FUNDAÇÃO DE ENSINO “EURÍPIDES SOARES DA ROCHA”
CENTRO UNIVERSITÁRIO “EURÍPIDES DE MARÍLIA” - UNIVEM
PROGRAMA DE MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

ALEXANDRE PONCE DE OLIVEIRA

PERS - UM PROCESSADOR ESPECÍFICO PARA REDES DE SENSORES COM PRIMITIVAS DE SEGURANÇA

Marília
2006

ALEXANDRE PONCE DE OLIVEIRA

PERS - UM PROCESSADOR ESPECÍFICO PARA REDES DE SENSORES COM PRIMITIVAS DE SEGURANÇA

Dissertação apresentada ao Programa de Mestrado do Centro Universitário Eurípides de Marília, mantido pela Fundação de Ensino Eurípides Soares da Rocha, para obtenção do Título de Mestre em Ciência da Computação. (Área de Concentração: Arquiteturas de Sistemas Computacionais).

Orientador:
Prof. Dr. Edward David Moreno Ordonez

Marília
2006

ALEXANDRE PONCE DE OLIVEIRA

PERS - UM PROCESSADOR ESPECÍFICO PARA REDES DE SENSORES COM PRIMITIVAS DE SEGURANÇA

Banca examinadora da dissertação apresentada ao Programa de Mestrado da UNIVEM/F.E.E.S.R., para obtenção do Título de Mestre em Ciência da Computação. Área de Concentração: Arquiteturas de Sistemas Computacionais.

Resultado: _____

ORIENTADOR: Prof. Dr. _____

1º EXAMINADOR: Prof. Dr. _____

2º EXAMINADOR: Prof. Dr. _____

Marília, 02 de agosto de 2006.

À minha família pela paciência, compreensão e motivação.

Aos meus amigos pelos incentivos e auxílio nos momentos difíceis.

Agradecimentos

Ao meu orientador Prof. Dr. Edward David Moreno Ordonez, pelo apoio, paciência e sugestões que foram essenciais no desenvolvimento e término deste trabalho.

Aos pesquisadores do LAS – Laboratório de Arquitetura de Sistemas, em especial ao Fábio Dacêncio Pereira pela amizade e ajuda na conclusão deste trabalho.

E especialmente à minha mãe, Sueli Ponce de Oliveira e ao meu pai, Antônio Venâncio de Oliveira pela ajuda financeira e por acreditar sempre em seus filhos.

OLIVEIRA, Alexandre Ponce de. **PERS - UM PROCESSADOR ESPECÍFICO PARA REDES DE SENSORES COM PRIMITIVAS DE SEGURANÇA**. 2006. 118 folhas. Dissertação de Mestrado em Ciência da Computação - Fundação de Ensino "Eurípides Soares da Rocha" Centro Universitário "Eurípides de Marília" - UNIVEM, Marília, 2006.

RESUMO

Este projeto apresenta os principais conceitos de redes, redes de sensores, segurança e processadores de redes, focalizando nos principais desafios encontrados para prover segurança para as redes de sensores que possuem uma grande limitação computacional de hardware. Assim propõe e projeta-se um processador específico para redes de sensores, bem como demonstrar os resultados obtidos com a inclusão de primitivas de segurança (criptografia) no processador específico, para que se tenha um bom nível de segurança nas informações trafegadas. Este processador foi denominado de PERS, e foi descrito usando VHDL e prototipado em um FPGA. Foram criadas 4 versões desse processador, sendo que a primeira versão tem 20 instruções e a última versão possui 32 instruções, considerando primitivas de segurança para os algoritmos criptográficos DES e AES. Os resultados obtidos com esses algoritmos inseridos no processador prototipado em FPGAs permitiram cifrar dados a uma velocidade de 183,72 e 240,28 Mbits/seg, respectivamente para o DES e AES. A inserção destas primitivas de segurança produziu uma diminuição na velocidade do processador em 25% e 56% para esses algoritmos.

Palavras-chave: Redes de Sensores. Processador específico. Primitivas de Segurança. Criptografia.

OLIVEIRA, Alexandre Ponce de. **PERS - UM PROCESSADOR ESPECÍFICO PARA REDES DE SENSORES COM PRIMITIVAS DE SEGURANÇA**. 2006. 118 folhas. Dissertação de Mestrado em Ciência da Computação - Fundação de Ensino "Eurípides Soares da Rocha" Centro Universitário "Eurípides de Marília" - UNIVEM, Marília, 2006.

ABSTRACT

This project presents the main concepts of networks, sensor networks, security and processor networks, focusing on the main challenges found to supply security for the sensor networks that possess a big limitation computational of hardware. It proposes and projected a specific processor for sensor networks, as well as shows the results obtained with the inclusion of security primitives (cryptography) in the specific processor, to have a good security level in the information, without interfere in his performance. It was development in VHDL and prototyped in a FPGA. There were created 4 versions of that processor, the first version has 20 instructions and the last has 32 instructions, considering security primitives for cryptography algorithms DES and AES. The results obtained with those algorithms inserted in the processor prototyped in FPGAs permitted to encode data in a speed of 183.72 and 240.28 Mbits/sec, respectively for the DES and AES. The insertion of these security primitives produced a reduction in the processor speed in 25% and 56% for those algorithms.

Keywords: Sensor Networks. Processor Networks. Security Primitives. Cryptography.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADC: Analogic to Digital Converter

AES: Advanced Encryption Standard

CISC: Complex Instruction Set Computer

CLB: Configurable Logic Block

CRC: Cyclical Redundancy Check

DARPA: Advanced Research Projects Agency

DES: Data Encryption Standard

DMC: Distributed Measurement and Control

DNS: Domain Name System

DoS: Denial of Service

DSN: Distributed Sensor Networks

ERB: Estação Rádio Base

FPGA: Field-Programmable Gate Array

FSM: Finite State Machine

FTP: File Transfer Protocol

HTTP: Hyper Text Transfer Protocol

IEEE: Institute of Electrical and Electronics Engineers

INSENS: Intrusion-tolerant routing protocol for wireless sensor networks

IP: Internet Protocol

ISA: Instruction Set Architecture

ISO: International Standards Organization

ISS: Integrated Smart Sensor

LAN: Local Area Network

LAS: Laboratório de Arquitetura de Sistemas

LUT: Lookup Table

MAC: Media Access Control

MAN: Metropolitan Area Network

MBPS: Mega *bits* por segundo

MEMS: Micro Electro-Mechanical Systems

MPS: Michigan Parallel Standard

MSS: Michigan Serial Standard

NPSOC: Novo Processador de Rede

OSI: Open Systems Interconnection

PC: Contador de Programa

PERS: Processador Específico para Redes de Sensores

R2NP: Reconfigurable RISC Network Processor

RC5: Rivest Cipher 5

RCNP: Reconfigurable CISC Network Processor

RI: Registrador de Instruções

RISC: Reduced Instruction Set Computer

SMTP: Simple Mail Transfer Protocol

SNEP: Secure Network Encryption Protocol

SOSUS: Sound Surveillance System

SPINS: Security protocols for sensor networks

µTESLA: Timed, Efficient, Streaming, Loss-tolerant Authentication Protocol

TCP: Transmission Control Protocol

UC: Unidade de Controle

UDP: User Datagrama Protocol

ULA: Unidade Lógica e Aritmética

VHDL: VHSIC Hardware Description Language

VHSIC: Very High Speed Integrated Circuit

VLIW: Very Large Instruction Word

WAN: Wide Area Network

WSCAD: Workshop em Sistemas Computacionais de Alto Desempenho

SUMÁRIO

1. Introdução

As Redes de Sensores vêm crescendo cada vez mais, impulsionadas pela diminuição dos componentes eletrônicos, pela evolução dos componentes de hardware e pelas comunicações sem fio. Estão sendo utilizadas para diferentes aplicações como: climáticas, químicas, biológicas, militares, educacionais, médicas, entre outras. Antigamente, o principal objetivo delas era a monitoração remota de ambientes inimigos ou de difícil acesso, tendo sido desenvolvidas e utilizadas em aplicações militares com o objetivo de monitorar o campo de batalha em busca de ameaças (LOUREIRO et al., 2002).

As Redes de Sensores são formadas por vários nodos sensores (pequenos sensores capazes de captar, processar e enviar dados) que, trabalhando em conjunto, podem ser usados nas mais diferentes aplicações (LOUREIRO et al., 2002). Alguns exemplos dessas aplicações são: monitoramento do fluxo de veículos em rodovias, descoberta de focos de incêndio em uma floresta, prover segurança nos mais variados estabelecimentos, entre outras (LOUREIRO et al., 2002).

Em HEIDEMANN et al. (2001), uma rede de sensores é definida como uma classe particular de sistemas distribuídos, onde as comunicações de baixo nível não dependem da localização topológica da rede. Desta forma, possui características particulares como a utilização de recursos restritos de energia, topologia de rede dinâmica e uma grande quantidade de nós. Estas características dificultam a reutilização de alguns algoritmos desenvolvidos para outros tipos de sistemas distribuídos. As soluções para estes problemas, como a sincronização da rede, a eleição de um líder e a aquisição de informações que representam o estado da rede devem considerar também características como a precisão, eficiência e o custo das operações.

As Redes de Sensores são classificadas de acordo com diferentes funções de comunicação, modelos de envio de dados, dinamismo da rede, métricas de desempenho e arquitetura. Esta taxonomia pode ajudar na definição apropriada de infra-estruturas de comunicação para diferentes sub-espaços de aplicações de redes de sensores, permitindo que projetistas de redes escolham o protocolo de arquitetura que melhor se adapte com os objetivos de sua aplicação. Além disso, esta taxonomia habilita novos modelos de redes de sensores para serem definidos para futuras pesquisas nesta área (TILAK et al., 2002).

As redes de sensores possuem como elementos principais: o sensor, o observador e o fenômeno, que são detalhados a seguir.

(i) o sensor é o dispositivo que implementa a monitoração física de um fenômeno ambiental e gera relatórios de medidas (TILAK et al., 2002). Um sensor produz uma resposta mensurável a mudanças em condições físicas, tais como temperatura, campo magnético e luz (MEGERIAN et al., 2002). Os dispositivos de detecção, geralmente, têm características físicas e teóricas diferentes. Muitos modelos de complexidade variada podem ser construídos baseados na necessidade da aplicação e características dos dispositivos. Na maioria dos modelos de dispositivos sensores, a habilidade de detecção diminui com o aumento da distância do sensor ao fenômeno e melhora com o aumento do tempo que o sensor fica exposto para coletar informações (MEGERIAN et al., 2002). Um sensor, tipicamente, consiste de cinco componentes: detector de hardware, memória, bateria, processador embutido e transmissor-receptor (TILAK et al., 2002).

(ii) O observador é o usuário final interessado em obter as informações disseminadas pela rede de sensores em relação a um fenômeno. Além disso, podem existir, simultaneamente, múltiplos observadores numa rede de sensores (TILAK et al., 2002).

(iii) O fenômeno é a entidade de interesse do observador, que está sendo monitorada e cuja informação potencialmente será analisada ou filtrada pela rede de sensores. Além disso,

múltiplos fenômenos podem ser observados concorrentemente numa rede (TILAK et al., 2002).

Numa rede de sensores típica, os sensores individuais apresentam amostras de valores locais e disseminam informação, quando necessário, para outros sensores e eventualmente para o observador.

Um dos pontos mais críticos das Redes de Sensores são suas limitações de recursos, como por exemplo, pouca capacidade computacional, pouca memória e o principal, sua limitação de energia (oriunda de uma bateria). Esta limitação tem influência nessas redes, pois os sensores são utilizados em áreas de difícil acesso, impossibilitando uma possível manutenção; Devido a isso, deve-se projetar essas redes visando baixo consumo de energia (LOUREIRO et al., 2002).

Em STANKOVIC (2002) são apresentados alguns fatores que criam diferentes desafios para a tecnologia de sensores:

- os nós encontram-se embutidos numa área geográfica e interagem com um ambiente físico;
- são menores e menos confiáveis que roteadores de redes tradicionais;
- geram dados detectados, ao contrário de roteadores de rede;
- podem ser móveis.

Esses fatores representam um novo paradigma para as operações de rede, ferramentas de software e protocolos para habilitar a programação e o uso efetivo de tais sistemas de computação embutida em redes (STANKOVIC, 2002).

Além disso, os serviços de segurança, como a inclusão de um algoritmo de criptografia nos dados trafegados na rede, são muito importantes para manter a privacidade dos dados que trafegam, pois o acesso aos dados será possível apenas com o conhecimento da chave para decifrar a informação recebida.

Para obter-se segurança em uma rede de sensores, devem-se cumprir determinados requisitos. Esses requisitos são importantes, pois pesam na escolha do melhor algoritmo de criptografia a ser utilizado (PERRIG et al., 2001).

A rede deve estar sempre disponível para usuários autorizados, portanto deve estar livre de ataques de negação de serviço (DoS - *Denial of Service*). Este é um tipo de ataque que sobrecarrega os recursos da rede. Deve-se ter cuidado também para que determinados serviços que consumam muita energia não sejam utilizados para que a rede não tenha um tempo de vida reduzido (STANKOVIC, 2002).

Outro requisito de segurança é a confidencialidade dos dados onde um intruso que roube as informações trocadas pelos nós não tenha condição de compreendê-las. Para isso pode-se utilizar criptografia nos dados, onde as chaves criptográficas devem ficar em poder dos nós. Quanto mais chaves cada nó tiver condição de utilizar mais confidencial será a informação, porém a inclusão de criptografia irá causar problemas para a rede, devido as suas limitações de recursos computacionais (PERRIG et al., 2001).

A autenticação garante que todas as informações recebidas por um determinado nó são realmente de uma fonte segura, evitando assim que sensores maliciosos façam injeção de dados. A autenticidade se faz necessária, principalmente, para proteger informações relevantes ao funcionamento correto da rede ou para evitar que invasores se passem por usuários autorizados e façam alterações nos dados da mesma. Segundo PERRIG et al. (2001), para verificar se o dado foi realmente originado pelo nó indicado podem ser utilizados protocolos que fazem desafios aos nós transmissores. Estes enviam mensagens em texto claro para que os nós que estão sendo autenticados criptografem com sua chave. A autenticidade é confirmada através da decriptografia dos dados enviados ao mecanismo autenticador, que posteriormente ao recebimento do desafio verifica se a chave utilizada é realmente de quem diz ser e se a mensagem é a mesma que foi originada (PERRIG et al., 2001).

A atualização dos dados garante que determinadas informações não sejam copiadas e posteriormente sejam injetadas novamente na rede. Os dados copiados seriam autênticos, porém não seriam mais válidos. A atualização dos dados pode ser alcançada através de renovações de chaves criptográficas, onde as chaves só são válidas por intervalos de tempo e os nós que se comunicam e pertençam à rede têm conhecimento dessas mudanças (PERRIG et al., 2001).

Já a integridade dos dados garante que os dados não foram alterados em trânsito por um adversário. Um determinado dado pode ser manipulado sem que o atacante nem ao menos saiba do que se tratava, por estar criptografado. Em PERRIG et al. (2001), a integridade de dados é alcançada pela autenticação dos dados.

Os sensores devem ser resistentes a manipulação, pois um usuário malicioso, ao ter acesso a um nó, não pode obter informações sigilosas como dados, código e até mesmo a chave criptográfica ou alguma pista que lhe leve a tal (HU e EVANS, 2003).

A cooperação entre os nós sensores é de vital importância para o funcionamento da rede, ou seja, nenhum nó pode entrar na rede e se negar a encaminhar pacotes de dados ou de controle (HU e EVANS, 2003).

1.1 Justificativa

O acesso contínuo às informações é essencial para o usuário nas comunicações móveis sem fio. Assim, estes tipos de redes são apropriados para situações onde não se pode ter uma instalação com fios e que requer acesso imediato à informação (MALLADI e AGRAWAL, 2002).

Com isso, utiliza-se as redes de sensores para aplicações críticas voltadas para o monitoramento, como por exemplo, monitoramento ambiental, rastreamento de objetos, monitoramento da saúde, sistemas militares, controle de tráfego aéreo, usinas, etc.

