REVISTA CIENTÍFICA ELETRÔNICA DE SISTEMAS DE INFORMAÇÃO - ISSN 1807-1872
PUBLICAÇÃO CIENTÍFICA DA FACULDADE DE CIÊNCIAS JURÍDICAS E GERENCIAIS DE GARÇA/FAEG
ANO II, NÚMERO, 04, FEVEREIRO DE 2006. PERIODICIDADE: SEMESTRAL

MÓDULO DE SEGURANÇA EM FPGAS USANDO O PADRÃO PKCS#11 E CRIPTOGRAFIA RSA

MUZZI; Fernando Augusto Garcia
TAMAE; Rodrigo Yoshio
ROSA; Adriano Justino
Docentes da Faculdade de Ciências Gerenciais e Jurídicas de Garça – FAEG/Garça
fagmuzzi@yahoo.com.br ; rytamae@yahoo.com.br; adriano@faef.br

RESUMO

Neste projeto implementou-se a especificação do PKCS#11 em hardware, utilizando o VHDL e PFGA. Foi implementado por meio de uma máquina de estados finito, usando o algoritmo RSA, formando o projeto modular e facilmente adaptável a expansões futuras para a comunicação entre máquina e dispositivos. Qualquer algoritmo de criptografia pode ser implementado dentro da especificação PKCS#11 e neste projeto implementou-se o algoritmo RSA. Além da implementação da FSM foi possível realizar implementação e testes usando chave do RSA de 56 bits, 128 bits, 256 bits e 512 bits.
Palavras-chave: PKCS#11; hardware

ABSTRACT

In this paper we have designed the PKCS#11 specification on hardware, specifically, for using VHDL and FPGAs. It was implemented through a finite state machines and it works together to RSA algorithm which also was implemented for us into the FPGAs. Our implementation is modular, it is easily adaptable to future expansions for the communication between machine and devices, thus any cryptography algorithm can be implemented inside of the specification PKCS#11. Our results were obtained for the RSA algorithm working with 56, 128, 256 and 512 bits.
Keywords: PKCS#11; hardware

1. INTRODUÇÃO

O grande gargalo nas comunicações tem sido os equipamentos de rede, principalmente os roteadores. Eles centralizam o tráfego de pacotes e muitas vezes são os responsáveis pela falta de eficiência da rede. As

tecnologias de rede têm evoluído, principalmente com os avanços da internet e os roteadores precisam aumentar sua capacidade de processamento dos pacotes, para evoluir juntamente com as demais tecnologias e deixar aos pouco, de ser um dos gargalos das comunicações.

A falta de segurança na transmissão de dados e pacotes na rede é muito grande, por isso é necessário a criação de mecanismos de segurança para rede. Como por exemplo algoritmos criptográficos, protocolos, entre outros. Essas soluções podem ser implementadas em software ou em hardware. Recentemente, uma tendência é ter processadores de rede, onde a inserção de criptografia é fundamental para a transferência segura de pacotes na rede.

Existem diversas soluções de segurança baseadas em criptografia, porém a maior parte dos estudos atuais se concentram em processadores de rede (NPs). Soma-se a isso que há soluções já padronizadas (tais como o *Public Key Cryptography Standards, PKCS*), porém não são relacionadas com os NPs nem com implementações em hardware.

Os processadores de rede têm fundamental importância para a comunicação de dados, já que a velocidade de transmissão de dados têm aumentado consideravelmente. No entanto é necessário segurança para os processadores de rede, ou seja, a criptografia de pacotes, para garantir a segurança da informação. Com a implementação em hardware das especificações do PKCS#11 que propiciará a criptografia, um processador de rede poderá receber e transmitir pacotes criptografados seguindo o padrão.

A implementação de um algoritmo de criptografia baseado na norma PKCS#11 é importante para segurança dos dados (pacotes que trafegam na rede), e pode ser implementado em hardware, especificadamente em FPGAs.

O padrão PKCS#11 é utilizado como interface para invocar operações criptográficas em hardware e é utilizado para prover suporte aos *tokens*.

Importante salientar que o PKCS#11 é um padrão criado para interagir com hardware podendo ser utilizado em rede, baseado em segurança de dados que utilizam como base o algoritmo de criptografia RSA.

A principal contribuição deste artigo é apresentar detalhes de nossa implementação em hardware (FPGAs) do padrão PKCS#11. Os resultados usando o algoritmo RSA mostram a viabilidade do projeto e a possibilidade de inserir módulo em outros protótipos em hardware.

