Aspectos de Desempenho do SICO – Sistema de Comunicação de Dados com Suporte Dinâmico a Segurança

Rodolfo B. Chiaramonte; Edward D. M. Ordonez; Fábio D. Pereira; Kalinka R.L.J C. Branco
Laboratório de Arquitetura de Sistemas Computacionais – Centro Universitário Eurípides de Marília (UNIVEM)
{chiaramonte, edmoreno, fabiopereira, kalinka}@fundanet.br

Resumo

Atualmente existe uma grande quantidade de algoritmos criptográficos, o que dificulta muito definir qual será o melhor algoritmo a ser utilizado em uma determinada conexão. Com isto, o objetivo do projeto foi a implementação um sistema capaz de definir através de alguns parâmetros coletados do ambiente em que estiver sendo executado qual é o algoritmo ou conjunto de algoritmos mais indicado para uma determinada conexão segura. Inicialmente foram implementados diversos algoritmos de criptografia, e posteriormente, o sistema para o controle da comunicação bem como um protocolo que permita alterar dinamicamente as chaves e os algoritmos. A implementação do sistema, chamado de SICO - Sistema Inteligente de Comunicação - foi desenvolvida utilizando a linguagem Java.

1. Introdução

Hoje em dia, devido ao grande crescimento das redes de computadores e das aplicações que as utilizam, existe um grande número de algoritmos de criptografia e sistemas de segurança que surgiram com a necessidade de preservar as informações que trafegam por essas redes [25][29].

Esses algoritmos se diferem quanto ao objetivo (confidencialidade, integridade, autenticação, etc.), nível de segurança e desempenho [23][16]; com isso, quando uma conexão segura tiver de ser estabelecida, há a necessidade de escolher o conjunto de algoritmos que deverá ser responsável pela segurança das informações. Essa escolha não é simples, pois como descrito anteriormente, existe um grande número deles, cada um com suas características e objetivos. Além de definir o algoritmo, existe também a necessidade de definir uma chave ou conjunto de chaves para essa comunicação, o que também impõe a necessidade de escolher a configuração necessária para elas (tamanho em bits e outros parâmetros utilizados na escolha de chaves).

Tendo em vista a complexidade em escolher o conjunto algoritmos/chaves para a conexão segura a ser estabelecida, foi proposto um sistema que avalie as características necessárias para a comunicação, tais como: o desempenho, o nível de segurança desejado, a velocidade da rede utilizada, o tempo de processamento disponível nas estações e outros parâmetros; e com base nesses parâmetros e nas características presentes nos algoritmos seja apresentado um conjunto que melhor satisfaça as condições estabelecidas pelo ambiente para a conexão segura desejada.

Outro fator importante para esse sistema é manter a possibilidade de alternar os algoritmos e as chaves dinamicamente, já que vários parâmetros que influenciam na escolha são dinâmicos e o conjunto escolhido no início da conexão pode não ser o mais indicado em outros instantes da conexão.

Os sistemas que existem atualmente permitem a escolha do algoritmo e da chave no início da conexão e utilizam este conjunto durante todo o tempo de conexão. O protocolo SSL [11][9] por exemplo, define o conjunto algoritmo/chave somente no início da conexão; outro fator importante é que esta escolha não considera alguns parâmetros importantes que afetam o desempenho, como por exemplo, o tempo de CPU necessário para executar um determinado algoritmo criptográfico em conjunto com a execução da aplicação.

Um outro exemplo de sistema que realiza a escolha do conjunto algoritmo/chave somente no início da conexão é a middleware CriptoQoS apresentada por Meylan [17]. No entanto esta escolha considera alguns parâmetros importantes (utilização de CPU, utilização de memória, utilização de interface de rede, utilização

de disco), inclusive parâmetros de QoS [22] [27] [30] [31] (Quality of Service – Qualidade de Serviço).

Neste contexto, este trabalho apresenta um sistema com diversos algoritmos criptográficos, capaz de realizar a troca destes algoritmos e chaves em tempo de execução. Este sistema também é capaz de realizar testes em tempo de execução para verificar qual o algoritmo mais rápido em um determinado ambiente e escolhe este algoritmo para o restante da conexão, obtendo assim, um melhor desempenho do sistema como um todo.

A seção 2 apresenta uma análise entre desempenho e segurança realizada durante a conferência para a escolha do novo algoritmo padrão de criptografia simétrica (AES); a seção 3 apresenta alguns trabalhos envolvendo QoS e segurança relacionando-os com a proposta do SICO; a seção 4 apresenta o SICO descrevendo sua arquitetura e protocolo utilizado; a seção 5 discute os resultados obtidos através da utilização do sistema; finalmente a seção 6 apresenta as conclusões e possíveis trabalhos futuros.