Os processadores de rede possuem uma grande importância na rede, pois eles podem oferecer processamento em tempo real, segurança, *store and forward*, manipulação de pacotes IP (TANENBAUM, 1997). Isto é possível, pois eles são aplicados no meio da rede (PRADO, 2004), isto é, em nós (nodos) que compõem as redes de sensores.

A idéia principal desta dissertação de mestrado foi desenvolver uma solução via *hardware* de um processador específico para redes de sensores pois, com isso, pode-se aumentar o desempenho, escalabilidade e segurança da aplicação específica.

As Redes de Sensores podem sofrer vários tipos de ataques, pois a comunicação sem fio as tornam mais vulneráveis. Para prover segurança nas Redes de Sensores, deve-se saber quais os requisitos e objetivos utilizados na aplicação (HU e EVANS, 2003).

Os principais ataques às redes de sensores são o *Spoofing*, Encaminhamento seletivo, Ataque sorvedouro, Ataque *Sybil*, *Wormholes*, Ataque de inundação HELLO e *Spoofing* de reconhecimento positivo (KARLOF e WAGNER, 2003).

Para evitar todos os tipos de ataques acima, as Redes de Sensores necessitam de um algoritmo de criptografia para evitar que os dados enviados e recebidos sofram algum tipo de interferência, chegando ao seu destino final com integridade e eficiência.

Os melhores algoritmos de segurança são aqueles que conseguem proteger da melhor forma a aplicação e também consumir o mínimo dos recursos que a redes de sensores possuem.

Os problemas e soluções citados acima levaram a um estudo aprofundado e minucioso das Redes de Sensores, Processadores de Redes e Segurança, de modo que todos esses fatores motivaram a elaboração deste projeto, pois os mecanismos de segurança em

redes de sensores ainda têm muito que evoluir e ainda se mostram em aberto para pesquisa e desenvolvimento técnico.

1.2 Objetivos

Este trabalho de pesquisa em nível de mestrado teve como objetivo apresentar os problemas que existem hoje nas redes de sensores, as principais ameaças que elas são submetidas com ataques maliciosos e propor uma solução que impeça um bom número de ataques sem afetar o desempenho da rede, pois elas possuem muitas limitações como, por exemplo, processamento, capacidade e longevidade.

Portanto, o projeto consistiu em propor e projetar o PERS, um processador específico para redes de sensores incluindo primitivas de segurança (criptografia), e verificar o impacto no desempenho do processador quando essas primitivas são inseridas. O processador foi projetado usando VHDL e FPGAs. Com isso, pode-se introduzir um determinado nível de segurança para proteger as informações que são transmitidas por sensores em uma rede de sensores, conforme figura 1.1.

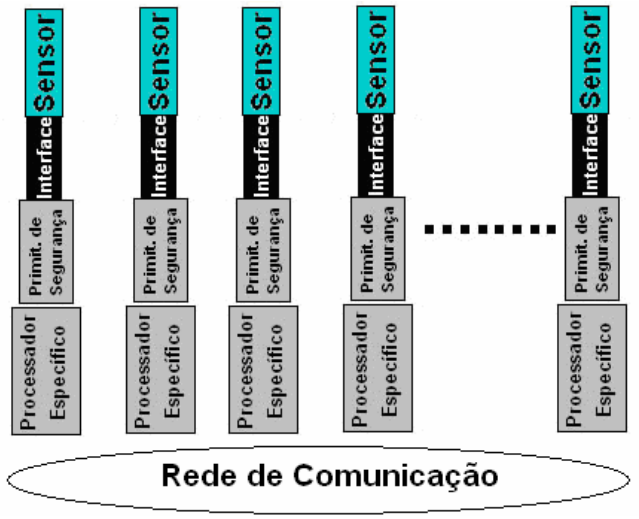


Figura 1.1. Criptografia em uma rede de sensores.

Na figura 1.1 pode-se observar a inclusão de criptografia na comunicação entre todos os nós sensores da rede, inseridas em um processador específico para redes de sensores. Com isso, visa-se manter sigilosas as informações trocadas entre os nós sensores e também dos nós sensores para a estação base.

1.3 Organização do trabalho

A dissertação é composta de 06 capítulos, o capítulo 1 apresenta alguns conceitos introdutórios das redes de sensores, o objetivo e organização deste trabalho.

O capítulo 2 discute os principais conceitos sobre Redes e Redes de Sensores.

O capítulo 3 discute os principais conceitos sobre Segurança.

O capítulo 4 apresenta a base de um processador NPSOC que é um novo processador de rede implementado por (PRADO, 2004) no Laboratório de Arquitetura de Sistemas (LAS) da Fundação Eurípides Soares da Rocha.

O capítulo 5 apresenta a implementação do novo processador PERS, específico para Redes de Sensores e a implementação e testes das primitivas de segurança, usando os algoritmos criptográficos DES e AES.

Finalmente, o capítulo 6 apresenta as conclusões deste trabalho, assim como sugestões de assuntos para trabalhos futuros.

2. Redes de Computadores e Redes de Sensores

Este capítulo apresenta os principais aspectos relacionados às Redes de Computadores e às Redes de Sensores, pois Redes é o assunto mais abordado neste trabalho.

2.1 Conceito de Redes de Computadores

Redes de Computadores são definidas como um conjunto de computadores interligados entre si e que podem trocar informações e compartilhar recursos, através de um meio de comunicação.

As redes são montadas com o principal objetivo de compartilhar recursos e isso pode representar uma redução considerável no custo de uma empresa (TANENBAUM, 1997).

Segundo ROSSI (2004), com as redes pode-se ter maior confiabilidade nos sistemas de informação, pois existem várias alternativas de fornecimento, além de oferecer uma escalabilidade, na qual se pode aumentar gradativamente o desempenho e os recursos de um sistema à medida que cresça e alcance um volume maior de dados.

2.2 Amplitude das Redes de Computadores

As redes de computadores podem ser classificadas de acordo com sua dispersão geográfica em: redes locais, redes metropolitanas, redes geograficamente distribuídas e inter-redes. Na Tabela 2.1 mostra-se uma classificação detalhada.

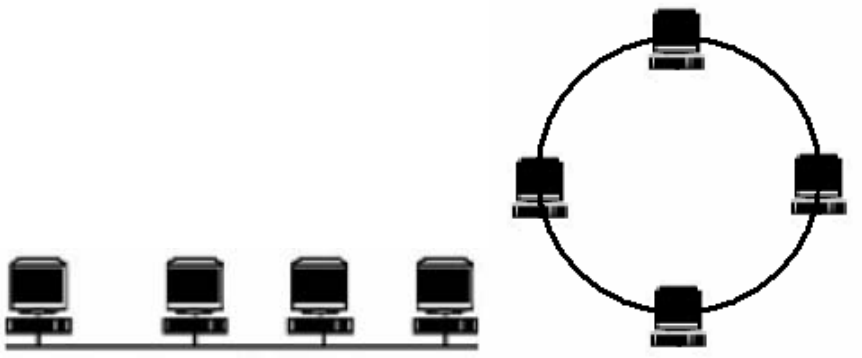
Tabela 2.1 Classificação das redes por amplitude

Distância do interprocessador	Processadores no(a) mesmo(a)	Exemplo
0,1 m	Placa de circuitos	Máquina de fluxo de dados
1 m	Sistema	Multicomputador
10 m 100 m 1 Km	Sala Prédio Campus	Rede Local (LAN)
10 Km	Cidade	Rede Metropolitana (MAN)
100 Km 1000 Km	País Continente	Rede Geograficamente Distribuída (WAN)
10000 Km	Planeta	Inter-Rede

Fonte: (TANENBAUM, 1997)

2.2.1 Rede Local (LAN)

Essas redes são utilizadas para interligar computadores numa mesma sala, empresa ou universidade, com a finalidade de compartilhar recursos entre os servidores e estações de trabalho. A velocidade de transmissão dessas redes varia de 10, 100 ou mais Mbps[1]. A tecnologia de rede local mais utilizada obedece ao padrão IEEE 802.3, conhecido como Ethernet (IEEE, 1985). Na figura 2.1 são apresentadas duas topologias básicas de Redes Locais: Barramento e Anel.



a) **b)**

Figura 2.1. Topologia de redes locais, a) Barramento; b) Anel

[1] Mbps: unidade de medida de largura de banda usada para expressar um milhão de *bits* por segundo.

a) Topologia em Barramento

Na topologia em barramento as estações estão todas interligadas através de um mesmo condutor, o barramento, do qual são derivadas as ligações das estações. Neste modelo os dados são transmitidos através de uma linha, ou barra, bidirecionalmente, ficando disponíveis às estações, as quais estão todas ligadas nesta barra. Esse modo de operação é chamado de difusão (*broadcasting*). Esta topologia é de fácil implementação, porém a qualquer problema de conexão toda a rede para, além dos sinais serem atenuados com a distância (IEEE, 1985).

b) Topologia em Anel

Na topologia em anel as máquinas são interligadas em um anel lógico através de um concentrador, dispositivo similar ao *hub*, porém é utilizado em redes *Token Ring*. Uma ficha (*token*) circula pela rede, caso esteja vazia, um computador pode “encher” a ficha e enviar um quadro de dados para outra máquina na rede. Esta ficha circula a rede inteira até encontrar a máquina destino, sendo que o identificador consta no quadro de dados. O computador de destino “esvazia” a ficha e envia um sinal para o computador transmissor (IEEE, 1985).

2.2.2 Rede Metropolitana (MAN)

As MANs são usadas para interligação de computadores dispersos numa área geográfica mais ampla, onde não é possível ser interligada usando tecnologia de LAN. Uma MAN pode abranger uma região de uma cidade, chegando às vezes, a interligar até computadores de cidades vizinhas próximas (IEEE, 1985).

2.2.3 Rede Geograficamente Distribuída (WAN)

São redes que usam linhas de transmissão fornecidas por empresas de telecomunicações para interligar diversas LANs. Elas abrangem computadores localizados em diferentes cidades, estados, países ou continentes (IEEE, 1985).

2.2.4 Inter-Redes

Estabelece conexões entre diferentes redes, ou seja, possibilitam a comunicação através da utilização de equipamentos chamados de *gateways*, que são conhecidos também como conversores de protocolo. Define-se inter-rede como um conjunto de redes interconectadas, como por exemplo a Internet, que é a rede mundial dos computadores (IEEE, 1985).

2.3 Camadas, Protocolo, Interfaces e Serviços

Com o objetivo de reduzir o grau de complexidade, a estrutura de *software* de uma rede divide-se em diferentes camadas. A camada n de uma máquina se comunica com a camada n da outra, o protocolo de camada que estabelecerá o conjunto de regras da comunicação entre as partes (TANENBAUM, 1997). Na figura 2.2 é mostrado o conceito de forma esquemática.

Os dados não são transferidos diretamente entre a camada n de uma máquina para a camada n da outra, ao invés disso a informação passa por todos os níveis até atingir a camada

requisitada. O meio físico se inclui neste caminho, onde a comunicação propriamente dita é feita.

Entre as camadas existe uma interface que define as operações e serviços que cada uma das camadas tem a oferecer para a outra. Todo este conjunto de camadas, protocolos e interfaces, denomina-se arquitetura de rede.

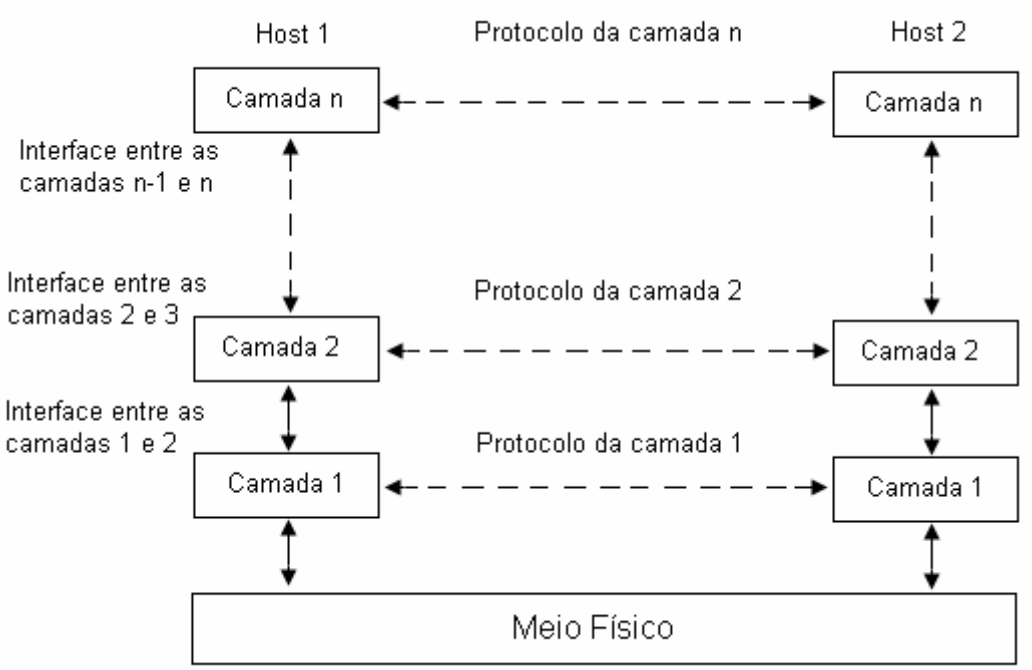


Figura 2.2. Camadas, protocolos e interfaces (TANENBAUM, 1997)

Existem serviços orientados à conexão e serviços sem conexão ou de datagrama. No serviço orientado à conexão, o usuário estabelece uma conexão, utiliza a conexão e finalmente a libera. O emissor envia *bits* e o receptor os recebe na ordem em que foram enviados. O serviço sem conexão ou de datagrama, é baseado no sistema postal. Cada mensagem tem o endereço de destino completo que são roteados através do sistema independentemente de todos os outros. Este serviço é usado para tráfego de voz digital ou transmissão de um vídeo, quando não há problema quando aparecem *bits* de ruído ou *pixels* errados durante a transmissão (TANENBAUM, 1997).

Segundo ROSSI (2004), um serviço é especificado por um conjunto de primitivas disponíveis para que uma entidade possa acessá-lo. Por exemplo, a primitiva *request* que faz referência a uma entidade querendo que o serviço execute alguma ação, ou a primitiva *response*, fazendo referência a uma entidade querendo responder a algum evento.

2.4 Transmissão de Dados

Segundo TORRES (2001), a transferência de dados na rede ocorre através das seguintes formas:

a) Simplex: transferência em apenas uma direção.

b) *Half duplex*: transferência em duas direções, porém não simultaneamente.

c) *Full duplex*: transferência em duas direções simultaneamente.

2.4.1 Transmissão Paralela

Na transmissão paralela o transmissor envia todos os *bits* para o receptor ao mesmo tempo, por exemplo uma palavra de oito *bits*. Neste tipo de transmissão existe um fio para cada *bit* transmitido, podendo causar interferência no fio adjacente, para evitar essa interferência recomenda-se o comprimento máximo do cabo entre 10 e 15 metros (TORRES, 2001).

2.4.2 Transmissão Serial

Nesta transmissão existe apenas um fio onde o transmissor envia *bit* a *bit* para o receptor, assim esta transmissão é bem mais lenta que a paralela. Porém na transmissão serial

não existe o problema de interferência, consequentemente a distância dos cabos são bem superiores do que a transmissão paralela (TORRES, 2001).

2.5 Modelo de Referência

Existem dois modelos de referência mundialmente conhecidos, o modelo ISO/OSI e o modelo TCP/IP. Embora estes modelos sejam chamados de arquiteturas de rede, não são, pois fazem apenas referência a camadas e não a protocolos e interfaces (TORRES, 2001).

2.5.1 Modelo de Referência ISO/OSI

Este modelo surgiu a partir de uma proposta da Organização de Padrões Internacionais (ISO – *International Standards Organization*) para que os fabricantes pudessem criar protocolos a partir de um modelo. A sigla OSI (*Open Systems Interconnection*) significa interconexão de sistemas abertos. A maioria dos protocolos existentes não segue a risca o modelo, porém este modelo constitui uma ferramenta didática essencial, pois através dele é possível entender o funcionamento do protocolo ideal. A figura 2.3 mostra as sete camadas do modelo OSI (TORRES, 2001). A seguir é feita uma descrição de cada camada a partir de informações obtidas em (TORRES, 2001).