2. O PADRÃO PKCS

O PKCS Padrão de criptografia de Chave Pública (*Public Key Cryptography Standards*) é uma série de especificações produzidas pelos Laboratórios RSA em cooperação com desenvolvedores de sistemas de segurança de várias partes do mundo, que visa acelerar, por meio da padronização, a utilização e o desenvolvimento de algoritmos de chave pública (RSA, 2002).

O padrão PKCS#11 surgiu em 1991, como resultado de encontros de um pequeno grupo de precursores no uso da tecnologia de chave pública e desde então tem se tornado referência até mesmo para padrões já estabelecidos, como ANSI X9, PKIX, SET, S/MIME e SSL. Atualmente seu desenvolvimento ocorre basicamente através de lista de discussões e workshops ocasionais.

Tendo sido realizado pela empresa possuidora da patente do RSA, o sistema de chave pública descrita nesses padrões é basicamente o próprio algoritmo RSA.

Atualmente existem doze padrões deste tipo: PKCS#1, #3, #5, #6, #7, #8, #9, #10, #11, #12, #13 e #15 (RSA, 2002). Os objetivos da RSA na publicação destes padrões são (RSA, 2002):

Os PKCS visam preencher o vazio que existe nas normas internacionais relativamente a formatos para transferência

3. O ALGORITMO RSA

O RSA é um sistema de criptografia de chave assimétrica ou criptografia de chave pública que foi inventado por volta de 1977 pelos professores do MIT (*Massachsetts institute of Technology*) Ronald Rivest, Adi Shamir e o professor Leonard Adleman da USC (*University of Southern Califórnia*) (RSA, 2002).

O sistema consiste em gerar uma chave pública (geralmente utilizada para cifrar os dados) e uma chave privada (utilizada para decifrar os dados) através de números primos grandes, o que dificulta a obtenção de uma chave a partir da outra.

O algoritmo RSA usado para a geração da chave pública e privada usadas para cifrar e decifrar as mensagens são simples. (RSA, 2002) (CHIARAMONTE, 2003).

4. ESPECIFICAÇÃO DO PKCS#11

O algoritmo do PKCS#11 da RSA foi implementado em linguagem C (RSA, 2002).

A maioria das chaves utilizadas nos dias de hoje tem 1024 bits de comprimento (RSA, 2002) e quando a implementação é em hardware, usa-se a norma PKCS#11, que é o mais utilizado em interfaces API *(Application Programming Interface*) para módulo criptográfico. Este padrão especifica uma interface (API), chamado Cryptoki, para dispositivos que fazem segurança usando criptografia e executam funções de criptografia usando uma chave que é chamada de crypto-chave.

A biblioteca Cryptoki é planejada para dispositivos de criptografia associados com um único usuário, assim são omitidas algumas características que poderiam ser incluídas em uma interface de propósito geral. Por exemplo, Cryptoki não tem usuários múltiplos e distintos (RSA, 2002)

5. O PADRÃO PKCS#11 EM HARDWARE

O padrão PKCS#11 é utilizado para operações criptográficas em hardware. O PKCS#11 é baseado no padrão que fornece recomendações para a execução de criptografia baseada em chave pública e o algoritmo é o RSA. Esta seção apresenta uma versão em hardware do PKCS#11, projetado e implementado pelos autores deste artigo.

5.1 IMPORTÂNCIA DO PKCS#11 EM HARDWARE

O padrão PKCS#11 é utilizado para operações criptográficas em hardware (tokens, smart cards e etc.) para prover suporte aos tokens.

O PKCS#11 é baseado no padrão que fornece recomendações para a execução de criptografia baseada em chave pública e o algoritmo utilizado é o RSA.

Devido a seu pioneirismo e simplicidade do algoritmo, tornaram-se padrão de fato em PKIs.

A segurança da informação se torna mais importante no mundo de hoje, e é necessário que o equipamento de networking permita funções de criptografia.

5.2 DESCRIÇÃO DA MÁQUINA DE ESTADOS FINITO PADRÃO PKCS#11

Nós realizamos o projeto e implementação do padrão PKCS#11 em hardware, especificamente em FPGA. Em nossa implementação, os estados variam do estado 0 até o estado 6, ou seja, a máquina de estado finito RSA padrão PKCS#11 é constituída de 7 estados que estão relacionados entre si. Em cada momento dependendo de um evento, seja baseado no *clock* ou no estado da máquina, um estado estará em funcionamento e passará para o estado seguinte assim que o estado anterior for completado e a respectiva condição for satisfeita.