2. Aspectos de Desempenho x Segurança de Algoritmos Simétricos

Durante a conferência para a escolha do substituto do algoritmo DES [1][10][15][19] foi realizada uma comparação entre o desempenho e a segurança dos algoritmos de criptografia concorrentes. Através desta comparação foi possível identificar que nem sempre o algoritmo mais rápido é o mais seguro.

Isso motiva o estudo para verificar qual o melhor algoritmo a ser utilizado em uma determinada situação. Desta forma, foi proposto um sistema inteligente de comunicação, capaz de realizar esta escolha de acordo com alguns parâmetros provenientes do ambiente.

Um estudo apresentado por Biham [2] mostra dados relacionados à quantidade de ciclos necessários para a execução de determinado algoritmo utilizando o número padrão de iterações, comparando com a quantidade de ciclos necessários para executar o número de iterações que forneça o mínimo de segurança aceitável.

A Tabela 1 apresenta esses dados onde pode ser observado que o algoritmo mais rápido para o mínimo de segurança é o algoritmo Twofish [24]. Por outro lado, o algoritmo que precisou de uma menor quantidade de iterações mantendo um bom desempenho foi o algoritmo Rijndael [8], que foi o vencedor do concurso e conseqüentemente escolhido para se tornar o padrão AES [1].

Tabela 1. Número mínimo de iterações por algoritmo e desempenho [2]

	Original		Valores para segurança mínima	
Algoritmo	Ciclos	Iterações	Iterações	Ciclos
TWOFISH	1254	16	10	784
SERPENT	1800	32	17	956
MARS	1600	32	20	1000
RIJNDAEL	1276	10	8	1021
CRYPTON	1282	12	11	1175
E2	1808	12	10	1507
RC6	1436	20	21	1508
CAST-256	2088	48	40	1740
SAFER+	4424	8	7	3871
DFC	5874	8	9	6608
DEAL	8950	6	10	14917
LOKI97	6376	16	38	15143
MAGENTA	23136	6	11	42508
FROG	2182	8	--	--
HPC	2602	8	--	--

3. Suporte à Segurança Oferecido pelos Sistemas de QoS

Como descrito anteriormente, existem vários projetos que objetivam estabelecer conexões seguras, no entanto há uma grande dificuldade para a escolha dos algoritmos a serem utilizados uma vez que existem muitos algoritmos que variam em nível de segurança, objetivo e desempenho. Com isto existem alguns projetos que consideram parâmetros para a escolha dos algoritmos/chaves a serem utilizados nas conexões.

Nesta seção são examinados os sistemas apresentados por Guelfi [13] e Meylan [17] que são sistemas de QoS e possuem enfoque na segurança.

No projeto proposto por Guelfi [13] é apresentada uma Middleware de comunicação para o gerenciamento de serviços multimídia a qual é composta por vários módulos para prover QoS e segurança em uma transmissão multimídia.

Nesta Middleware, o módulo responsável pela segurança é chamado de Gerente de Segurança e tem a função de prover serviços como autenticação, não-repúdio, confidencialidade e integridade. O módulo Gerente de Segurança possui um componente chamado de Gerente de PKI para o controle de certificados digitais.

Segundo Guelfi [13], o nível de segurança provido por esse módulo deve ser variável e dependente da demanda de usuários e do valor da informação. No momento em que uma conexão estiver sendo estabelecida e for verificado que os parâmetros de segurança fornecidos pelo usuário informam a necessidade de criptografia, será feita uma análise se

os parâmetros solicitados (por exemplo, algoritmo escolhido e tamanho de chave) são viáveis de serem utilizados naquela conexão. O autor sugere que esta análise seja feita com base em parâmetros como largura de banda, tempo de atraso e utilização da CPU.

O trabalho apresentado por Meylan [17] também apresenta uma integração entre QoS e segurança o que resultou em um sistema chamado CriptoQoS. Uma característica da CriptoQoS é a presença de elementos chamados de agentes que capturam informações sobre os recursos computacionais disponíveis nas estações. Essas informações são importantes, pois quando um novo pedido de conexão for realizado elas deverão ser consultadas para autorizar ou não a nova conexão.

Para a escolha do algoritmo de criptografia a ser utilizado, bem como o tamanho da chave, existe um elemento chamado Gerente de Políticas. Este elemento estabelece as regras para segurança (autenticação e confidencialidade) e os parâmetros de QoS requeridos.