Camada 7	Aplicação
Camada 6	Apresentação
Camada 5	Seção
Camada 4	Transporte
Camada 3	Rede
Camada 2	Enlace
Camada 1	Física

Figura 2.3. Modelo de referência ISO/OSI

a) Camada de Aplicação

A camada de aplicação faz a interface entre o protocolo de comunicação e o aplicativo que solicitou alguma tarefa através do usuário.

b) Camada de Apresentação

A camada de apresentação converte os dados recebidos pela camada de aplicação em um formato a ser usado na transmissão, ou seja, um formato entendido pelo protocolo. Esta camada se preocupa com a sintaxe e a semântica dos dados transmitidos.

c) Camada de Sessão

A camada de Sessão permite que dois computadores diferentes estabeleçam uma sessão de comunicação. Nesta camada os dados são marcados de forma que se houver uma falha na rede, os computadores reiniciarão a comunicação de onde parou, a partir da última marcação recebida pela máquina receptora.

d) Camada de Transporte

A camada de transporte é responsável por receber os dados vindos da camada de sessão, dividí-los em pacotes e mandá-los para a camada de rede. No receptor, esta camada é responsável por pegar os pacotes recebidos da camada de Rede e remontar o dado original para enviá-lo à camada de Sessão.

e) Camada de Rede

A camada de rede é responsável pelo endereçamento dos pacotes, de forma que os pacotes consigam chegar corretamente ao destino. Essa camada também determina a rota que

os pacotes irão seguir para atingir o destino no caso de existir mais de uma rota para o envio dos dados. Esta camada resolve os problemas existentes no trajeto do pacote de uma ponta na outra ponta, como o tamanho do pacote enviado, ou seja, possibilita a comunicação de redes heterogêneas.

f) Camada de Enlace de Dados

Transforma os pacotes de dados recebidos da camada de rede em quadros a serem trafegados pela rede, acrescentando informações importantes como o endereço da placa de rede de origem e destino, os dados e o CRC[2] (*Cyclical Redundancy Check*).

g) Camada Física

A camada física pega os quadros enviados pela camada de Enlace de Dados e os converte em sinais compatíveis com o meio onde os dados deverão ser transmitidos. O papel desta camada é desempenhado pela placa de rede.

2.5.2 Modelo de Referência TCP/IP

No começo da década de 1970 o crescimento da ARPANET forçou a criação de um novo modelo de referência. O modelo mostrado na figura 2.4 ficou conhecido como modelo de referência TCP/IP (*Transmission Control Protocol/Internet Protocol*) (TANENBAUM, 1997).

[2] CRC: operação que consiste em somar todos os *bytes* de um pacote e enviar o resultado dentro do próprio pacote. A placa de rede do receptor verificará este resultado.

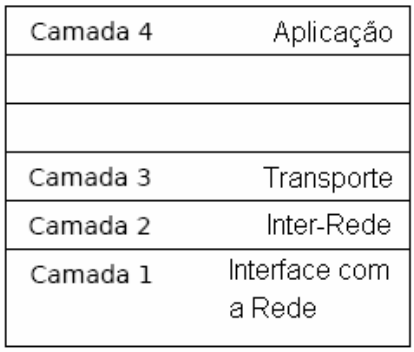


Figura 2.4. Modelo de referência TCP/IP

A seguir uma breve descrição da funcionalidade de cada camada segundo informações obtidas em (TORRES, 2001).

a) Camada de Aplicação

A camada de aplicação corresponde às camadas 5, 6 e 7 do modelo OSI e faz a comunicação entre os aplicativos e o protocolo de transporte. Entre os principais protocolos que operam nesta camada de alto nível destacam-se o Telnet, protocolo de terminal virtual para efetuar *login* remoto em uma máquina de uma rede; HTTP (*Hyper Text Transfer Protocol*), protocolo de transferência padrão da *Web*; SMTP (*Simple Mail Transfer Protocol*), protocolo de correio eletrônico; FTP (*File Transfer Protocol*), protocolo de transferência de arquivos entre *hosts*[3] conectados à rede; DNS (*Domain Name System*): protocolo utilizado para identificação de máquinas através de nomes e não através de endereços IP.

A camada de aplicação comunica-se com a camada de transporte através de uma porta. As portas são numeradas e as aplicações padrão usam sempre uma mesma porta. Por exemplo, o protocolo SMTP utiliza sempre a porta 25, o HTTP a porta 80 e o Telnet a porta 23.

[3] *Hosts*: termo utilizado para fazer referência a um sistema de computador conectado a uma rede. Similar ao termo nó de rede.

b) Camada de Transporte

Esta camada equivale à camada de transporte do modelo OSI. É responsável por pegar os dados enviados pela camada de aplicação e transformá-los em pacotes, a serem repassados para a camada de rede.

Nesta camada operam dois protocolos, o TCP é um protocolo orientado à conexão confiável que permite a entrega sem erros de um fluxo de *bytes* originado de uma máquina determinada, em qualquer nó da inter-rede.

O UDP (*User Datagrama Protocol*) é um protocolo sem conexão, não confiável, para aplicações que não necessitam de controle de fluxo, manutenção da seqüência das mensagens enviadas.

c) Camada de Inter-Redes

Esta camada permite que os *hosts* injetem pacotes em qualquer rede e garante que eles sejam transmitidos independentemente do destino. A ordem de chegada dos pacotes pode ser diferente da ordem enviada, ficando as camadas superiores responsáveis pela organização.

Essa camada define um formato de pacote oficial e um protocolo chamado IP (*Internet Protocol*). O roteamento de pacotes é muito importante nessa camada.

d) Camada de Interface com a Rede

Corresponde às camadas 1 e 2 do modelo OSI, é responsável por enviar os pacotes recebidos pela camada de Inter-rede em forma de quadro através da rede.

2.6 Sistema de Instrumentação

O fluxo de informação relacionado com os sistemas de medição e controle não envolve apenas a comunicação entre objetos e sistemas de medida, mas também a interface homem-máquina (ROSSI, 2004).

Nesse fluxo de informação encontram-se envolvidos três mundos diferentes, como mostra na figura 2.5 (YAMASAKI, 1996).

a) Mundo Físico: representa a magnitude medida e a variável controlada. As leis naturais regem este mundo, onde a causalidade encontra-se estritamente estabelecida.

b) Mundo Lógico: representa o sistema de processamento de informação para medição e controle. As regras lógicas regem este mundo, onde a informação é descrita através de sinais e códigos.

c) Mundo Intelectual do ser Humano: representa o contexto interno do cérebro humano. Neste mundo a informação é traduzida em conhecimentos e conceitos.

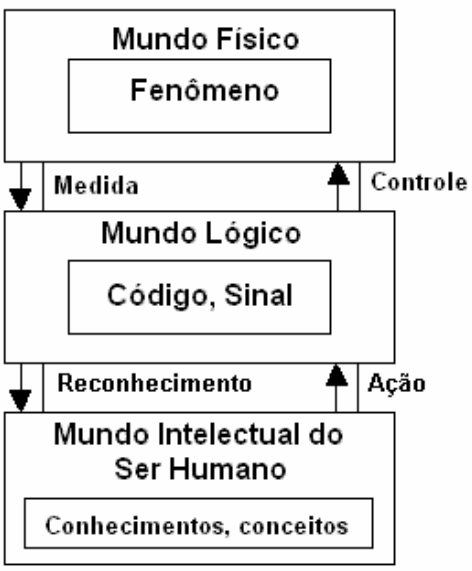


Figura 2.5. Sistema de instrumentação (YAMASAKI, 1996)

A informação do mundo físico é extraída através de dispositivos sensores e técnicas de instrumentação e transferida para o mundo lógico.

A informação é processada, então, de acordo com os objetivos do sistema de medição e controle é reconhecida, depois, pelo homem através da interface homem-máquina, encarregada de apresentar os resultados finais.

Deste modo, a informação é transferida para o mundo intelectual do ser humano para ser transformada em conhecimento e conceitos. As intenções são logo transformadas em comandos e transmitidas ao sistema através da interface homem-máquina.

Em função do comando, o sistema de controle agirá sobre uma variável física através de um dispositivo atuador. Em virtude destes conceitos, define-se “medição e controle” como o processo que envolve medição, reconhecimento, ação e controle em três diferentes mundos ou contextos, realizando-se no mundo físico, a ação desejada pelo ser humano (ROSSI, 2004).

2.6.1 Dispositivo Transdutor

O termo transdutor deriva-se do latim *transducere,* que siginifica “conduzir através de”. Um transdutor é um dispositivo capaz de converter energia de um domínio para outro, na mesma ou diferente forma. Os sinais, assim como a energia, podem ser conduzidos através do elemento transdutor (SZE, 1994).

Os termos sensor e transdutor são por vezes utilizados como sinônimos. A diferença entre ambos os termos é muito sutil (PALLÁS-ARENY e WEBSTER, 1991), (GARDNER, 1994). Um sensor executa uma ação de transdução e pode ser definido como sendo um dispositivo capaz de converter uma grandeza física ou química em um sinal elétrico. Outro caso semelhante é o do atuador. Um atuador é qualquer dispositivo capaz de converter um sinal elétrico em uma grandeza física ou química e, portanto, também executa uma ação de transdução. Assim sendo, pode-se dizer que ambos dispositivos são transdutores (ROSSI, 2004).

Utilizar-se-á a definição proposta pelo IEEE, definindo sensor como sendo um transdutor que converte um parâmetro físico, biológico ou químico em um sinal elétrico e, atuador, como sendo um transdutor que aceita um sinal elétrico para convertê-lo em uma ação física (IEEE, 1999), (IEEE, 1997).

O sensor é utilizado como elemento primário para detectar as condições de um ambiente ou de um processo, por exemplo, mudanças de temperatura. O atuador, por exemplo, uma válvula ou um relé, é utilizado para executar uma ação específica com base na informação fornecida, por exemplo, por um sensor (ROSSI, 2004).

Um transdutor pode ser conectado a uma rede, a fim de que as informações que fornece possam ser compartilhadas por outros dispositivos. Tipicamente, o sinal proveniente de um sensor geralmente é fraco e pode ser facilmente afetado por interferências. Com o objetivo de condicionar o sinal, existem os circuitos de condicionamento, que executam funções como ampliação, filtragem, casamento de impedância, modulação e demodulação (PALLÁS-ARENY e WEBSTER, 1991). Na seqüência, o sinal precisa ser convertido do domínio analógico para o domínio digital. Para tal fim emprega-se um conversor analógico/digital (ADC – *Analogic to Digital Converter*). O sinal em formato digital, então, pode ser enviado para uma unidade microprocessadora ou para um microcontrolador que é o dispositivo encarregado de processar os dados. Finalmente, a informação pode ser enviada para a rede. Na figura 2.6 mostra-se a seqüência de blocos funcionais entre o transdutor e a rede (ROSSI, 2004).

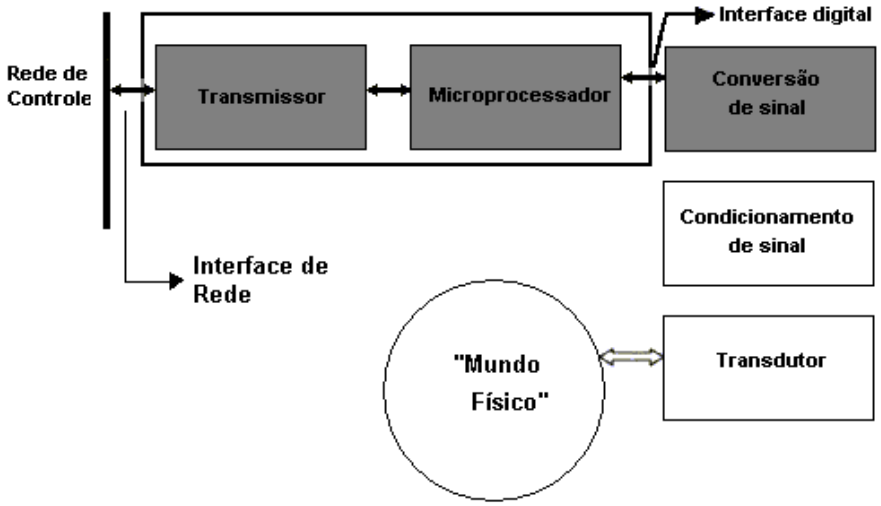


Figura 2.6. Dispositivo transdutor conectado a uma rede (PALLÁS-ARENY e WEBSTER, 1991)

Um ponto relevante no sistema da figura 2.6 é o intercâmbio de informação através das interfaces envolvidas. As interfaces devem ser claras e bem definidas, deste modo simplifica-se a substituição de algum componente do sistema sem mudar a natureza das mesmas (ROSSI, 2004).

2.6.2 Sensor Inteligente

Na literatura atual, existem diferentes pontos de vista sobre a definição de sensor inteligente. Em certos casos, o sensor inteligente é considerado como sendo um dispositivo que integra pelo menos um elemento sensor e um circuito de processamento de sinal. Entretanto, essa definição é inconveniente porque uma ampla gama de sensores cai nessa categoria (ROSSI, 2004).

O termo inteligente é mais adequado para denotar a integração do dispositivo anterior com um processador, que possibilita a introdução de inteligência. A princípio, pode-se estabelecer a seguinte classificação (GARDNER, 1994):

(a) Sistema Sensor: sistema não integrado composto por um sensor, circuitos de pré-processamento e um processador.

(b) Sensor integrado em um sistema sensor: aquele que possui o sensor e circuitos de pré-processamento integrados e uma unidade processadora não integrada.

(c) Sensor inteligente: sistema integrado pelo sensor, os circuitos de pré-processamento e a unidade processadora.

Na figura 2.7 ilustra-se esquematicamente a concepção básica de um sensor inteligente (LEE e SCHNEEMAN, 1999).

Figura 2.7. Concepção básica de um sensor inteligente (GARDNER, 1994)

Em termos simples, um sistema de Medição e Controle Distribuído (DMC – *Distributed Measurement and Control*) consiste em um conjunto de transdutores conectados através de uma rede (LEE e SCHNEEMAN, 1999). A rede mencionada pode ser composta por diferentes níveis até atingir, por exemplo, a Internet. O objetivo de um sistema DMC é controlar um determinado processo, sendo que os transdutores são os dispositivos primários de detecção e atuação. Na figura 2.8 é ilustrado o conceito fundamental de sistema DMC (YAMASAKI, 1996).

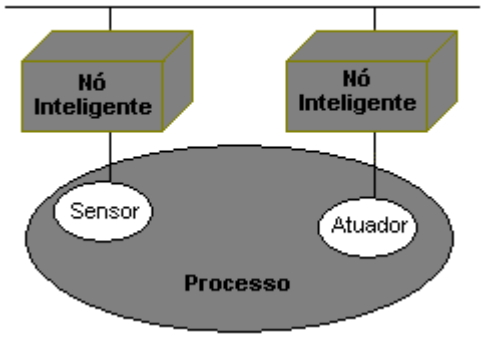


Figura 2.8. Sistema DMC elementar (YAMASAKI, 1996)

Segundo YAMASAKI (1996), em um sistema DMC os transdutores são conectados à rede através da intermediação de nós inteligentes de rede. Nó inteligente é um módulo baseado em algum tipo de processador que:

(a) tem a capacidade de se identificar no sistema;

(b) possibilita o acesso do transdutor à rede;

(c) é responsável pelo processamento de dados do transdutor e da interface de rede associada.

Desta maneira, através da rede de controle, os dados associados aos transdutores podem ser disponibilizados na Internet.

Um nó inteligente de rede possui capacidade de processamento local, podendo tomar decisões sobre o transdutor associado, por exemplo, quando o valor de temperatura fornecido por um sensor for maior do que um valor limite estabelecido, o nó de rede pode acionar um ventilador a fim de estabilizar o sistema. Assim, as atividades podem ser distribuídas através da rede e, ao mesmo tempo, pode se dispor de uma ou várias estações de monitoramento em um nível superior da rede (YAMASAKI, 1996).