Em nosso projeto, a máquina de estado finito padrão PKCS#11 tem inicio no estado 0 (zero). Quando a máquina está em estado 0 (zero),

ou seja, a variável estado recebe o valor 0 (zero) e a variável “CH” recebeu o valor 0 (zero). Assim, no estado 0 a máquina é inicializada.

No estado 1 são atribuídos os valores dos tokens conforme escolha do usuário, ou seja a variável VAR_CH irá receber o valor do token escolhido, aqui é a variável ESTADO recebe DOIS. Assim o estado da máquina passará para o próximo estado. No estado 2, a informação do token é enviada serialmente pela porta serial do computador.

No estado 3, o dado do token é enviado para a o módulo de criptografia que implementa o algoritmo RSA, é quando o dado é liberado para criptografia e a variável S_P_CRIP recebe o valor 1 (um) indicando que acabou a criptografia. A variável estado recebe o valor 4 indicando para passar para o estado seguinte.

No estado 4, o dado criptografado é enviado para a porta serial do computador ou seja o dado que foi criptografado no estado 3 agora é enviado pela porta serial no estado 4. A variável “Lib_est_quatro” receberá o valor 1 (um) indicando que passou pelo estado quatro e o valor D_OUT receberá o valor de S_D_CRIP que corresponde ao dado criptografado. Quando “Lib_est_quatro” recebe o valor 1 (um) indica que já enviou o dado pela porta serial do micro.

No estado 5, o dado libera a função de decriptografia RSA e se a variável S_P_DECRIP for igual a 1 (um) indica que foi realizada a decriptografia e a variável estado recebe o valor 6, indicando que a máquina irá para o próximo estado, que corresponde ao estado 6.

No estado 6, o dado decriptografado será transmitido pela porta serial do micro, D_OUT receberá o valor de S_D_DECRIP que corresponde ao dado decriptografado pelo módulo RSA. Quando isso ocorre, “Lib_Est_SEIS” recebe o valor 1 (um) indicando que o dado decriptografado foi enviado para o outro computador pela porta serial.

No estado 5 e 6 pode-se visualizar que é realizado a decifragem, mesmo após a realização de cifragem, isso ocorre para validar os testes, porque foi implementado para realizar teste em um único FPGA, podendo em projetos futuros usar dois FPGAs, um para cifrar e outro para decifrar.

A figura 1 mostra a simulação usando a máquina de estados finito na ferramenta da Xilinx 3.1 chamada FPGA Express para efetuar simulação. A variável VAR_CH recebe o dado a ser criptografado ou seja 6768 que corresponde aos bytes CD e nota-se que D_OUT no estado 2 tem uma subida de borda indicando que o dado foi enviado serialmente para outro computador. No estado 4 S_D_CRIP recebe o dado cifrado ou seja 1A68FE e D_OUT tem uma subida de borda indicando que o dado cifrado foi enviado pela porta serial para outro computador.

5.3 PADRÃO PKCS#11 EM HARDWARE

Nesta seção pode-se visualizar os resultados obtidos em hardware no padrão PKCS#11 através da Máquina de estados finito.

Pode-se notar na figura 1 que no estado 1 é mostrado que VAR_CH está no valor 6768, que corresponde ao dado que será criptografado, no estado 2 é criptografado o dado usando o RSA e enviado pela porta serial usando o sinal D_OUT. No S_D_CRIP tem-se o dado criptografado que corresponde a 1A68FE que está em hexadecimal. Os valores atribuídos representam um exemplo neste projeto.

A variável “conta” corresponde a um contador que varia de 1 até 83333 e sincroniza o clock (CLK) da máquina principal com o clock (s_clk) da saída do dado criptografado pela porta serial, denominado slot pelo padrão PKCS#11 e em nosso projeto é o slot default. Quando o valor do contador conta for igual a 166666, a variável conta recebe novamente o valor 0, para sincronizar com 50 Megahertz do FPGA que é a frequência de funcionamento no teste usado.

O reset é usado para zerar a máquina quando o reset está com o valor 0. Quando recebe o valor 1, dá inicio ao funcionamento da máquina de estados finito.

O vetor CH, de tamanho 4, corresponde ao Token, onde pode-se determinar qual será o valor do token a ser usado. Quando se escolhe o 3, por exemplo corresponde ao valor 67 e 68 que na tabela ASCII

corresponde aos bytes C e D, esse valores são exemplos de utilização no teste, podendo ser usado qualquer outro valor.