No entanto, os serviços de segurança são providos por um elemento chamado Gerente de Segurança. Este elemento é responsável em fornecer autorização para o estabelecimento de uma nova conexão e em implementar os serviços de criptografia e certificação digital.

Quando uma conexão for solicitada, o Gerente de Segurança deverá identificar os parceiros da comunicação. Estas informações são então repassadas ao Gerente de Políticas que definirá se a conexão será aceita e quais os parâmetros necessários para ela. De acordo com a política estabelecida pelo Gerente de Políticas, o Gerente de Segurança realizará a escolha dos algoritmos, tamanho das chaves e mecanismos de autenticação, porém essa seleção é realizada pelo usuário de uma forma estática (antes de realizar a conexão).

Estes projetos sugerem e exemplificam a necessidade de se ter um sistema que consiga acompanhar em tempo real as diferentes características estabelecidas entre as conexões, porém não chegam à fase de implementação deste acompanhamento em nível de segurança, mas sim em nível de QoS. Interessante notar que estes projetos utilizaram poucos algoritmos e não consideraram a tecnologia física de rede e protocolos.

Os sistemas apresentados por Guelfi e Meylan são sistemas voltados para a qualidade de serviço que também visam manter a segurança do sistema, no entanto, nestes sistemas é considerado um único algoritmo de criptografia. Seria interessante que esses sistemas tivessem outras implementações e realizassem adaptações também quanto ao algoritmo utilizado em determinada situação. O SICO possui a implementação de vários algoritmos de criptografia e permite uma adequação do melhor algoritmo para determinada situação e a troca dinâmica de algoritmos criptográficos e chaves, no entanto não são considerados aspectos de qualidade de serviço.

4. O Sistema Inteligente de Comunicação (SICO)

Tendo em vista a dificuldade de escolha do algoritmo para executar em determinada situação, foi proposto um sistema que consiga identificar os principais fatores relacionados com esta escolha, e desta forma determinar um melhor conjunto de algoritmos para utilizar durante uma determinada conexão e estabelecer a troca de informações de maneira segura e eficiente. Este sistema foi então nomeado SICO.

Inicialmente foi desenvolvido um protótipo do SICO em linguagem C [4][5][7]. Neste protótipo existe um módulo responsável pela comunicação utilizando Sockets. A implementação desse protótipo inicialmente permitiu que uma estação somente envie dados cifrados e a outra receba esses dados e realize a decifragem.

4.1. Aspectos Relativos ao Sistema

A partir da implementação do primeiro protótipo em linguagem C foi possível verificar as vantagens e desvantagens relacionadas com a linguagem utilizada. A implementação em linguagem C apresentou um bom desempenho e atendeu aos requisitos iniciais propostos.

No entanto, houve grandes dificuldades quanto à criação e sincronização de *threads* fazendo com que o sistema ficasse restrito a um único sistema operacional, no caso o Windows 98. Outro problema, também relacionado com o uso de múltiplas *threads*, é a complexidade para controlar e sincronizá-las para o sistema completo, onde a comunicação ocorre nas duas direções utilizando os recursos disponíveis pela linguagem C[28].

Desta forma, para a implementação da arquitetura completa, optou-se por utilizar a linguagem Java que possui recursos que facilitam a criação e controle das *threads* e não exige chamadas diretas ao sistema operacional, uma vez que a máquina virtual Java é responsável pelo controle do fluxo de execução, o que poderia impossibilitar a migração do SICO para diferentes sistemas operacionais.

Outra vantagem apresentada pela linguagem Java é a programação orientada a objetos, que facilita a estruturação do sistema como um todo e viabiliza a

adição de novos recursos ao sistema sem grandes alterações. Seguindo a abordagem orientada a objetos também é possível definir uma API (*Application Programming Interface*) em Java que permita que sejam desenvolvidas aplicações utilizando o SICO[6].

4.2. Arquitetura do SICO

Inicialmente foram estudadas quais as principais classes necessárias para melhor organizar o uso dos algoritmos de criptografia no SICO, uma vez que uma característica principal do sistema é alternar os algoritmos dinamicamente, exigindo assim que as classes que implementam estes algoritmos possuam métodos padrões. Adicionalmente, foi construída uma ferramenta para integrar vários algoritmos criptográficos e validar a estrutura de classes criada, esta ferramenta foi chamada de WEBCry [18].

Como descrito anteriormente, esta implementação em Java deverá realizar a comunicação nas duas direções. Desta forma, o sistema foi dividido em duas partes: cliente e servidor. Inicialmente o cliente requisita ao servidor uma conexão utilizando um determinado nível de segurança; o servidor então atende a requisição e inicia-se a comunicação segura utilizando um protocolo que permite a troca de chaves e algoritmos.