Os transdutores conectados em rede possibilitam o compartilhamento das informações de um sistema de instrumentação e fazem com que seja possível supervisionar as variáveis envolvidas, de forma remota, de um processo completo (YAMASAKI, 1996).

As tendências atuais na área de instrumentação mostram as arquiteturas DMC envolvendo transdutores inteligentes como sendo as candidatas a construir a próxima geração de sistemas de instrumentação distribuída (YAMASAKI, 1996).

Com base nos conceitos até aqui estudados, a definição de transdutor inteligente reformula-se da seguinte maneira: transdutor inteligente é um dispositivo com capacidade de processamento local, habilitado para tomar decisões baseando-se no sinal de entrada. E que pode enviar ou receber dados em formato digital, facilitando as atividades em sistemas distribuídos (YAMASAKI, 1996).

Uma analogia interessante é o corpo humano composto por inúmeros elementos sensores. A tarefa mais importante de processamento de informação do sistema de sensoriamento é extrair a informação necessária dos receptores de sinais e transmitir a

informação útil para o cérebro. A informação necessária é transferida de maneira distribuída, liberando assim, a carga de trabalho do cérebro (YAMASAKI, 1996).

2.7 Transdutores de Rede

Nos finais da década de 1970, a *Advanced Research Projects Agency* (DARPA), iniciou as pesquisas na área de sensores em rede, através do programa Redes de Sensores Distribuídos (DSN – *Distributed Sensor Networks*). No ano de 1978 a DARPA patrocinou um *workshop* em que foram identificadas tecnologias para serem aplicadas em redes de sensores distribuídos como técnicas de comunicação, técnicas de processamento e algoritmos para aplicações distribuídas.

Com o transcorrer da década de 1980 várias universidades viram-se envolvidas neste tipo de pesquisa. Atualmente, as pesquisas em redes são de importância vital em diversos centros militares.

Realizando-se uma retrospectiva das tecnologias vinculadas às redes de sensores, podem ser citadas como exemplo, as redes de radar utilizadas para o controle de tráfego aéreo e algumas aplicações de uso militar. Durante a “Guerra fria” o *Sound Surveillance System* (SOSUS) foi empregado pelos americanos com a finalidade de detectar e rastrear submarinos soviéticos. Este sistema era composto de sensores acústicos dispostos de maneira estratégica no fundo do oceano.

Atualmente o SOSUS é utilizado por organizações oceanográficas para monitorar eventos no mar como sismos e atividades relacionadas com o mundo animal. Este último é apenas um exemplo de sistema concebido para aplicações militares que mais tarde tornou-se uma ferramenta de pesquisa nas mãos da comunidade científica.

O desenvolvimento de redes de sensores, ou de modo mais geral, redes de transdutores, precisa da colaboração de três diferentes áreas da engenharia: instrumentação,

informática e comunicação. Portanto, pode-se afirmar que a concepção de um projeto envolvendo transdutores em rede é multidisciplinar. Este fato constitui um desafio, embora hoje existem tecnologias não disponíveis há 20 anos. Sensores, processadores, dispositivos e comunicação são hoje menores e muito mais baratos. Assim, as perspectivas para a próxima década indicam a utilização da tecnologia embarcada, dispositivos de tamanho reduzido e de peso desprezível (CHONG e KUMAR, 2003).

Segundo ROSSI (2004), as tecnologias disponíveis atualmente mudaram aqueles primeiros conceitos na área de sensores em rede. O uso de dispositivos sensores microeletromecânicos (MEMS), a tecnologia de rede sem fios (*wireless*), as tecnologias associadas à Internet e a grande quantidade de ferramentas para desenvolvimento de sistemas digitais, são os novos protagonistas na área de instrumentação.

2.7.1 Conexão de Sensores em Rede

Ainda hoje, uma maneira típica de conectar sensores com instrumentos e sistemas baseados em computador é através de configurações ponto-a-ponto ou de forma multiplexada. Estas técnicas de conexão fazem com que os sistemas de instrumentação fiquem volumosos devido à quantidade de cabos que precisam ser utilizados, além de terem custo elevado e tornarem difíceis as tarefas de manutenção. Junto à evolução acentuada das tecnologias aplicadas às redes de computadores, projetistas e fabricantes de transdutores tentam encontrar maneiras adequadas de adaptar essas tecnologias às redes de transdutores para serem aplicadas em instrumentação e controle (WARRIOR, 2001).

Com o transcorrer das décadas de 1980 e 1990 diversos barramentos para sensores e redes de controle foram desenvolvidos para serem utilizados em diferentes aplicações, abrangendo desde o controle de processos industriais até a automação residencial. Essas

tecnologias, geralmente são denominadas de barramentos para sensores (*Sensorbus*), redes de controle e redes orientadas a dispositivos (*Devicebus*) e barramentos de campo (*Fieldbus*) (WARRIOR, 2001).

2.7.2 Barramento para Sensores

O barramento para sensores ou *Sensorbus* é um sistema básico de interconexão de sensores e atuadores em sistemas de controle baseados em algum tipo de processador. Desta maneira, os dados podem ser transferidos de forma direta para um microcontrolador ou para um microcomputador que age como o controlador do sistema. A comunicação se realiza de forma bidirecional através de um barramento digital de dados e controle como mostrado na figura 2.9 (LEE, 2001).

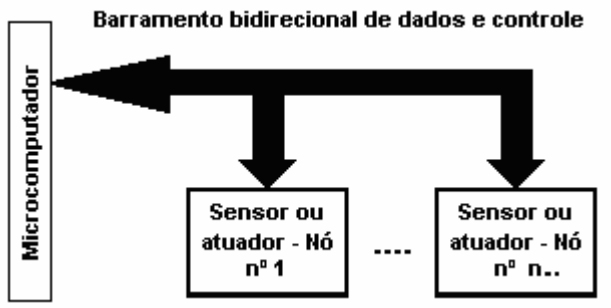


Figura 2.9. Barramento para sensores (LEE, 2001)

Cada nó conectado ao barramento pode conter um ou mais sensores ou atuadores. Note-se que a transmissão de dados realiza-se sempre através do mesmo meio físico, que pode ser cabo de par trançado, cabo coaxial ou fibra óptica. A modalidade de transmissão analógica, por exemplo, a tradicional malha 4-20 mA, precisa de dois fios por variável transmitida, gerando elevados custos de instalação e manutenção (KESTER et al., 2005).

Dentre os primeiros barramentos para sensores conhecidos, destaca-se o Padrão Michigan Paralelo (MPS- *Michigan Parallel Standard*), desenvolvido na Universidade de Michigan (NAJAFI e WISE, 1990).

Este barramento paralelo implementa a interface entre o *host* e os nós através de 16 linhas: 8 linhas bidirecionais de dados, 4 linhas de controle para sincronizar a transferência de dados, um sinal de paridade e 3 linhas de alimentação. A configuração mencionada é apresentada na Figura 2.10 (NAJAFI e WISE, 1990).

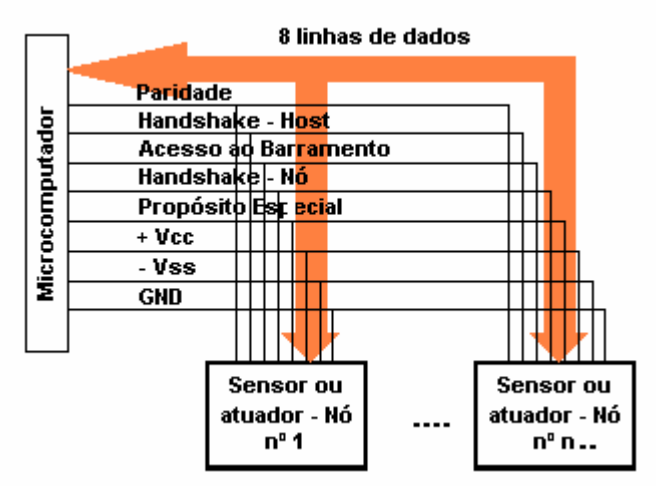


Figura 2.10. Barramento MPS (*Michigan Parallel Standard*) (NAJAFI e WISE, 1990)

No barramento MPS, o *host* inicia uma transferência de mensagem enviando um sinal através da linha *Handshake-Host*[4]. O primeiro *byte* enviado é um endereço que, logo depois, é decodificado por todos os nós conectados ao barramento. O nó correspondente reconhecerá o evento através de um sinal da linha *Handshake-Nó* e, a seguir, colocará um sinal em nível lógico "1" na linha Acesso ao Barramento, para impedir que os outros nós acessem o canal de comunicação. O barramento é liberado quando o sensor muda o estado do sinal de acesso. O sinal de propósito geral pode ser empregado para gerar uma interrupção e requisitar o serviço imediato por parte do *host*, ou para detectar um evento de inserção, caso um novo nó seja conectado ao barramento (ROSSI, 2004).

Uma outra alternativa de interfaceamento entre o *host* e os nós do sistema, denomina-se Padrão Michigan Série (MSS - *Michigan Serial Standard*), mostrado na Figura

2.11. Neste caso, a transferência de dados realiza-se em série, através de uma linha de dados que utiliza um sinal de relógio com a finalidade de sincronizar a transmissão. Esta configuração implementa apenas 5 linhas e possui elevada imunidade ao ruído (ROSSI, 2004).

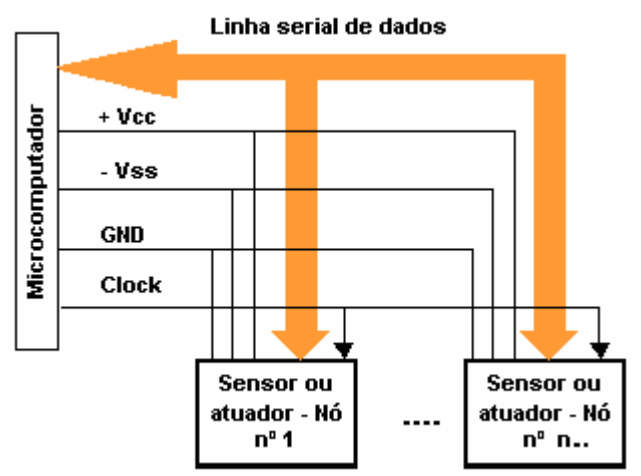


Figura 2.11. Barramento MSS (*Michigan Serial Standard*)

A mensagem emitida pelo *host* contém o endereço correspondente, o tamanho da mensagem, o comando e os dados propriamente ditos, ao passo que a mensagem enviada pelo nó contém o código correspondente, o tamanho da mensagem e os dados (ROSSI, 2004).

Com o objetivo principal de manter os custos em um valor razoável, a Universidade de Tecnologia de Delft na Holanda propôs uma configuração conhecida como *Integrated Smart Sensor* (ISS), apresentada na Figura 2.12 (LEE, 2001), (CORREIA et al., 1997).

[4] *Handshake*: Termo que faz referência ao processo de estabelecimento de um contato inicial entre dois dispositivos de comunicação.

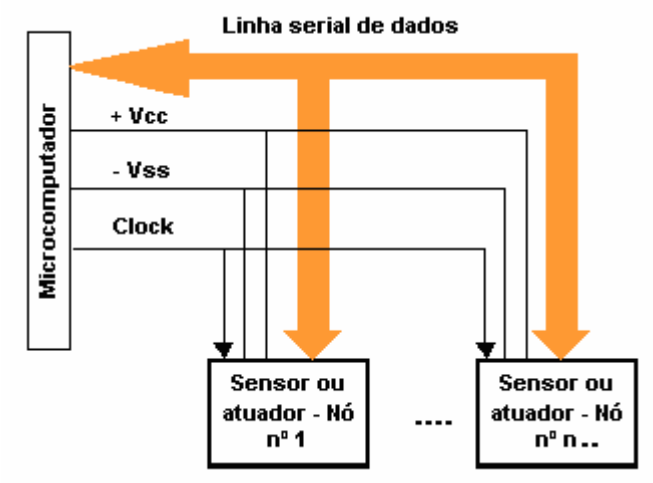


Figura 2.12. Barramento ISS (*Integrated Smart Sensor*) (HUIJSING, 1992)

Nesta configuração, os nós são sensores com processadores integrados que utilizam o barramento ISS para estabelecer a transferência de dados com um controlador central. O barramento ISS utiliza a técnica de comunicação mestre-escravo e é composto por 4 linhas, duas de alimentação, a linha de dados e um sinal de relógio. O mestre controla o relógio, o qual pode ser ajustado até um máximo de 400 kHz. Os dispositivos escravos podem requisitar serviço do mestre quando precisarem de atenção (CORREIA et al., 1997).

Os conceitos apresentados neste capítulo foram essenciais no desenvolvimento do objetivo deste trabalho. Os conceitos de redes baseados no modelo de referência OSI, tipos de transmissão, tipos de redes contribuíram para a definição da topologia utilizada no PERS (Processador Específico para Redes de Sensores). Os conceitos de sensores e transdutores de redes, barramentos para redes de sensores entre outros, foram essenciais o início do desenvolvimento do PERS, pois as Redes de Sensores possuem características diferentes das redes convencionais.

3. Segurança de Dados e Criptografia

Este capítulo apresenta os principais aspectos relacionados à segurança eletrônica dos dados que é importante e discutida. Hoje em dia informações são transmitidas com mais eficiência e velocidade, mas nem sempre de maneira segura (MORENO et al., 2005). Este capítulo apresenta também vários tipos de ataques que as redes de sensores podem sofrer, bem como algoritmos de segurança que podem ser aplicados para eliminar os ataques a essas redes.

3.1 Segurança da informação

O conceito de segurança da informação é muito mais amplo do que a simples proteção dos dados em nível lógico. Para proporcionar uma segurança real é preciso estar atento a vários fatores internos e externos. Em primeiro lugar há a necessidade de se caracterizar o sistema que irá armazenar as informações para poder identificar as ameaças, e neste sentido, fazer a seguinte subdivisão (LOPEZ, 2004):

(i) Sistemas Isolados - São aqueles que não estão conectados a nenhum tipo de rede;

(ii) Sistemas Interconectados - Atualmente, a maioria dos computadores existentes, pertence a uma rede de computador, enviando e recebendo informações do exterior quase que constantemente. Neste tipo de sistema sempre há riscos potenciais.

Quanto às questões de segurança, o objetivo de segurança da informação pode ser classificado em:

(i) Segurança Física - Esta categoria engloba todos os assuntos relacionados com a salvaguarda e o suporte físico da informação. Neste nível estão as medidas contra incêndio, sobrecargas elétricas, políticas de *backup*, entre outros;

(ii) Segurança da Informação - Nesta categoria apresenta-se a preservação da informação frente a pessoas não autorizadas. É neste nível que entra a criptografia, simétrica e assimétrica;

(iii) Segurança do Canal de Comunicação - Raramente os canais de comunicação são seguros. Devido ao fato da maior parte das redes existentes pertencerem a terceiros, se torna muito improvável assegurar totalmente a segurança nela;

(iv) Problemas de Autenticação – Devido aos problemas nos canais de comunicação, se torna necessário assegurar que a informação que se recebe em um computador vem realmente de quem se pensa que a enviou. Neste nível é conveniente a utilização da Assinatura Digital;

(v) Problemas de Suplantação – Nas redes tem-se o problema crônico de que qualquer usuário autorizado possa acessar as informações, mesmo de fora. Tem-se que garantir que usuários não estão sendo suplantados por intrusos. Normalmente se utiliza mecanismos baseados em senha para solucionar este problema;

(vi) "Descarte de Informação" ou Não Repúdio – É necessário haver uma política de não descarte de informação, pois esta, se armazenada, pode revelar o autor de uma determinada ação. Ex: um usuário ter realizado um ato de comércio e posteriormente tentar negar esse mesmo ato.

A tabela 3.1 mostra a expansão dos objetivos principais dos Sistemas Criptográficos.