O sinal S_D_DECRIP recebe o valor decriptografado quando o estado da máquina for igual a 5.

O “contb” é um contador usado para indicar que o dado foi enviado pela serial. Quando o valor da variável “contb” for igual a 15, a variável contb é zerada, e Lib_Est_DOIS recebe o valor 1 indicando que o dado criptografado foi enviado serialmente. Quando for igual ao valor 23, indica que o dado decriptografado foi enviado pela serial, e o sinal “Lib_est_seis” recebe o valor 1, assim quando “contb” chegar a esse número, a variável “Lib_est_seis” receberá o valor 1 indicando que o dado já foi enviado serialmente.

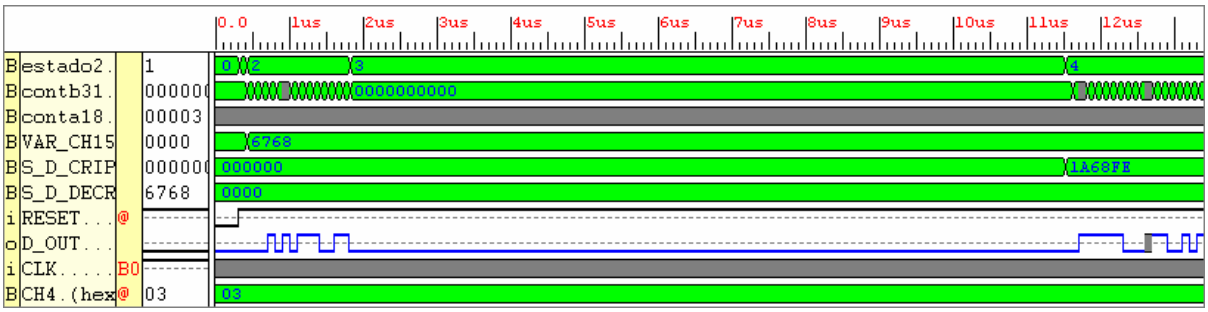

Figura 1 – Simulação do padrão PKCS#11 usando uma Máquina de estados Finito (MUZZI, 2005)

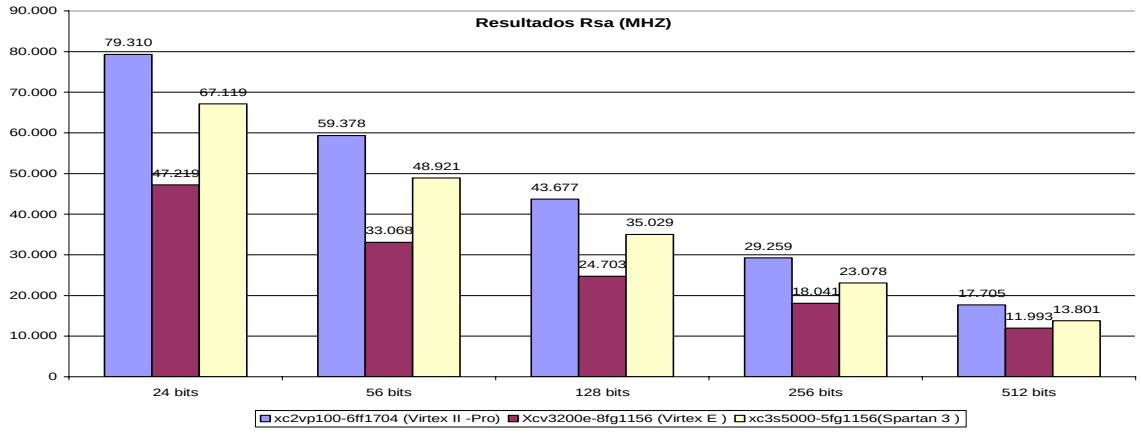

Figura 2 – Impacto do tamanho das chaves em (Bits) na velocidade do algoritmo RSA (MUZZI, 2005)

Em nosso projeto também implementamos o algoritmo RSA em hardware, em FPGA. Além disso, o nosso módulo RSA em hardware é parametrizado, permitindo mudar facilmente o tamanho da chave.