Nesta arquitetura é necessário um módulo específico responsável pelo protocolo de comunicação, pois é necessário o envio de sinais de controle que indicam troca de chave ou algoritmo. Também são necessários módulos específicos para o controle de decisão sobre a troca de algoritmo e a troca de chaves de maneira segura.

A Figura 1 apresenta a arquitetura geral do SICO. Através dela pode-se visualizar os módulos Protocolo, Key Exchange (para realizar trocas de chaves de maneira segura) e Controle de Decisão.

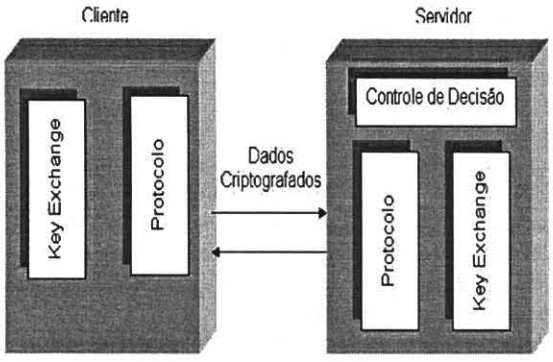

Figura 1. Esquema Geral do SICO

Para o correto funcionamento do SICO foi necessário definir e implementar um protocolo para a troca de informações relativas à chave e algoritmos utilizados. Com isso foi possível implementar a troca de chaves e algoritmos dinamicamente, entretanto, inicialmente essa troca de chaves ocorre sem um algoritmo específico para distribuição de chaves, o que diminui a segurança do sistema.

Foi também realizado um levantamento quanto às implementações dos algoritmos de criptografia existentes e as implementações escolhidas para a inserção no SICO são as versões do DES e do AES presentes na JCA [14][26] e as implementações do RC6 [20], Serpent [21], Twofish [24], Mars [3] e AES [8] desenvolvidas por [12], inseridas no SICO por meio da utilização de JNI. Para diferenciar entre as duas versões presentes do algoritmo AES, a primeira versão (utilizando JCA) é denominada de AES; e a segunda versão [12] é denominada JAES.

5. Análise de Desempenho

Foram realizados testes para verificar o impacto que a troca dinâmica de chaves e algoritmos causa para o sistema.

Para os testes de desempenho, foram utilizados três ambientes: no primeiro foi utilizado um computador Athlon XP 2400 (2.0 Ghz) com 256 Mb de memória RAM (*Random Access Memory*) com o SICO executado localmente; no segundo ambiente foi utilizado um computador Pentium 4 – 2.8 Ghz e 512 Mb de memória RAM com o SICO executado localmente; o terceiro ambiente foi formado por dois computadores Pentium 4 – 2.8 Ghz e 512 Mb de memória RAM com um computador executando a parte cliente e outro a parte servidor do SICO. Para simplificar, estes ambientes são descritos como: Athlon Local, P4-Local e P4-Remoto, respectivamente.

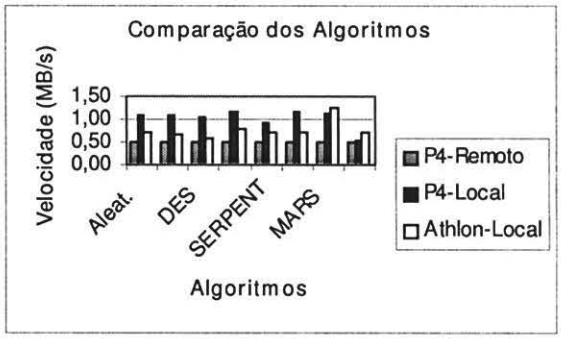

Figura 2. Comparação dos Algoritmos no SICO

Com isto foi definido como conjunto de testes o uso do SICO para a transmissão de um arquivo de vídeo com o tamanho de 108 Mbytes (113.839.120 bytes) utilizando os algoritmos descritos na seção 5.2.

A Figura 2 mostra uma comparação entre os algoritmos implementados relacionando seu desempenho em cada um dos ambientes de teste definidos. Pode-se perceber que o P4-Local obteve quase sempre o melhor desempenho, entretanto, não foi superior a 1,2 Mbytes/s. Esta lentidão pode estar relacionada ao tempo de comunicação existente no SICO, desta forma, foi realizado um estudo para verificar possíveis otimizações e foi encontrado um possível fator para tal lentidão.