Tabela 3.1 Objetivos dos Sistemas Criptográficos (MENEZES et al., 1996)

Características	**Descrição**
Privacidade ou confidencialidade	Manter informações secretas disponíveis somente para quem está autorizado
Integridade dos dados	Garantir que a informação não seja alterada por meios desconhecidos ou não autorizados.
Autenticação ou identificação da entidade	Validação da identidade de uma entidade
Autenticação de mensagens	Confirmar a origem da informação
Assinatura	Maneira de ligar a informação à entidade

Tabela 3.1 Objetivos dos Sistemas Criptográficos (MENEZES et al., 1996)

Características	**Descrição**
Autorização	Comunicar, à outra entidade, de permissão oficial para fazer algo ou ser alguém
Validação	Maneira de fornecer tempo limite de autorização para usar ou manipular informações e recursos
Controle de Acesso	Restringir o acesso aos recursos à entidades privilegiadas
Certificação	Confirmação de uma informação por uma entidade confiável
Selo de criação	Gravação da hora de criação ou tempo de existência da informação
Testemunhar	Verificação da criação ou existência da informação por uma entidade diferente da criadora
Recibo	Reconhecimento de que a informação foi recebida
Confirmação	Reconhecimento de que o serviço foi providenciado
Propriedade	Maneira de prover uma entidade com os direitos legais para usar ou transferir um recurso para outras entidades
Anonimato	Ocultar a identidade de uma entidade envolvida no processo
Não-repúdio	Impedir a recusa de compromissos ou ações anteriores
Anulação	Revogação de certificação ou autorização

3.2 Criptografia

A palavra criptografia provem do grego "*kryptós*" que significa oculto, e "*grápheim*" que significa escritura, e sua definição é "*Arte de escrever com chave secreta de modo enigmático*" (LOPEZ, 2004).

Pode-se definir a criptografia como sendo um conjunto de métodos e técnicas para cifrar ou codificar informações legíveis por meio de um algoritmo, convertendo um texto original em um texto ilegível sendo possível, mediante processo inverso, recuperar as informações originais (MORENO et al., 2005). A Figura 3.1 mostra esse processo.

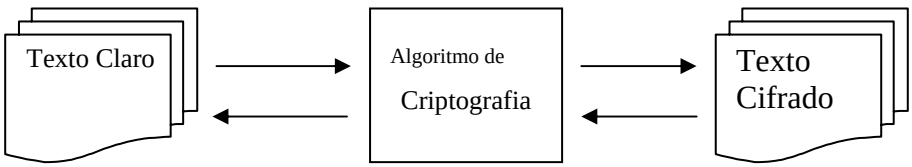


Figura 3.1. Esquema geral para cifragem de um texto (PEREIRA, 2004)

Uma definição um pouco mais técnica, é que a Criptografia é o estudo de técnicas matemáticas que relatam os aspectos da informação, assim como a confidencialidade, integridade da informação, autenticação de entrada e autenticação de dados na origem (MENEZES et al., 1996).

A criptografia pode ser utilizada basicamente por meio de códigos ou de cifras. Os códigos funcionam trocando partes da informação inicial por códigos predefinidos. Para se conseguir visualizar essa informação que foi codificada, deve-se conhecer quais códigos foram utilizados. Já as cifras são técnicas na qual a informação inicial é cifrada, ou seja, há a transposição e / ou substituição dos caracteres da informação inicial, gerando uma segunda informação não legível. Somente terá acesso ao significado da informação cifrada quem conhecer o processo de cifragem (MORENO et al., 2005). As cifras incluem o conceito de chaves, que dependendo do tipo de algoritmo utilizado, podem ser públicas ou privadas.

A criptografia é uma arte tão antiga quanto a própria escrita. Já estava presente no sistema de escrita hieroglífica dos egípcios e os romanos utilizavam códigos secretos para comunicar planos de batalha.

Um dado interessante é que a criptografia permaneceu estagnada até meados do século XX, tendo a partir deste momento um desenvolvimento acelerado com a utilização do computador, onde a criptografia realmente cresceu, incorporando complexos algoritmos matemáticos. Isso ocorreu após a Segunda Guerra Mundial (PEREIRA, 2004).

3.2.1 Criptografia tradicional

A criptografia tradicional baseava-se em sistemas de substituições monoalfabéticas (troca dos caracteres que compõe a mensagem, caractere a caractere, numa lógica ou seqüência pré-estabelecida), polialfabéticas (substituições de grupos de caracteres numa

seqüência pré-estabelecida), permutações ou transposições (alteração da ordem dos caracteres).

Outra técnica é conhecida como esteganografia, que consiste em ocultar a mensagem de forma que não seja percebida. Uma forma moderna de esteganografia são mensagens escondidas em imagens e que trafegam despercebidas pela internet (SILVA, 2003).

Um dos algoritmos de transposição monoalfabéticas mais antigos que se tem notícia é a chamada cifra de César, cuja autoria é atribuída ao Imperador Romano Júlio César. Consistia basicamente em somar a cada letra do alfabeto um valor fixo, de forma que ficasse ilegível. Para ler a mensagem bastava subtrair esse valor fixo somado para achar o significado da mensagem. Neste algoritmo já se podia notar o conceito de chave de cifragem, revolucionando a criptografia (PEREIRA, 2004).

A Figura 3.2. a seguir demonstra a cifra de substituição monoalfabética.

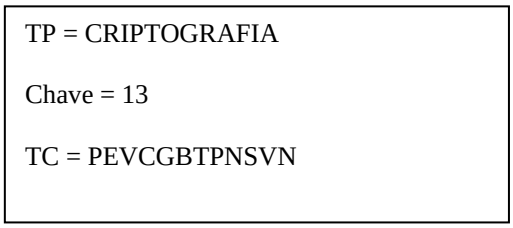


Figura 3.2. Cifra de substituição (SILVA, 2003)

Na Figura 3.3 vê-se o exemplo de cifra de transposição com chave = SENHA, os números sob essa palavra representam a sua ordem alfabética e representam a ordem em que as colunas serão lidas.

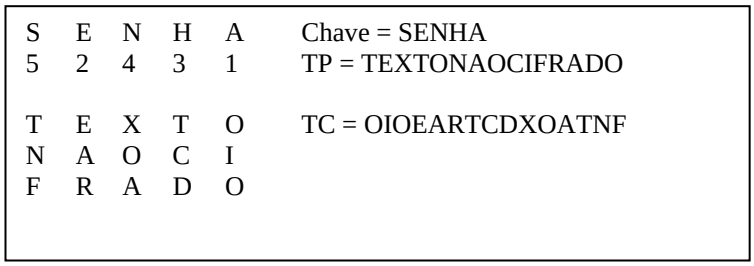


Figura 3.3. Cifra de transposição (SILVA, 2003)

As substituições polialfabéticas correspondem a aplicar-se “n” cifras monoalfabéticas. Um exemplo desta técnica é a cifra de Vigenere que utiliza como chave um conjunto de letras (l1,l2....ln). Primeiramente divide-se a mensagem “M” em grupos de “n” caracteres (m1,m2....mn) e aplica-se a seguinte operação, utilizando o alfabeto de 26 letras:

M’ = (m1 + l1) mod 26, (m2 + l2) mod 26 ..... (mn + ln) mod 26 (LOPEZ, 2004)

A figura 3.4 a seguir exemplifica a cifra de Vigenere, os números (6,17,8,19,14) correspondem a posição que as letras da chave (GRITO) ocupam no alfabeto.

M = VAMOSATACARAMANHASEMFALTA

Chave = GRITO = (G,R,I,T,O) = (6,17,8,19,14)

Divide-se M em bloco de 05 letras = VAMOS.ATACA.RAMAN.HASEM.FALTA

Após a substituição M’ = BRUHG.GKIVO.XRUTB.NRAXA.LRTMO

Figura 3.4. Cifra de Vigenere (CARVALHO, 2001)

A última forma de criptografia clássica utilizada antes da invenção dos computadores foram os sistemas rotores e que foram largamente utilizados durante a Segunda Guerra Mundial. A máquina ENIGMA, que é a mais famosa utilização desses sistemas, tinha a aparência de uma máquina de escrever comum, no entanto, em seu interior existiam 3 rotores que faziam a codificação. Quando se desejava mandar uma mensagem, digitava-se normalmente no teclado da máquina e em um painel luminoso as letras cifradas correspondentes ficavam visíveis, copiava-se então a mensagem cifrada e quando chegava ao destino digitava-se a mensagem cifrada e lia-se a mensagem original no painel luminoso (SILVA, 2003). A figura 3.5 apresenta um pequeno esquema do funcionamento da máquina ENIGMA.

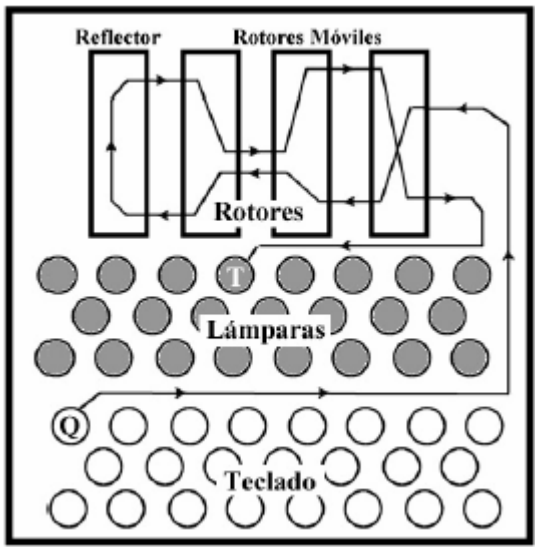


Figura 3.5. Máquina ENIGMA – Diagrama Simbólico (LOPEZ, 2004)

3.2.2 Criptografia moderna

Teve início com a invenção dos computadores, a qual possibilitou a evolução dos algoritmos de criptografia, não trabalhando com caracteres e sim diretamente nos bits.

Além da evolução dos algoritmos de criptografia, a invenção dos computadores possibilitou a quebra das antigas técnicas em poucos segundos, devido a alta capacidade de processamento presente nos computadores. A criptografia teve seu grande desenvolvimento em épocas de guerra (SILVA, 2003).

3.3 Primitivas Criptográficas

As primitivas criptográficas são estruturas elementares utilizadas para a construção dos criptossistemas e dos protocolos criptográficos. O modelo de classificação mais completa das diversas primitivas criptográficas é o proposto por (MENEZES et al., 1996) conforme Figura 3.6.

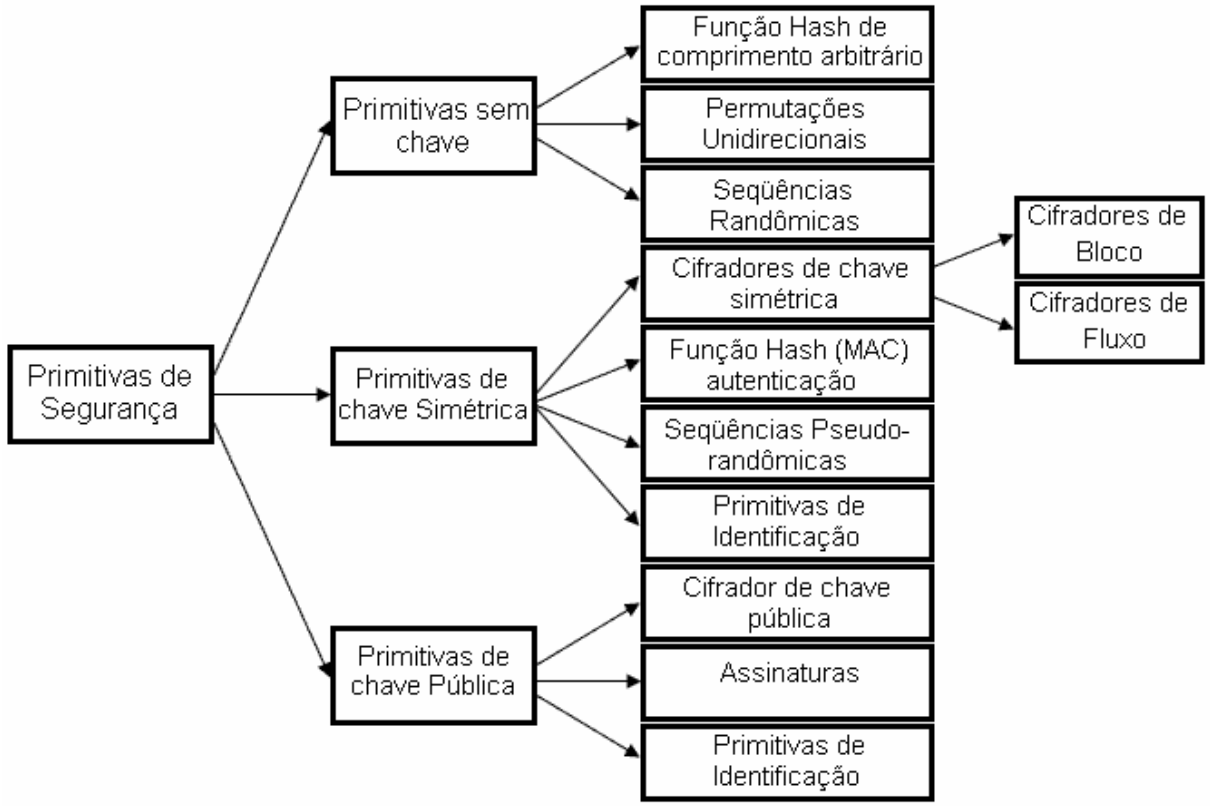


Figura 3.6. Classificação das Primitivas Criptográficas (MENEZES et al., 1996)

Além desta classificação, (MENEZES et al., 1996) sugere os seguintes critérios para avaliação de tais primitivas:

(i) Nível de segurança – Normalmente é calculada com base na quantidade de operações necessárias para alcançar o seu objetivo;

(ii) Funcionalidade – As primitivas precisam ser combinadas para atender aos objetivos de segurança da informação. Quais delas atendem melhor estas características é determinado pelas propriedades básicas desta primitiva;

(iii) Método de Operação – A utilização das primitivas depende dos modos de operação possíveis;

(iv) Desempenho – Refere-se a eficiência com que a primitiva executa sua função;

(v) Facilidade de Implementação – Refere-se a complexidade exigida pela primitiva para sua implementação tanto em hardware quanto em software.

3.4 Sistemas Critográficos

Um sistema criptográfico é como se fosse uma quíntupla (M, C, K, E, D), onde M é o conjunto de todas as mensagens não cifradas, C é o conjunto de todas as mensagens cifradas, K é o conjunto de chaves possíveis de se utilizarem o sistema, E é o conjunto de funções ou primitivas para cifrar M e obter C, e D é o conjunto de funções ou primitivas para decifrar C e obter novamente M (LOPEZ, 2004).

A notação desse sistema é Dk(Ek(m)) = m.

Existem duas correntes para a definição dos sistemas criptográficos. Para uma corrente, um sistema criptográfico é representado pelos algoritmos e pelo conjunto de todos os possíveis textos puros, textos cifrados e chaves de cifragem. Já para uma segunda corrente, um sistema criptográfico é um conjunto de primitivas utilizadas para fornecer serviços de segurança às informações (SILVA, 2003).

3.4.1 Classificação dos Sistemas Criptográficos

Os sistemas criptográficos dividem-se basicamente em dois tipos fundamentais:

(i) Criptossistemas de chave privada ou simétrica - utiliza a mesma chave tanto para o processo de cifragem como no de decifragem. A figura 3.7 traz um modelo de funcionamento do modelo simétrico. Como exemplo, o DES (*Data Encryption Standard*), onde ocorre o problema de distribuição de chaves. A chave tem que ser enviada para todos os usuários autorizados antes que as mensagens possam ser trocadas. Essa ação resulta num atraso e possibilita que a chave chegue a pessoas não autorizadas (MORENO et al., 2005).

