Vídeo MPEG.

REFERÊNCIAS BIBLIOGRÁFICAS

Aqui estão indicadas as referências bibliográficas utilizadas neste trabalho.

[Agi 96] Iskender Agi and Li Gong; "An Empirical Study of Secure MPEG Video Transmission". In: **Proceedings of the Internet Society Symposium on Network and Distributed System Security**, p. 137-144, San Diego, CA, Fev 1996.

[Bush 99] Christoph Bush, Wolfgang Funk, Stephen Wolthusen; "Digital Watermarking: From Concepts to Real-Time Video Applications". In: **IEEE Computer Graphics and Applications**, v. 19, n. 1, p. 25-35, Jan/Fev 1999.

[Comer 93] D. E. Comer, D. L. Stevens; "**Internetworking with TCP/IP, Client-Sever Programming and Applications BSD Socket Version – Vol. III**". Prentice-Hall International Editions, 1993.

[Dierks 99] T. Dierks, C. Allen; "The TLS Protocol, Version 1.0", **Request for Comments: 2246**, Category: Standards Track, Jan 1999.

[Diffie 76] W. Diffie and M. E. Hellman; "New Directions in Criptography". In: **IEEE Transactions on Information Theory**, v. 11-22, n. 6, Nov 1976.

[Feghhi 99] Jalal Feghhi, Jalil Feghhi, Peter Wlliams; "**Digital Certificates – Applied Internet Security**". Addison Wesley, 1999.

[Glade] Glade; "**Glade: GTK+ User Interface Builder**". Disponível em: http://glade.pn.org. Acessado em 26/11/2000.

[Griwodz 98] Carsten Griwodz, Oliver Merkel, Jana Dittmann and Ralf Steinmetz; "Protecting VoD the Easiear Way". In: **Proceedings of the 6th ACM International Conference on Multimedia**, Bristol, UK, Nov 1998, p. 21-28.

[Hardjono 00] Thomas Hardjono, Gene Tsudik; "IP Multicast Security: Issues and Directions". In: **Annales de Telecom**, v. 55, p. 324-340, Jul/Ago 2000.

[Jsavard] John Savard, "**The Advanced Encryption Standard (Rijndael)**". Disponível em: http://home.ecn.ab.ca/~jsavard/crypto/co040801.htm Acessado em: 26/11/2000.

[Kunkelmann 97] Kunkelmann, R. Reinema, R. Steinmetz, T. Blecher; "Evaluation of Different Video Encryption Methods for a Secure Multimedia Conferencing Gateway". In: **4th COST 237 Workshop**, Lisboa, Portugal, December 1997.

[Kunkelmann 98] T. Kunkelmann, U. Horn; "Partial Video Encryption Based on Scalable Coding". In: **5th International Workshop on Systems, Signals and Image Processing (IWSSIP'98)**, Zagreb, Croatia, Jun 1998.

[LeGall 91] D.J. LeGall; "MPEG: A Video Compression Standard for Multimedia Applications". In: **Communications of the ACM**, v. 34, n. 4, Abr 1991.

[Li 96] Yocheng Li, Zhigang Chen, See-Mon Tan and Roy H. Campbell; "Security Enhanced MPEG Player". In: **Proceedings of the First International Workshop on Multimedia Software Development (MMSD '96)**, Berlin, Germany, March 1996.

[Meyer 95] Jürgen Meyer and Frank Gadegast; **"Sicherheitsmechanismen für Multimedia-Daten am Beispiel MPEG-I Video"**. Projektbericht, TU Berlim, 1995. Disponível no site http://www.mpeg1.de

[Mitchel 96] JoanL. Mitchel, WilliamB. Pennebaker, Chad E. Fogg and Didier J. LeGall; **"MPEG Video Compression Standard"**. Chapman and Hall, 1996.

[MPEG 94] MPEG Software Simulation Group; "**MPEG-2 Video Encoder / Decoder, Version 1.1**", Jun 1994. MPEG-L@netcom.com

[OpenSSL] OpenSSL.org; "**OpenSSL: The Open Source Toolkit for SSL/TLS**". Disponível em: http://www.openssl.org. Acessado em: 26/11/2000.

[Orzessek 98] Michael Orzessek, Peter Sommer; "**ATM & MPEG-2, Integrating Digital Video into Broadband Networks**". Prentice Hall, 1998.

[Qiao 97] Lintian Qiao and Klara Nahrstedt; "A New Algorithm for MPEG Video Encryption". In: **Proceedings of the First International Conference on Imaging, Science, Systems and Technology (CISST '97)**, Las Vegas, Nevada, Jul 1997.

[Qiao 97b] Lintian Qiao, Klara. Nahrstedt and I. Tam; "Is MPEG encryption using Random Lists instead of Zig Zag Order Secure?". In: **IEEE International Symposium on Consumer Electronics**, Singapore, Dez 1997.

[Qiao 98] Lintian Qiao and Klara Nahrstedt; "Comparison of MPEG Encryption Algorithms". In: **Computer & Graphics**, v. 22, n. 4, 1998.

[Rijmen] Vincent Rijmen, Joan Daemen; "**The Block Cipher Rijndael**" Disponível em: http://www.esat.kuleuven.ac.be/~rijmen/rijndael/. Acessado em: 26/11/2000.

[Rivest 78] R. L. Rivest, A. Shamir and L. Adleman; "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: **Communication of the ACM**, 1978.

[Schultzrinne 98] H. Schultzrinne, A. Rao, R. Lanphier; "**Real Time Streaming Protocol – Internet Draft**". IETF, 1998.

[Shostack 95] Adam Shostack, "**An Overview of SSL(version 2)**". Disponível em: http://www.homeport.org/~adam/ssl.html, Maio 1995. Acessado em: 26/11/2000.

[Silveira 98] R. M. Silveira; "Padrão de Compressão de Vídeo: MPEG". **Relatório Técnico do LARC - EPUSP**, 1998.

[Spanos 95] George A. Spanos and Tracy B. Maples; "Performance Study of a Selective Encryption Scheme for the Security Networked, Real-Time Video". In:

Proceedings of Fourth International Conference on Computer Communications and Networks, Las Vegas, Nevada, Set 1995.

[Spanos 96] George A. Spanos and Tracy B. Maples; "Security for Real-Time MPEG Compressed Video in Distributed Multimedia Applications". In: **Fifteenth IEEE International Phoenix Conference on Computers and Communications**, Scottsdale, AZ, Mar 1996.

[Stallings 98] William Stallings, "**Criptography and Network Security, Principles and Practice**", Prentice Hall, 2nd Edition, 1998.

[Tang 96] L. Tang; "Methods for Encrypting and Decrypting MPEG Video Data Efficiently". In: **Proceedings of The Fourth ACM International Multimedia Conference (ACM Multimedia '96)**, Boston, MA, Nov 1996.

[Tremblett 99] Paul Tremblett; "X.509 Certificates". In: **Dr. Dobbs Journal**, n. 301, Jul 1999.

[Voyatzis 99] George Voyatzis, Ioannis Pitas; "Protecting Digital-Image Copyrights: A Framework". In: **IEEE Computer Graphics and Applications**, v. 19, n. 1, p. 18-24, Jan/Fev 1999.