Este fator é o tamanho do buffer de recepção utilizado pelo sistema, que no caso da Figura 2 estava configurado com 64 Kbytes. Com isso, foram realizados testes variando o tamanho do buffer entre 64 Kbytes e 8 Mbytes para verificar qual seria o tamanho que melhor atendesse as necessidades do SICO. Estes testes estão listados na seção 5.1, já na seção 5.2 são apresentados testes mais detalhados utilizando o tamanho de buffer mais adequado.

5.1. Variação do tamanho do buffer

Para resolver o problema da lentidão do sistema, foram realizados testes variando o tamanho do buffer até que se chegasse ao valor adequado. Nestes testes não foram consideradas as políticas de troca de chave e algoritmo e sim definido um intervalo de tempo em que as trocas devem ocorrer. Foram então definidos testes variando algoritmos e chaves aleatoriamente dentro deste intervalo de tempo.

As Figuras 3, 4 e 5 apresentam a relação entre o tamanho do buffer e o desempenho do sistema verificado através da velocidade média de transmissão. Pode-se perceber que à medida que o tamanho do buffer é incrementado a velocidade média do SICO aumenta. Entretanto, a partir de 1024 Kbytes o desempenho parece estabilizar; e em alguns casos até diminui. Importante destacar que no caso da Figura 5 a tendência é um pouco diferente do que ocorre com os testes locais não demonstrando diminuição do desempenho a medida que o tamanho do buffer aumenta. Com isso, definiu-se como padrão para o SICO o tamanho de buffer de 1024 Kbytes uma vez que este valor permite obter um bom desempenho em ambos os casos.

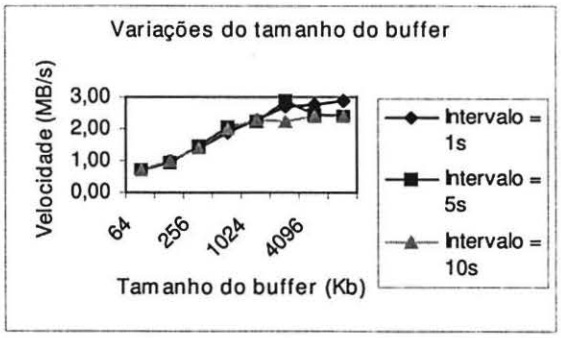

Figura 3. Variações do tamanho do buffer - Athlon-Local

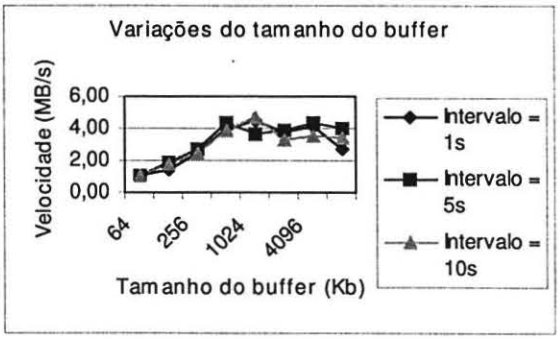

Figura 4. Variações do tamanho do buffer – P4-Local

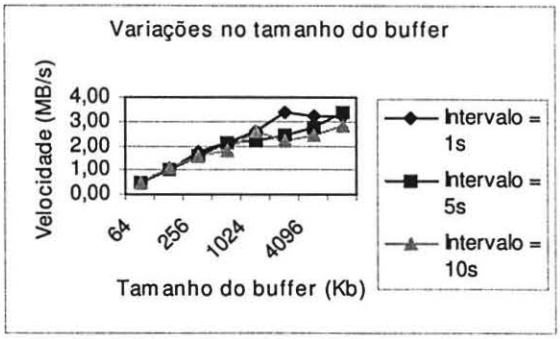

Figura 5. Variações do tamanho do buffer – P4-Remoto

5.2. Testes com o buffer de 1024 Kbytes

Nestes testes ainda não foram consideradas as políticas de troca de chave e algoritmo e sim definido um intervalo de tempo em que as trocas devem ocorrer. Foram então definidos testes com algoritmos fixos variando somente a chave e testes variando chaves e algoritmos aleatoriamente.

As Figuras 6, 7 e 8 apresentam os gráficos que relacionam o intervalo de troca de chaves com o desempenho geral do SICO. Pode-se perceber que à medida que aumenta o intervalo entre as trocas, o número de trocas de chaves diminui, no entanto, o desempenho do sistema não é muito afetado. Isto significa que, ao contrário do que se imagina, neste contexto é viável executar quantas trocas de chave forem necessárias. Desta forma, pode-se concluir que é possível utilizar a troca de chaves e algoritmos dinamicamente sem sofrer grandes perdas de desempenho.