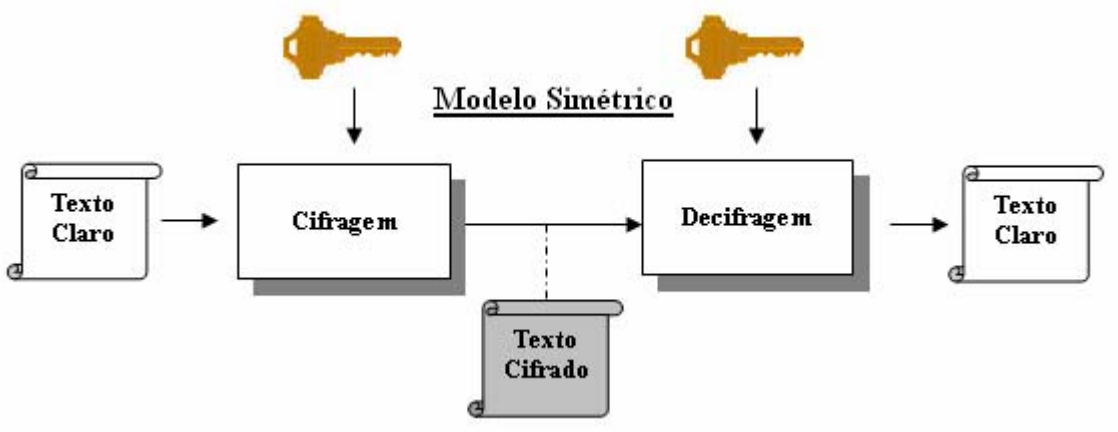


Figura 3.7. Funcionamento do modelo simétrico de criptografia (PEREIRA, 2004)

(ii) Criptossistema de chave pública ou assimétrica – utilizam um par de chaves, sendo uma delas pública e a outra privada. A criptografia de chaves públicas foi inventada em 1976 por Whitfield Diffie e Martin Hellman (DIFFIE e HELLMAN, 1976), a fim de resolver o problema de distribuição de chaves (MORENO et al., 2005). A figura 3.8 traz um modelo de funcionamento do modelo assimétrico. Normalmente neste tipo, cifra-se com a chave pública e decifra-se com a chave privada.

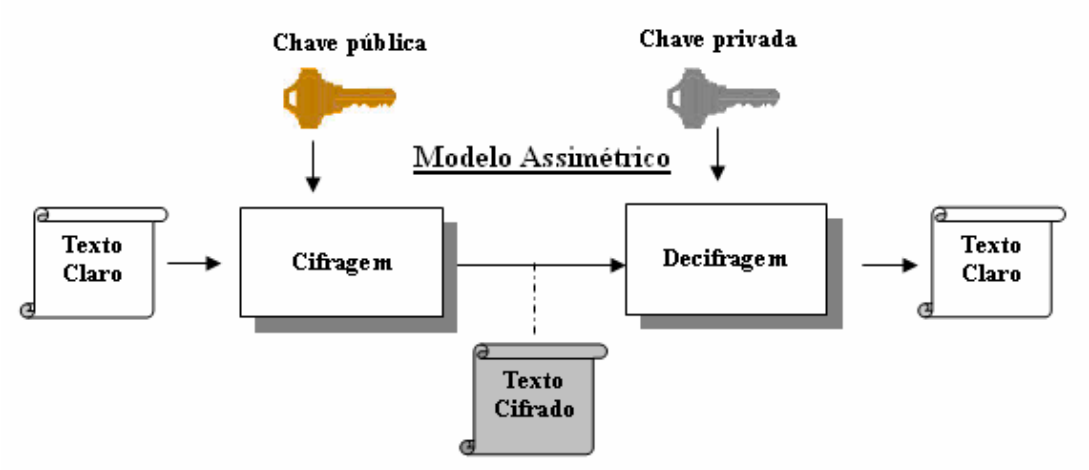


Figura 3.8. Funcionamento do modelo assimétrico de criptografia (PEREIRA, 2004)

No entanto as chaves são intercambiáveis, podendo-se cifrar com a chave privada e decifrar com a chave pública. Esta última serve como “assinatura digital”.

É baseada em cálculos matemáticos com números de elevada grandeza, normalmente a chave possui mais de mil bits, ocasionando uma menor velocidade de processamento se comparado aos algoritmos de chave simétrica. É utilizada quando não há a possibilidade de troca de chaves por um meio seguro (SILVA, 2003). A figura 3.9 mostra o modelo de geração de uma assinatura digital.

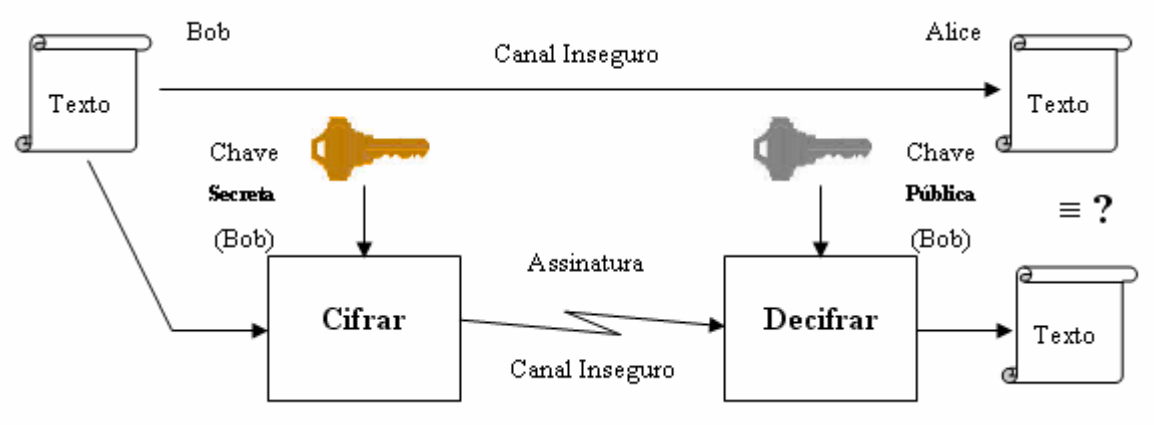


Figura 3.9. Geração de assinatura digital de um documento (PEREIRA, 2004)

Existe ainda a possibilidade de um modelo híbrido, que pode ser desenvolvido aproveitando-se as vantagens de cada tipo de algoritmo. O algoritmo simétrico, por ser mais rápido, é utilizado no ciframento da mensagem em si, enquanto o algoritmo assimétrico, embora lento, permite implementar a distribuição de chaves e é utilizado em aplicações de assinatura digital (MORENO et al., 2005).

Com o estudo sobre os principais conceitos gerais sobre criptografia, tipos de criptografia e sistemas criptográficos, obteve-se uma base conceitual para a definição dos algoritmos de segurança utilizados no PERS. Esta segurança foi um dos pontos principais desde o início do projeto.

3.5 Segurança em Redes de Sensores

Toda a rede para ser considerada segura deve cumprir determinados requisitos. O importante é saber, diante da aplicação em questão, quais dos objetivos (HU et al., 2003) são relevantes e devem ser levados em consideração pelo administrador da rede durante a fase de escolha do algoritmo a ser utilizado para não sobrecarregar os sensores, dadas as suas limitações (PERRIG et al., 2001).

Num algoritmo de criptografia para redes de sensores existe um compromisso entre a segurança provida pelo algoritmo e a quantidade de energia que ele utiliza para criptografar dados, decriptografar, enviar dados, receber, processar informações, verificar assinaturas e a quantidade de energia armazenada num sensor é o seu principal obstáculo ou limitação (AKYILDIZ et al., 2002).

Outro fator relevante é o comportamento durante o processo em que o sensor fica em *standby* para economizar energia. Nesse momento, os sensores podem perder o sincronismo necessário para o funcionamento dos algoritmos de segurança, pois existe a troca de informações que são utilizadas durante o processo de atualização de chaves. Se um nó perder tais informações poderá ficar impedido de trocar informações com a rede. Além do que esse mecanismo de *standby* deve ser cuidadosamente utilizado, porque o fato do sensor entrar e sair desse estado muitas vezes pode gastar mais energia do que se estivesse ligado o tempo todo (AKYILDIZ et al., 2002).

3.5.1 Segurança em nível de camadas

Algumas características e ataques podem ser específicos de determinadas camadas do modelo OSI, os ataques são mais incisivos se ocorrerem sobre a camada de enlace e de rede. A seguir alguns dos tipos de ataques que ocorrem nas camadas (AKYILDIZ et al., 2002).

3.5.1.1 Camada de enlace

Em ambientes sem fio, segurança em nível de enlace é mais crítica que nas redes cabeadas devido à característica do meio ser aberto.

Ataques à camada de enlace, mais precisamente à sub-camada MAC (*Media Access Control*), podem prejudicar a rede ao nível de pacote. Isso pode ser feito através de indução de colisões, danificação de pacotes de dados ou de controle. Porém, esses ataques podem ser detectados através do *checksum* e corrigidos. O que isso pode ocasionar é a repetição das mensagens até que elas consigam ser recebidas corretamente se for utilizado algum mecanismo de confiabilidade para transferência de dados. Para roubar informações será necessário muito tempo de interceptação de mensagens para ser possível a extração de uma quantidade suficiente de dados úteis (HU et al., 2003).

3.5.1.2 Camada de rede

As redes de sensores provêem segurança na camada de rede que é a mais afetada e a que causa maiores danos. Isso se deve a sua característica de transmissão ser por múltiplos saltos, o que obriga que os dados passem por nós intermediários até atingir o seu destino podendo representar vulnerabilidades.

A seguir, mostram-se os diversos tipos de ataques que prejudicam o roteamento e a transferência de dados, com informações obtidas em (KARLOF e WAGNER, 2003).

a) Spoofing, alteração ou repetição de informações de roteamento é tipo de ataque que causa loops na rede, atrai ou repele tráfego, gera mensagens de erro de rota falsas, divide a rede, dentre outros danos. Tudo por ter como alvo principal os pacotes de controle

responsáveis pelas informações de roteamento, através de repetições ou modificações dos mesmos.

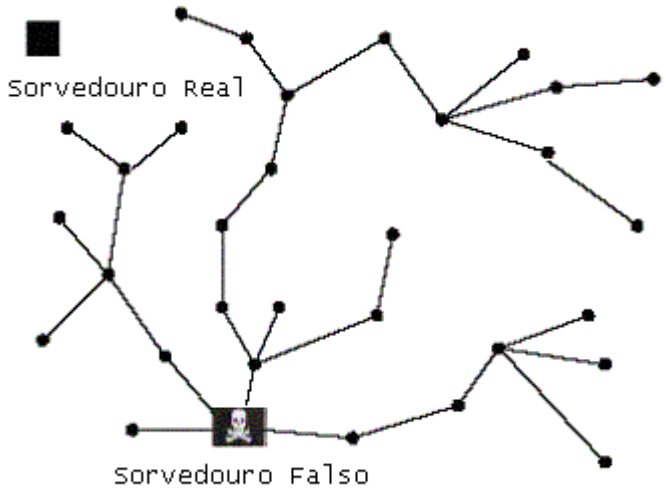


Figura 3.10. Ataque de *Spoofing* (KARLOF e WAGNER, 2003)

b) Encaminhamento seletivo acontece quando um nó malicioso se recusa a encaminhar determinados pacotes descartando-os, funcionando de forma não colaborativa com a rede. Esse tipo de ataque pode acontecer devido às características de transmissão da informação ser salto a salto, onde os nós têm responsabilidade de encaminhar pacotes vindo de seus vizinhos. Um nó malicioso pode funcionar como um buraco negro ao não encaminhar os dados recebidos independente de quem enviou.

Com o passar do tempo os pacotes não seriam mais encaminhados a ele, por esse nó não dar continuidade na transmissão dos dados. Isso poderia representar um defeito nesse nó ou através de algum mecanismo de detecção de intruso representaria uma anomalia.

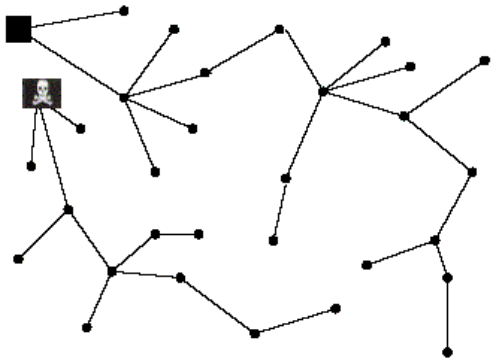


Figura 3.11. Ataque de encaminhamento seletivo (KARLOF e WAGNER, 2003)

c) Ataque sorvedouro acontece quando o tráfego é desviado para um determinado nó malicioso. Os nós vizinhos ou o próprio nó podem manipular os dados e fazer modificações. Esse tipo de ataque acontece porque os adversários podem alterar as mensagens de roteamento. Essa atitude faz com que um nó se torne atraente aos nós vizinhos fazendo parte de suas rotas, podendo inclusive atingir outros através de inundações da rede com rotas falsas.

d) Ataque *Sybil* é o tipo de ataque que o nó pode apresentar múltiplas identidades e assim se fazer passar por outros controlando grande parte da rede. Os nós afetados acham que um nó malicioso que esteja aplicando esse tipo de ataque representa nós disjuntos quando na verdade não o é. Novamente, quanto maior o poder do nó mais efetivo pode ser o seu ataque por aumentar a sua área de influência. Em DOUCEUR (2002), para se defender desse ataque pode se introduzir um nó confiável ou mais de um que faça o papel de agência certificadora de identidades dos nós.

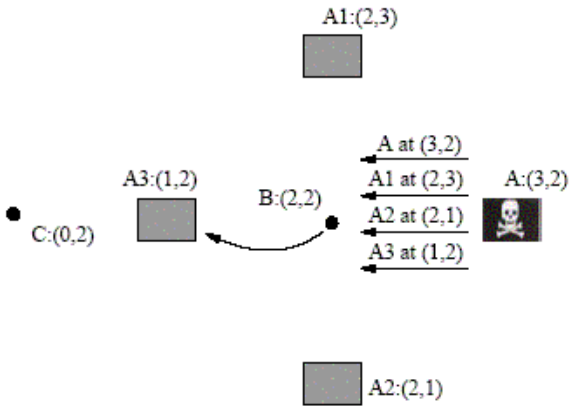


Figura 3.12. Ataque de *Sybil* (KARLOF e WAGNER, 2003)

e) *Wormholes* é um túnel criado pelo atacante. As mensagens entram nesse túnel numa parte da rede e são propagadas até uma outra parte, normalmente esses túneis possuem uma latência superior.

Um wormhole é instanciado por dois nós maliciosos que ficam nas extremidades do túnel. Cada um deles irá convencer os nós vizinhos que o *wormhole* é o melhor caminho através de transmissão de pacotes de roteamento com métricas forjadas que façam o túnel mais atraente diante das outras possibilidades. Essa transmissão pode ser feita por inundação. Quanto mais próximo do nó sorvedouro mais informação passará por dentro desse túnel.

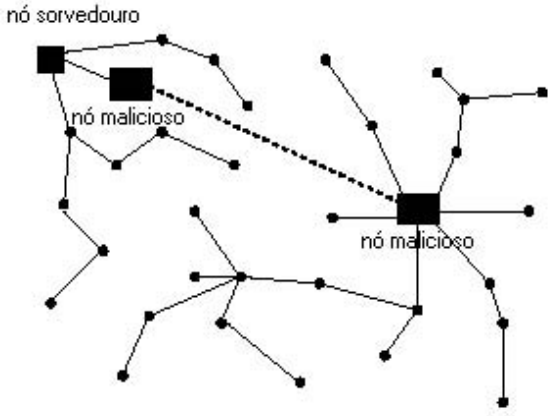


Figura 3.13. Um *wormhole* entre dois nós maliciosos (KARLOF e WAGNER, 2003)

f) Ataque de inundação de HELLO quando um nó mal intencionado envia pacotes de HELLO para qualquer nó da rede desde que possua um transmissor capaz. Assim os sensores ao receber esses pacotes julgam esse nó como vizinho e começam a aceitar as rotas que são anunciadas por ele. Essas rotas anunciadas vão induzir os nós a encaminhar seus pacotes por onde o nó malicioso quiser.

A inundação que esse tipo de ataque menciona não é feita em múltiplos saltos e sim num único salto. Portanto, a inundação de pacotes de HELLO se dá num único salto por ser feito por um nó de maior porte.