A figura 2 apresenta o impacto que o tamanho em bits das chaves do algoritmo RSA tem na ocupação do FPGA. Como se esperava, aumentando o tamanho (em bits) do algoritmo RSA, a ocupação (medida através de slice no FPGA), aumenta. Em nosso projeto foi possível verificar que mesmo a chave do algoritmo RSA com 512 bits junto com a máquina de estado na especificação do padrão PKCS#11, verifica-se que o máximo de ocupação do FPGA foi de 17,96% sendo possível usar o espaço não ocupado em outros projetos futuros, como por exemplo um processador de rede.

Nós verificamos o impacto de ter tamanhos diferentes de chaves do algoritmo de criptografia RSA, sendo usado chave 24 bits, 56 bits, 128 bits, 256 bits e 512 bits. Quanto maior o número de bits usados na chave, maior será a ocupação do FPGA.

Esta seção apresenta os resultados obtidos usando a criptografia RSA em hardware. Foi utilizado para análise dos resultados um computador Pentium IV 1.6 GHZ com 128 de memória RAM e sistema operacional Windows 2000 e o Xilinx 6.2i.

O módulo de sintaxe retorna dados que são muito importantes para o levantamento de resultados, desde a ocupação do código RSA no FPGA até o número de IOBs, Flip Flop, LUTs e a velocidade máxima, medida em MHZ, obtidos após prototipação em FPGAs.

Na figura 3 pode-se visualizar a foto de um teste real usando computador, osciloscópio e FPGA no projeto que recebe o dado do FPGA pela porta serial do computador. Primeiro é enviado o dado original escolhido, por exemplo dois bytes que corresponde às letras “DE” logo após é enviado o dado criptografado e depois o dado decriptografado.

Foram criadas três versões da máquina de estados finito - FSM, a versão 3.1, 6.1 e 6.2. A versão 3.1 foi implementada na ferramenta Xilinx versão 3.1 e simulada usando a ferramenta de síntese FPGA Express. A

versão 6.1 foi implementada no Xilinx 6.2 e finalmente a versão 6.2 foi também implementada no Xilinx 6.2.

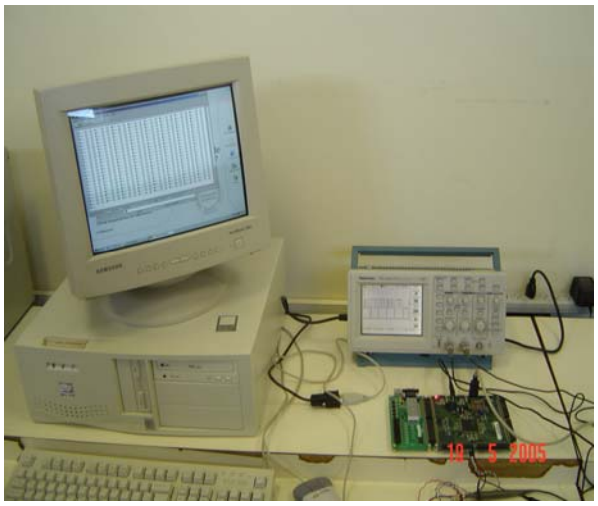

Figura 3 - Comunicação real com o PKCS#11 em hardware.

6. CONCLUSÃO

A criação de um protótipo com módulo PKCS#11 para segurança em processadores de rede é importante, já que o PKCS#11 é um padrão de segurança criado para criptografia baseada em token, podendo ser usado em software como em hardware.

A implementação do protótipo com o padrão PKCS#11 foi feita em VHDL e prototipado em hardware usando FPGA, seguindo uma implementação baseada em máquinas de estado finito. Além da análise de resultados obtidos usando criptografia em hardware, foi possível o envio de um dado criptografado usando o padrão PKCS#11 para outro computador usando um FPGA através de comunicação serial.

A máquina de estado implementada em hardware, chamada de FSM 6.2, usa o algoritmo de criptografia RSA, que também foi implementado em FPGAs, podendo ser usado qualquer um outro algoritmo de criptografia, não precisa ser específico o RSA. O importante neste projeto foi mostrar que o padrão de criptografia PKCS#11 pode ser implementado em hardware e não somente em software, enfatizando nos recursos utilizados em um determinado FPGA.

Está implícito no padrão uma série de especificações para estabelecer uma comunicação segura entre slots, comunicações baseadas em tokens, como por exemplo o login de um usuário no token e a abertura de uma sessão. Enquanto a sessão estiver aberta a comunicação entre dois slots é estabelecida, quando termina a sessão a mesma é fechada.

Obteve-se resultados significativos para demonstrar o padrão PKCS#11 usando criptografia RSA se comunicando com o slot e com outro computador. Foi possível enviar um dado original, um cifrado e um decifrado usando hardware o que torna possível maior segurança para a comunicação entre computadores e dispositivos.