Outro detalhe importante a se perceber observando as Figuras 6, 7 e 8 é que o algoritmo com melhor performance varia para cada caso. Isto mostra a importância da escolha correta do algoritmo.

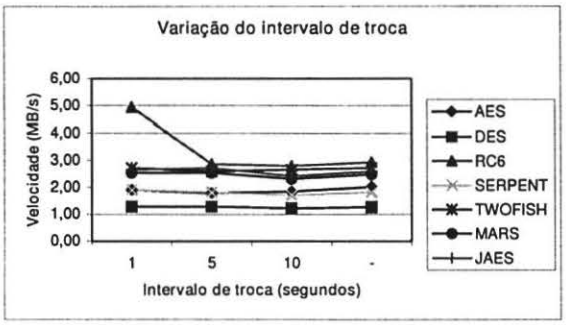

Figura 6. Variação do intervalo de troca – Athlon-Local

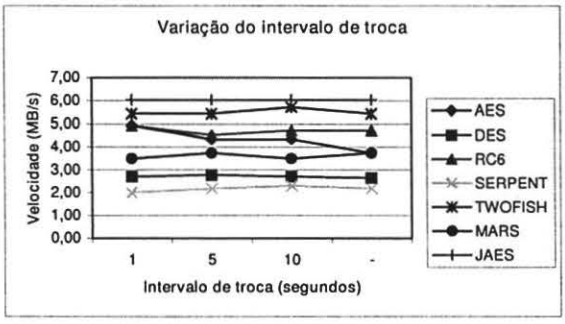

Figura 7. Variação do intervalo de troca – P4-Local

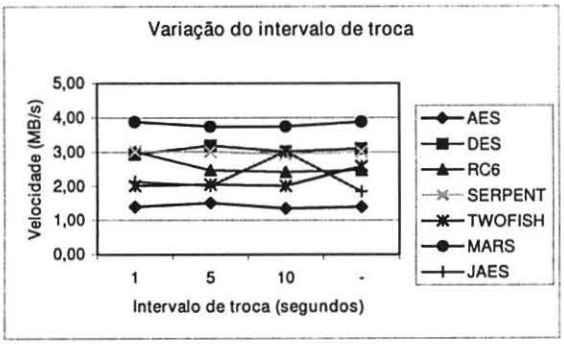

Figura 8. Variação do intervalo de troca – P4 Remoto

Desta forma, foi proposta e implementada uma maneira de se escolher o melhor algoritmo no inicio de uma conexão. Esta escolha foi realizada através de uma análise dos desempenhos dos algoritmos em tempo de execução. Os resultados desta política são apresentados na seção 5.3.

5.3. Testes realizados utilizando políticas para a troca dos algoritmos

Com base nos resultados anteriores, foi possível identificar que não há um impacto significativo quanto ao número de trocas de chaves e algoritmos. Desta forma foi proposto realizar testes em tempo de execução com todos os algoritmos implementados e verificar qual obteve o melhor desempenho. Após definir o algoritmo com melhor desempenho, o sistema o utiliza, variando somente as chaves utilizadas.

Para verificar qual o algoritmo possui o melhor desempenho no contexto em que está sendo executado, cada algoritmo é utilizado durante o intervalo de 1,5 segundos e é verificado qual algoritmo processou a maior quantidade de dados. Considerando que existem sete algoritmos implementados, o tempo total para testá-los é de 10,5 segundos. Como o teste é executado ao mesmo tempo em que o sistema já está ativo e processando as informações, não é necessário um tempo extra para esta escolha.

A Figura 9 mostra o gráfico comparando o desempenho obtido com essa política e o obtido trocando os algoritmos e chaves aleatoriamente. Pode-se perceber que esta política obteve sempre o melhor resultado. Entretanto, quando comparamos essa política com os resultados obtidos utilizando somente o RC6, o JAES e o MARS, que foram os mais rápidos para estes ambientes, pode-se perceber que nem sempre a política obteve um melhor resultado.

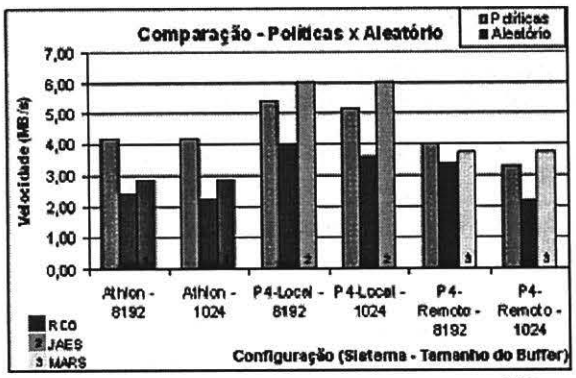

Figura 9. Testes realizados com uma política para troca de algoritmos

Mas ainda assim, para alguns casos a política ainda obteve um melhor resultado. Isto pode ocorrer pois o ambiente de execução é dinâmico e o algoritmo criptográfico que foi definido como o mais rápido em um teste da política pode não ser o melhor algoritmo em outros instantes, ou seja, o algoritmo mais rápido no início pode não se manter sempre como o mais rápido. Motivo pelo qual seria interessante fazer mais testes da política de escolha do algoritmo durante uma conexão.