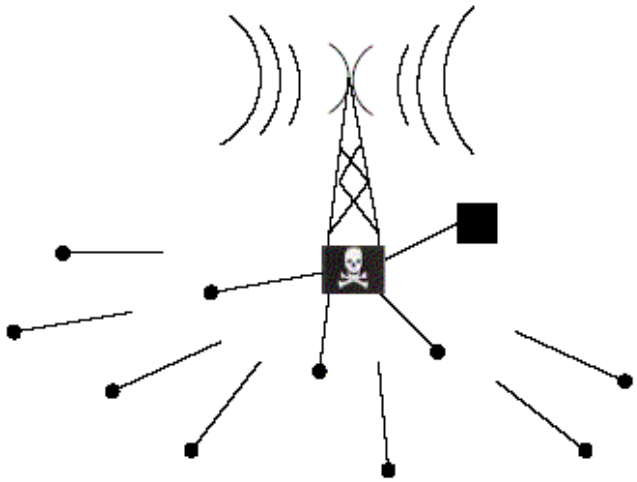


Figura 3.14. Ataque de inundação de HELLO (KARLOF e WAGNER, 2003)

g) Spoofing de reconhecimento positivo esse tipo de ataque pode ser utilizado para fazer parecer que um canal ruim ou um nó que já esteja fora da rede ainda está funcionando normalmente. Isso pode ser feito após um nó transmissor receber um reconhecimento positivo vindo de um nó malicioso.

Há então a transferência da mensagem pelo nó atacante. O reconhecimento positivo é característica de alguns algoritmos de transmissão de dados. Podendo inclusive ser ao nível MAC no caso de estar sendo usado o padrão IEEE 802.11.

3.5.2 Algoritmos de Segurança com nível de roteamento

Os algoritmos que mostram melhor desempenho são aqueles que conseguem proteger da melhor forma a aplicação e ao mesmo tempo consumir o mínimo de energia possível.

Conforme foi se verificando a importância de introduzir mecanismos de segurança em redes de sensores, maior foi a necessidade da elaboração de algoritmos capazes de prover tais funcionalidades. As primeiras soluções encontradas foram incluir mecanismos de segurança em protocolos de roteamento já existentes, principalmente nos que já eram utilizados em redes ad hoc.

As soluções integradas são as mais eficientes que as que tentaram incluir segurança num segundo estágio. A maioria das propostas foi feita em nível de camada de rede, pois é onde os ataques se fazem mais incisivos como já pôde ser observado.

A maioria dos algoritmos propostos aplica criptografia de chave simétrica, pois a de chave pública consome muita energia. Segundo PERRIG et al. (2001), as variáveis necessárias para fazer os cálculos das chaves não caberiam na memória de um sensor. A propagação em broadcast também é um importante obstáculo a ser contornado, principalmente na questão de distribuição de chaves por não representar um meio confiável. A seguir apresentam-se alguns algoritmos de seguranças para redes de sensores.

a) **INSENS (*Intrusion-tolerant routing protocol for wireless sensor networks*)** (DENG et al., 2002) já leva em consideração a possibilidade da existência de alguns nós maliciosos, por isso é capaz de detectá-los e não utilizá-los para as tarefas da rede. O INSENS parte do princípio que um nó malicioso consegue prejudicar a sua vizinhança apenas, mas não toda a rede.

O INSENS tem como objetivo prevenir ataques do tipo de negação de serviço realizados através de inundações da rede ao limitar o tipo de comunicação. Somente a estação

rádio base tem permissão de fazer inundações. Para tal, a Estação Rádio Base (ERB) é autenticada para que nenhum nó malicioso se faça passar pela mesma. Para comunicação unicast todo o pacote deve passar pela ERB para que esta funcione como um filtro. Essa característica introduz infra-estrutura à rede de sensores.

Para prevenir o anúncio de rotas falsas a informação de controle de roteamento é autenticada. Para economizar energia, é utilizada criptografia simétrica para atingir confidencialidade e autenticação entre os nós comuns e a ERB. A ERB é utilizada para a disseminação e processamento das tabelas de roteamento, portanto os nós da rede apenas mantêm as tabelas recebidas e não as transmitem, evitando assim, a injeção de tabelas de roteamento falsas na rede.

Essa atitude minimiza computação, comunicação, armazenamento, e largura de banda necessária pelos nós sensores. Em compensação o nó sorvedouro irá necessitar de aumento de computação, comunicação, armazenamento, e largura de banda (HU et al., 2003).

O INSENS utiliza múltiplos caminhos para transferência de dados introduzindo redundância. Essa proposta tem como objetivo o aumento da robustez, pois se caso uma rota venha a ser comprometida devido à presença de um possível intruso, outros caminhos poderão ser utilizados.

b) Ariadne (HU et al., 2002) é um protocolo de roteamento sob demanda para redes ad hoc seguro que pode ser utilizado também em redes de sensores. Esse protocolo previne atacantes de alterar rotas e nós que estejam livres de qualquer intenção maliciosa.

O Ariadne emprega chaves simétricas e previne contra ataques de negação de serviço. Cada nó deve gerar sua própria cadeia de sentido único de chaves. As suas restrições de memória impedem que sejam geradas cadeias de chaves muito longas o que pode proporcionar um gasto de tempo muito grande de cálculos de chaves.

c) SPINS (*Security protocols for sensor networks*) (PERRIG et al., 2001) é composto por dois protocolos. O µTESLA é responsável por prover autenticação quando há comunicação em broadcast e o SNEP é responsável pela confidencialidade, autenticação da comunicação ponto a ponto e atualização dos dados com baixo *overhead*.

O SNEP confia num contador compartilhado entre transmissor e receptor que é utilizado como um vetor de inicialização para o algoritmo usado para criptografar e decriptografar os dados, no caso do SNEP a criptografia é realizada por um RC5 enxuto devido às limitações dos sensores. Como ambos participantes possuam o contador e o incrementam após cada bloco de dados criptografados, o contador não precisa ser enviado a cada transmissão.

Para autenticar transmissor e receptor e manter a integridade dos dados é utilizado um código de autenticação de mensagem.

A função de criptografia é aplicada a um contador com crescimento monótono para gerar uma única palavra que será multiplicada pelo texto plano num XOR. O processo de decriptografia é idêntico, conforme figura 3.15.

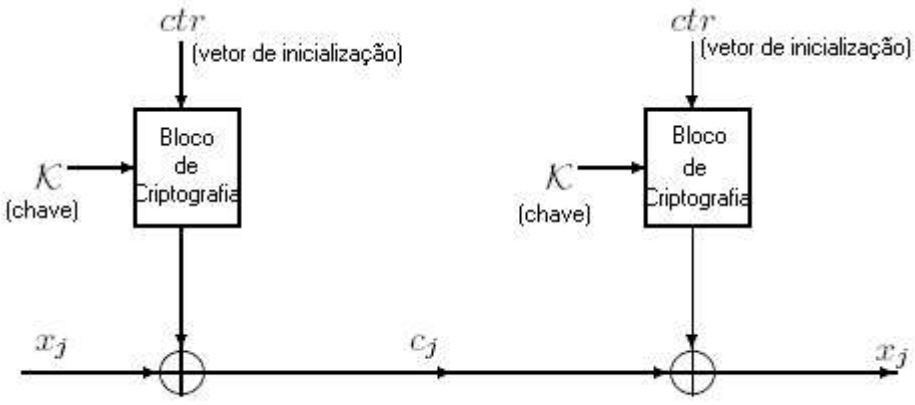


Figura 3.15 Contador utilizado para criptografar e decriptografar (PERRIG et al., 2001)

Segundo LIU e NING (2003), o µTESLA utiliza um método para autenticar comunicação em broadcast a partir de chaves simétricas emulando assimetria para que nenhum receptor não autorizado consiga obter a chave. Para isso ela envia em ponto a ponto a cada nó participante da rede os parâmetros necessários para a comunicação ser segura e para o algoritmo poder funcionar. A autenticidade desses parâmetros é garantida por uma assinatura digital. Existem propostas que tentam otimizar esse processo de transmissão de parâmetros para que não seja ponto a ponto, pois numa rede com muitos nós esse processo induziria um grande atraso.

A assimetria que o µTESLA introduz é devido à característica do protocolo de sempre atualizar a chave criptográfica simétrica e somente transmiti-las em broadcast no final de intervalos de tempo pelas ERBs. A partir dessa chave, os receptores terão condição de construir cadeias de chaves e assim autenticar as chaves recebidas, pois ao receber a chave essa deve pertencer a cadeia de chaves computadas através de uma função aleatória.

Só então as mensagens poderão ser decriptografadas. Dentro desse intervalo de tempo todos os nós utilizam a mesma chave. Essa cadeia de chaves é obtida a partir de um dos parâmetros que foi recebido no início do processo.

Ataques de repetição são evitados porque os nós têm como identificar a que intervalo pertence à chave recebida e, portanto não a utilizam posteriormente. A estação base ou nó sorvedouro novamente é considerado fora de risco de ataques e, portanto é confiável.

3.5.3 Gerenciamento de Chaves

O gerenciamento de chaves é o processo em que as chaves criptográficas são geradas, armazenadas, protegidas, transferidas, carregadas, usadas e destruídas.

Esse gerenciamento é problemático em redes de sensores por serem vulneráveis a manipulações devido às suas limitações de custo, por não poderem armazenar muitas chaves devido às suas limitações de espaço em memória e por não poderem utilizar algoritmos de criptografia mais robustos devido às suas limitações de energia.

Segundo LAW et al. (2003), para cumprir os requerimentos funcionais e de segurança da maioria das redes de sensores deve-se levar em consideração os requisitos a seguir.

(i) Uma rede de sensores não deve trabalhar com uma única chave, pois devido a sua falta de proteção ter uma chave somente e não ter nenhuma faz o mesmo efeito.

(ii) Uma rede não deve ter um nó centralizador ou ponto de vulnerabilidade. O SPINS parte do princípio que o nó sorvedouro está livre de qualquer falha.

(iii) Devem ser respeitados critérios de escalabilidade para que a adição de novos nós possam ser feitos a qualquer momento sem causar aumentos excessivos ao nível de processamento por nó, de comunicação e de overhead administrativo na rede.

Existem dois tipos de esquemas para a distribuição de chaves em redes de sensores. Um tipo aberto a toda a rede e um tipo específico por nó. O tipo aberto à rede equipa todo o nó da rede com a mesma chave e iguala o comprometimento de um único sistema de chaves com o comprometimento de toda a rede. Se houver o roubo de informações, a rede estará completamente comprometida. O tipo específico por nó determina uma única chave para toda a combinação de nós que estão se comunicando. A segurança atingida por esse esquema é otimizada, entretanto o hardware necessário para armazenar está fora das possibilidades dos sensores (HU et al., 2003).

O que se faz então é tentar encontrar soluções intermediárias que não sejam tão eficientes, mas que também não sejam tão vulneráveis.

Existem outras propostas para a distribuição segura de chaves. Dentre elas a feita por (CHAN et al., 2003) que propõe três tipos de métodos distintos, o *q-composite random key predistribution scheme* que atinge uma grande proteção sobre ataques de baixa escala enquanto troca um aumento da vulnerabilidade a ataques físicos em grande escala aos nós da rede. Essa vulnerabilidade ocorre porque o atacante tem condição de agregar muitas informações e não mais encontraria resistência da rede. Outro tipo é o *multi-path key reinforcement scheme*, que aumenta a segurança da rede ao transmitir a chave por múltiplos caminhos.

Por último, há a proposta *random-pairwise keys scheme*, que garante que mesmo se alguns nós estiverem comprometidos, a rede continua completamente segura. Isso ocorre porque a comunicação entre dois nós é sempre feita baseado no reconhecimento da chave que está sendo utilizada pelo par, como uma forma de autenticação.

Os conceitos de segurança e segurança em redes de sensores apresentados neste capítulo contribuíram muito na implementação do PERS. Desde o início do projeto a segurança sempre foi uma opção e com esses estudos, foi escolhido a utilização de algoritmos de chave simétrica no PERS, pois consegue conciliar as limitações de energia com o máximo de segurança possível às redes de sensores.

4. Processadores Específicos de Redes

Com o avanço tecnológico na área da microeletrônica, outras características vêm sendo incorporadas ao longo das últimas décadas aos microprocessadores, como unidades de gerenciamento de memória, memória *cache*, co-processador aritmético, etc, tornando-os cada vez mais complexos (PRADO, 2004).

Esses processadores podem ter aplicações diferentes, desde a CPU da placa-mãe de um computador, até os sistemas embarcados, passando por placas de vídeo 3D, som e placas de rede até roteadores (PRADO, 2004).

O surgimento de novas aplicações na área de redes de sensores promove uma constante busca por alternativas e arquiteturas que visem melhorar a performance.

Para isso é proposto o PERS – Um Processador Específico para Redes de Sensores que é baseado no NPSoC – Um novo processador de Rede (PRADO, 2004). O NPSoC foi o primeiro processador de rede, no Brasil, implementado em VHDL e prototipado em FPGAs.

As principais características do NPSoC são descritas na seguinte seção.

4.1 NPSoC – Um novo Processador de Rede (PRADO, 2004)

O NPSoc é baseado no RCNP Processador de Rede com Suporte a Multi-protocolo e Topologias Dinâmicas (FREITAS e MARTINS, 2000) e (FREITAS e MARTINS, 2001) e no R2NP (*Reconfigurable RISC Network Processor*) – Processador de Rede RISC Reconfigurável (FREITAS e MARTINS, 2002). Estes processadores não foram implementados ainda, somente um módulo, o do *CrossBar*[5], foi simulado na ferramenta Xilinx. Espera-se uma futura integração do trabalho aqui realizado com o módulo Crossbar da

[5] *CrossBar*: Chave comutadora de pacotes emulando conexões elétricas diretas.

PUC-MG. No NPSoC, foi definido um conjunto de instruções simples, porém que dá cobertura ao funcionamento do processador, como seus registradores, ULA (Unidade Lógica e Aritmética), a UC (Unidade de Controle), PC (Contador de Programa), RI (Registrador de Instruções) e a maioria das instruções de roteamento, necessárias para a execução de alguns algoritmos propostos para a simulação do funcionamento de algumas redes. Portanto para o desenvolvimento do NPSoC, deu-se especial atenção à estes dois processadores, RCNP e R2NP, os quais somente trabalharam em nível de simulação (PRADO, 2004).

4.1.1 Arquitetura do NPSoC

O NPSoC possui as características do RCNP, portanto modifica-se a proposta inicial devido à complexidade na implementação das *microengines*, as quais são verdadeiras caixas pretas, dificultando assim a visualização de seu funcionamento, mas não perdendo a característica RISC, já que a quantidade de instruções ficou bastante reduzida pelo fato de que só foram implementados alguns algoritmos de roteamento (para redes tipo anel, hipercubo e árvore) (PRADO, 2004). Sua arquitetura ficou então definida como se segue:

- 8 portas de saída nomeadas de B1 a B8;
- 8 registradores de 32 bits;
- ULA;
- Contador de Programa;
- Unidade de Controle;
- Registrador de Instruções;
- Registrador de Endereços.

Na figura 4.1, mostra que o NPSoC é um processador com registradores específicos de roteamento (SEC, PUT e CTRL), porém com todos os componentes necessários para o

funcionamento de qualquer processador, como contador de programa, unidade de controle, unidade lógica e aritmética e registrador de instruções (PRADO, 2004).