Foi realizada a implementação em linguagem C e VHDL e foi possível verificar a diferença entre a implementação em software usando linguagem C e em hardware (VHDL e FPGAs), e como esperado, em hardware houve um desempenho superior.

O algoritmo RSA em C foi implementado usando 24 bits e 56 bits, não foi possível aumentar o número de bits devido às variáveis em C não aceitarem tamanho maior que 32 bits, dessa forma não foi possível implementar o RSA de uma maneira simples em C com 128 bits, 256 bits e 512 bits e 1024 bits.

Na implementação usando VHDL foi possível implementar o RSA com 56 bits, 128 bits, 256 bits e 512 bits. Não foi possível implementar 1024 bits devido a que em nossa implementação usou-se um vetor de 1024 bits, e a ferramenta Xilinx 6.2 tem restrições para vetores deste tamanho.

Para implementação em software usando o padrão PKCS#11 é possível usar uma biblioteca em C chamada cryptoki, em hardware não foi possível usar essa biblioteca sendo necessário abstrair o padrão PKCS#11 para a realização da implementação em hardware usando FPGA.

Foi usado o FPGA Spartan2E, no qual foi possível a realização de teste enviando um dado criptografado para outro computador usando uma comunicação serial e o software chamado hyperterminal do windows para verificar o dado que computador receptor recebeu. Como trabalhos futuros

sugerimos implementar o algoritmo RSA com chaves maiores, e inserir a nossa solução em um processador de rede, também prototipado em FPGAs.

7. REFERÊNCIAS BIBLIOGRÁFICAS

CHAMELEON 2000, Chameleon Systems, “CS2000 Reconfigurable Communications Processor”, Family Product Brief, 2000

CHIARAMONTE, BARROS, R. C, “Implementação e teste em Hardware e Software de Sistemas Criptográficos, 2003.

C-PORT 2002, C-Port, C5e Network Processor Product Brief, January 2002, http://www.motorola.com

CRYPTOAPI. Digital Signatures with the Microsoft Cryptoapi. Dr. Dobb´s Journal, 1998.

EZCHIP 2002, EZChip Network Processors, http://www.ezchip.com

FREITAS 2001, H. C. Freitas, C. A. P. S. Martins, “Processador de Rede com Suporte a Multi-protocolo e Topologias Dinâmicas”, II Workshop de Sistemas Computacionais de Alto Desempenho, WSCAD’2001, Pirenópolis - GO, pp.31-38

FREITAS 2000, H. C. Freitas, C. A. P. S. Martins, “Projeto de Processador com Microarquitetura Dedicada para Roteamento em Sistemas de Comunicação de Dados”, I Workshop de Sistemas Computacionais de Alto Desempenho, WSCAD’2000, São Pedro - SP, pp.63, (Iniciação Científica).

MORENO 2003, MORENO, E. et al. Projeto, Desempenho e Aplicações de Sistemas Digitais em Circuitos Programáveis (FPGAs). Bless, 2003.

MORENO 2005, MORENO, David Edward., PEREIRA, Fábio Dacêncio., CHIARAMONTE, Rodolfo Barros., Criptografia em Software e Hardware. Novatec, 2005.

MUZZI 2005, Muzzi, Fernando Augusto Garcia. O Padrão de Segurança PKCS#11 em FPGAs: RSA Um estudo de caso. Dissertação de Mestrado, Ciência da Computação, Univem, Centro Universitário Eurípides Soares da Rocha de Marília, p.134, 2005

NP4GS3, IBM PowerNP NP4GS3 Databook, http://www.ibm.com

PATTERSON 1997, PATTERSON, D. A., J. L. Hennessy, “Computer Organization and Design: The Hardware/Software Interface”, Morgan Kaufmann Publisher, 1997.

PUC, Projeto de Processador de Rede com Suporte a Multi-protocolo e Topologias Dinâmicas : http://www.inf.pucminas.br/projetos/pad-r/

RSA 2002, RSA Labs. Public Key Criptography Standards (PKCS). Version 2.1 - 2002, disponível em http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/index.html.

RSA 1999, RSA Labs. Factorization of RSA-155. http://www.rsasecurity.com/rsalabs/challenges/factoring/rsa155.html, 1999.

XILINX 1998, XILINX Development Systems, “Synthesis and Simulation Design Guide – Designing FPGAs with HDL”, 1998.