A Figura 10 apresenta a relação entre o tempo total de execução do SICO para este conjunto de testes e o tempo gasto para a escolha do melhor algoritmo a ser utilizado. Pode-se perceber que em alguns casos, o tempo para a escolha chega a ser 50%, no entanto, quando se compara com os resultados obtidos na Figura 9 verifica-se que ainda assim há uma vantagem na utilização desta política.

É importante destacar que para esta política não foram considerados aspectos relacionados com a segurança necessária. Para trabalhos futuros, pretende-se incluir métodos que verifiquem junto ao usuário o nível de segurança desejado, e que essa informação reflita para o algoritmo final escolhido.

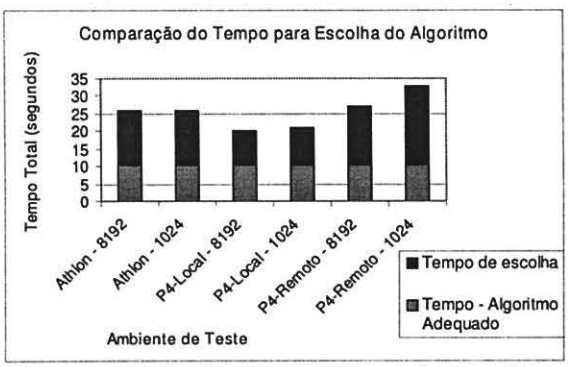

Figura 10. Relação entre o tempo de escolha e o tempo total

6. Conclusões e Trabalhos Futuros

Foi apresentado um sistema seguro para a transmissão de dados capaz de escolher através de alguns parâmetros, qual o melhor algoritmo criptográfico a ser utilizado em um determinado ambiente. Foram então realizados testes de desempenho para verificar se o sistema realmente atendia as necessidades impostas para uma comunicação rápida e segura. Através destes testes foi possível observar que o algoritmo escolhido para a comunicação possui uma grande influência para o desempenho total do sistema.

Desta forma, foi então proposto um método para testar o desempenho dos algoritmos implementados em tempo real, sendo que, após este teste o sistema poderia tomar a decisão de escolha do algoritmo criptográfico que mais se adaptou ao ambiente de execução. A implementação deste método permitiu um aumento de até 50% no desempenho total do sistema, do que quando comparado com uma escolha aleatória.

Futuramente, interessante a realização de testes utilizando aplicações que necessitem de grande quantidade de recursos de rede e processamento, como por exemplo, aplicações de vídeo em tempo real.

Referências

[1] Aeswinner. National Institute of Standards and Technology, NIST. <www.nist.gov/public_affairs/releases/g00-176.htm>. Acesso em 12 Dez 2003.

[2] Biham, E. A Note on Comparing the AES Candidates. Second AES Conference. Technical report, Computer Science Department, Technion. Israel Institute of Technology, 1999.

[3] Burwick, C. et al, MARS – A candidate cipher for AES. IBM Corporation, USA, 1999.

[4] Chiaramonte, R. B., Implementação e Teste em Hardware e Software de Sistemas Criptográficos. Trabalho de Conclusão de Curso (Apoio FAPESP nº 00/14166-8). UNIVEM – Centro Universitário Eurípides de Marília – Marília, 2003.

[5] Chiaramonte, R.B.; Moreno, E.D., Criptografia Posicional em Hardware (VHDL e FPGAs). Revista REIC-SBC (Revista Eletrônica de Iniciação Científica), Sociedade Brasileira de Computação . Ano II, Vol. II, No. IV, Dez. 2002, ISSN: 1519-8219.

[6] Chiaramonte, R.B.; Moreno, E.D., Sico - Um Sistema Inteligente e Adaptativo Para Comunicação de Dados: Resultados Preliminares; In: GCETE - Global Congress on Engineering and technology Education, Bertioga, 2005.

[7] Chiaramonte, R.B.; Pereira, F.D.; Moreno, E.D.; Um Algoritmo Criptográfico Posicional – Otimizações e Desempenho; Revista de Engenharia de Computação e Sistemas Digitais; ISSN: 1678-8435; número 2 – Novembro 2005.

[8] Daemen, J.; Rijmen, V. AES Proposal: Rijndael Block Cipher. 03/09/99

[9] Dierks, T.; Allen, C., The TLS Protocol Version 1.0. RFC 2246 –Janeiro 1999.

[10] FIPS-PUB46-3, "DATA ENCRYPTION STANDARD (DES)", Federal Information Processing Standard (FIPS), NIST, 1999

[11] Freier, A.O.; Karlton, P.; Kocher, P.C., The SSL Protocol Version 3.0. Internet Draft –Disponível em: http://wp.netscape.com/eng/ssl3/draft302.txt. Acesso: 04/2004

[12] Gladman, B. AES Second Round Implementation Experience. Disponível em: <http://fp.gladman.plus.com/cryptography_technology/aesr2/index.htm>. Acesso em: 10 fev 2006

[13] Guelfi, A. E., Middleware de Comunicação para Prover o Gerenciamento de Serviços Multimídia Flexíveis e Integrados em Ambientes Distribuídos e Heterogêneos. Tese de Doutorado. EPUSP – Escola Politécnica da Universidade de São Paulo – São Paulo, 2002.

[14] JCA. JavaTM Cryptography Architecture API Specification & Reference. Disponível em <http://www.wedgetail.com/jcsi/provider/>. Acesso em novembro de 2004.

[15] Matsui, M., Linear Criptanalysis Method for DES Cipher. In advances in Criptology, Eurocrypt 93, Lectures Notes in Computer Science Vol. 765, Springer Verlag, pp. 386-397, 1993.

[16] Menezes, A.; Oorschot, P.V.; Vanstone, S. Handbook of Applied Cryptography. New York: CRC Press, 1997.

[17] Meylan, F., CriptoQoS: Uma Plataforma de Gerenciamento e Desenvolvimento de Aplicações Distribuídas com Suporte Integrado à Qos e Segurança. Tese de Doutorado. EPUSP – Escola Politécnica da Universidade de São Paulo – São Paulo, 2003.

[18] Moreno, E.D.; Pereira, F.D.; Chiaramonte, R.B., Criptografia em Software e Hardware. São Paulo, Novatec Editora, 2005.

[19] Pereira, F.D.; Moreno, E.D. Otimização em VHDL e Desempenho em FPGAs do Algoritmo de Criptografia DES. Quarto Workshop em sistemas computacionais de alto desempenho (WSCAD). São Paulo, 2003.

[20] Rivest, R. RC6 Block Cipher. RSA Security Inc, USA, 2000.

[21] Ross, A.; Biham, E.; Knudsen, L. Serpent: A Proposal for the Advanced Encryption Standard. Cambridge: Cambridge University, 1998.

[22] Santos, A.P.S. dos, "Qualidade de Serviço na Internet", Boletim bimestral sobre tecnologia de redes – RNP (Rede Nacional de Ensino e Pesquisa), Volume 3, Número 6, ISSN 1518-5974, 12 de novembro de 1999. Disponível em: http://www.rnp.br/newsgen/9911/qos.html

[23] Schneier, B. Applied Cryptography: Protocols, Algorithms and Source in C. 2nd ed. New York: John Wiley and Sons, 1996.

[24] Schneier, B.; Kelsey, J.; Whiting, D.; Wagner, D.; Hall, C.; Ferguson, N., "Twofish: A 128-Bit Block Cipher", Disponível em: <http://www.schneier.com/paper-twofish-paper.html>, 1998. Acesso em: Nov de 2005.

[25] Stallings, W. Cryptography and Network Security - Principles and Practice, Second Edition. Prentice Hall, 1998

[26] Sun - Glossary of Terms. Site da Web. Consulta realizada em Dezembro 2005: http://java.sun.com/docs/books/tutorial/information/glossary.html

[27] Tanenbaum, A. S. "Computer Networks", Third Edition, Prentice Hall, New Jersey, 1996;

[28] Tanenbaum, Andrew S.; Woodhull, Albert S.. Sistemas operacionais : projeto e implementação. 2ª ed. Porto Alegre: Bookman, 2000. 759p.

[29] Terada, R. Segurança de Dados – Criptografia em Redes de Computadores, 1ª ed. Edgard Blücher, 2000.

[30] Walrand, J.; Varaiya, P.; "High-Performance Communication Networks", Second Edition, Academic Press, San Diego, 2000;

[31] Xiao, X., e Ni, M. N., "Internet QoS: A Big Picture", IEEE Network, March/April 1999