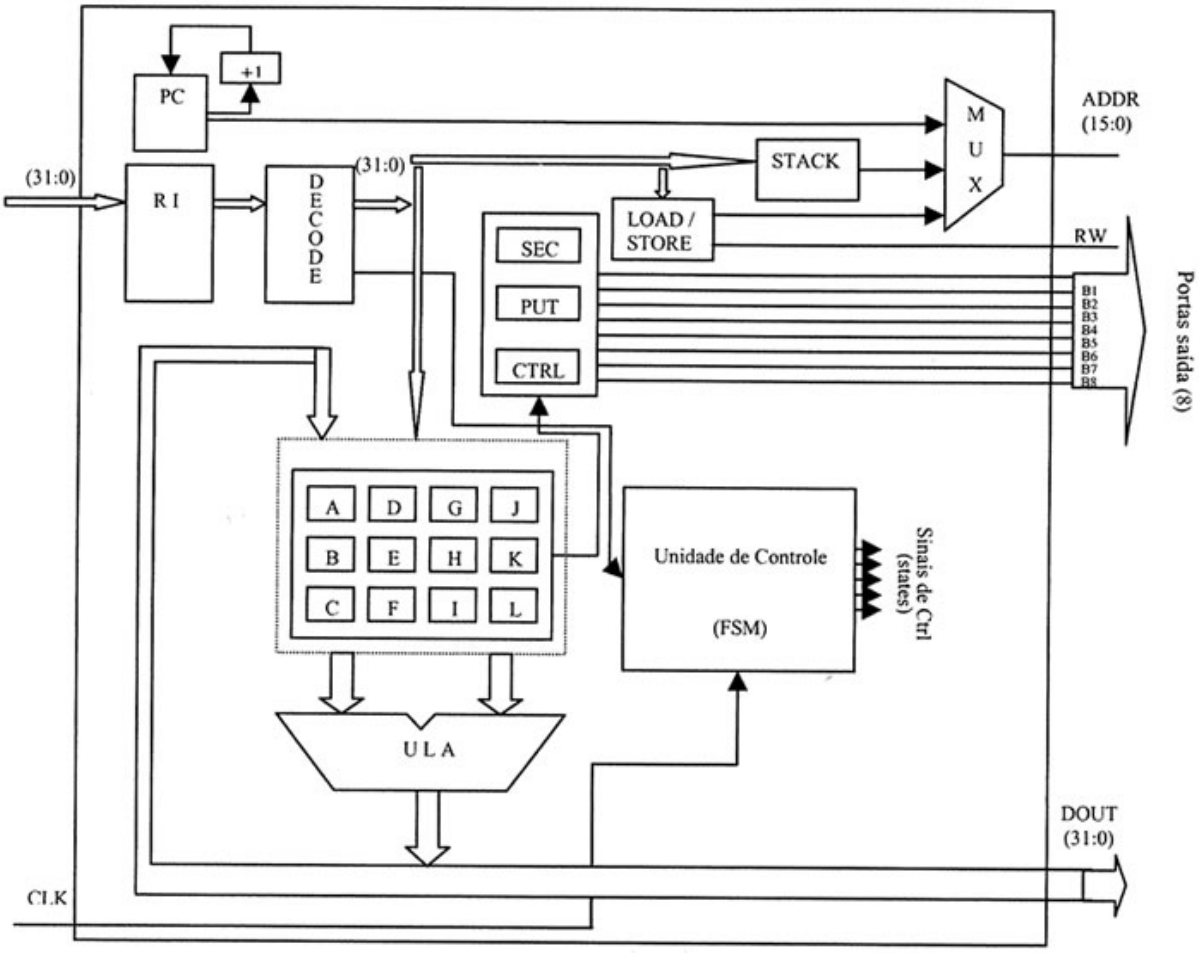


Figura 4.1. Arquitetura detalhada do NPSoC (PRADO, 2004)

Segundo PRADO (2004), o processador depende de uma Unidade de Controle para realizar suas operações de controle, o que, especificamente neste processador, se resolveu fazer utilizando o método de FSM (*Finite State Machine*), ou seja, máquina de estados finita, onde seus estados são muito bem definidos e respeitam o ciclo de busca, decodificação e execução dos computadores Von Neumann. O ciclo se executa desta maneira:

Passo 1: É feita a busca da 1ª instrução;

Passo 2: Soma-se 1 para o contador de programa, ou seja, atualiza o contador;

Passo 3: Decodifica a instrução, através de seus *opcodes*;

Passo 4: Se essa instrução trouxer dados, serão armazenados em registradores internos;

Passo 5: Executa a instrução, normalmente isso é feito pela ULA;

Passo 6: Registra os resultados em local apropriado: memória, *buffer* e etc; Retornando ao passo 1, ou seja, irá buscar a próxima instrução, dando continuidade ao ciclo.

A comunicação com esse local apropriado é feita por meio de um barramento cuja função é levar os dados da ULA para a memória, instruções para o RI, realimentar os registradores da ULA e etc. Para isso foi necessária a implementação de uma memória (uma pilha) de 512 Kbytes, como se pode observar na Figura 4.2 (PRADO, 2004).

No NPSoC, além das instruções comuns em qualquer processador, como instruções aritméticas e lógicas, de manipulação e de desvio, tem-se a implementação de um módulo especial com instruções específicas de roteamento (PRADO, 2004).

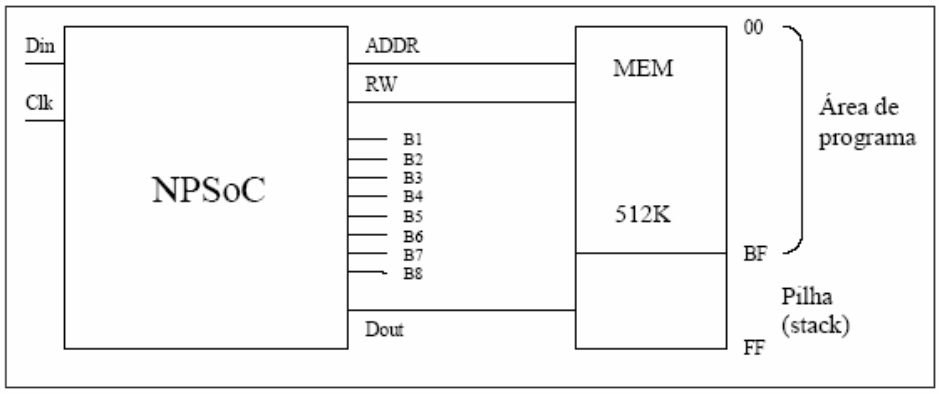


Figura 4.2. *Top Level* da arquitetura NPSoC (PRADO, 2004)

4.1.2 Conjunto de instruções (ISA)

O conjunto de instruções do Processador de Rede NPSoC segue o modelo RISC, onde, com apenas 32 instruções, é possível descrever a maioria dos algoritmos de roteamento presentes nos NPs comerciais atuais. As instruções foram divididas em quatro classes (PRADO, 2004):

Lógicas e Aritméticas:

- AND: Faz a operação AND entre dois registradores;
- OR: Faz a operação OR entre dois registradores;
- XOR: Faz a operação XOR entre dois registradores;
- ADD: Soma dois registros;
- SUB: Subtrai dois registros;
- INC: Incrementa um registro;
- DEC: Decrementa um registro;
- NEG: Subtrai o operando destino de 0 e retorna o resultado a este mesmo destino.

São aquelas utilizadas para realizar as operações aritméticas e as comparações lógicas necessárias nas tomadas de decisões.

Não se fez necessária a implementação de multiplicador e nem de divisor.

Movimentação:

- LDA: Guardam os dados indo para a memória;
- LDI: Atribui um valor para o registrador;
- MOV: Transfere os dados entre celulas de memória e registradores;
- STR: Guardam os dados vindos da memória;
- STX: Inverte os operandos;
- SPUSH: Coloca o conteúdo de um registro na pilha;
- ROD: Rotação à direita;
- ROE: Rotação à esquerda;
- SPOP: Retira um valor da pilha e coloca em um registro.

São aquelas utilizadas para executar a troca de valores entre registradores, rotação, retirada e colocação de dados na pilha e rotação.

Desvio:

- JMP: Faz um desvio incondicional
- JZ: Desvia se zero
- JNZ: Desvia se não zero
- JMI: Desvia se maior ou igual
- JMZ: Desvia se maior que zero

Especiais:

- BRC: Faz broadcasting para todas as portas de saída com o conteúdo do buffer de entrada
- PUT B: Atribui ao registrador PUT o conteúdo do registrador B
- SEC: Atribui ao registrador SEC o conteúdo do buffer
- FCX: Fecha conexão entre um buffer temporário e a porta de saída
- PUT: Indica a porta que o próximo buffer a ser escolhido para análise do pacote, será feito através do próprio hardware.
- SAI: Envia dados do registrador para a saída escolhida.
- ENT: Atribui ao registrador A o conteúdo do buffer da porta de entrada
- SUI: Recebe dados do buffer para um registrador escolhido
- SETDSC: Seta em registradores dos buffers temporários qual o valor que, contido no pacote, desfaz a conexão feita por FCX.
- SET C, D, E: Configura os registradores com o local exato onde encontrar a informação do tamanho do pacote que será recebido.

As instruções especiais se referem àquelas utilizadas para o roteamento e transição de dados entre os *buffers* reconfiguráveis e os registradores, bem como o seletor de conexões.

4.1.3 Estágio atual do NPSoC

O grupo de pesquisa da PUC-MG não chegou a implementar o processador propriamente dito em VHDL e conseqüentemente em FPGAs. Ao invés disso, eles desenvolveram um ambiente de simulação para auxiliar no ensino de Redes, que serve também como montador *assembler* e permite a execução de algoritmos fazendo uso de seu conjunto de instruções. Na verdade a única implementação em VHDL feita pelo grupo em questão foi o da chave *Crossbar* (PRADO, 2004).

Segundo PRADO (2004) o NPSoC, é o primeiro processador de rede, no Brasil, implementado em VHDL e prototipado em FPGAs. Os componentes principais do processador NPSoC já estão em pleno funcionamento, como: ULA, PC, RI, Registradores entre outros e ainda uma parte das instruções especiais ou de roteamento, necessárias para a execução dos algoritmos sugeridos.

Este capítulo mostrou a importância dos processadores de redes, que são um excelente campo de pesquisa e desenvolvimento e serviu como base para a implementação de nosso PERS – Processador Específico para Redes de Sensores.

O PERS aproveitou o conjunto de instruções Lógicas e Aritméticas, Movimentação e Desvio do NPSoC, bem como adicionou as instruções especiais específicas de Sensores e Redes e também instruções de segurança descritas em PEREIRA (2005) no Criptoprocessador VLIW para Algoritmos Criptográficos Simétricos.

5. Processador Específico para Redes de Sensores

Este capítulo tem como objetivo apresentar o PERS, um processador específico para redes de sensores com primitivas de criptografia. O processador proposto é baseado no NPSoC Um novo Processador de Rede (PRADO, 2004) e no Maté Uma Máquina Virtual para Redes de Sensores (LEVIS e CULLER, 2002). No PERS, foi definido um conjunto reduzido de instruções que, dá suporte ao funcionamento do processador através de seus registradores, ULA (Unidade Lógica Aritmética), a UC (Unidade de Controle), PC (Contador de Programa), RI (Registrador de Instruções) e as instruções específicas de roteamento que fazem o envio dos dados coletados pelo próprio sensor ou recebidos de outros sensores.

5.1 PERS – Um Processador Específico para Redes de Sensores

O PERS é um processador simples com o propósito de coletar informações de um determinado local (por exemplo, temperatura) através de sensores, processá-las e enviá-las para a estação base para assim executar ações necessárias de acordo com as informações coletadas.

Sua arquitetura foi baseada no NPSoC, mas foi modificado para se adequar à necessidade de funcionamento de uma rede de sensores, que possui características diferentes com relação a topologia, pois nesta primeira versão foi utilizada a topologia ponto-a-ponto. Sua arquitetura ficou então definida como segue:

1. Uma porta de entrada
2. Uma porta de entrada e saída
3. Dois registradores de propósito gerais de 20 bits
4. Quatro registradores específicos de roteamento de 4 bits
5. Um registrador específico de roteamento de 20 bits

6. ULA
7. Contador de Programa
8. Unidade de Controle
9. Registrador de Instruções;

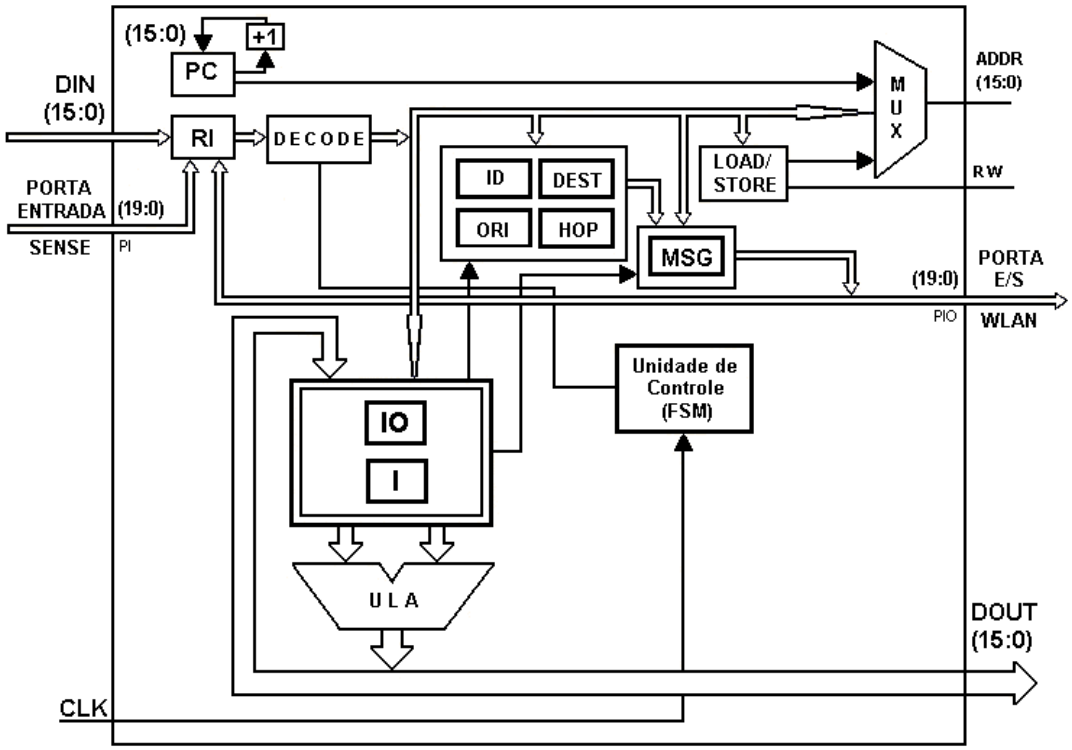


Figura 5.1. Arquitetura detalhada do PERS

Na figura 5.1 pode-se verificar que a Unidade de Controle controla todo o processador. Neste processador, usou-se o método de FSM (*Finite State Machine*), ou seja, máquina de estados finita, onde seus estados são bem definidos e respeitam o ciclo de busca, decodificação e execução dos computadores Von Neumann. O ciclo se executa da maneira seguinte:

Passo 1: É feita a busca de uma instrução. Imediatamente, soma-se 1 para o contador de programa.

Passo 2: Decodifica a instrução;

Passo 3: Se a instrução tem dados, eles são armazenados em registradores internos;

Passo 4: Executa a instrução;

Passo 5: Armazena os resultados em registradores internos; Retornando ao passo 1, ou seja, busca a próxima instrução, dando continuidade ao ciclo de execução.

Como pode ser observado, a arquitetura do processador possui duas portas: uma de entrada na qual será ligado ao sensor; e outra de entrada e saída que será ligada na rede.

Esta versão do PERS contempla mais instruções específicas de roteamento, não foram implementadas muitas instruções comuns contidas nos processador convencionais (tais como instruções aritméticas e lógicas, de manipulação e de desvio).

A mensagem de dados que é transmitida pela rede dos sensores possui 20 bits, a figura 5.2 ilustra a composição da mensagem de dados.

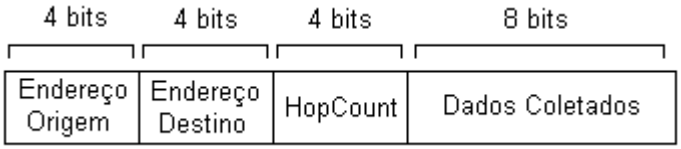


Figura 5.2. Composição da Mensagem de Dados

Pode-se observar que são necessários 8 bits para a informação coletada, 4 bits para o endereço origem, 4 bits para o endereço destino e 4 bits para o *hopcount* (quantidade de saltos), essas informações são importantes para identificar quem está enviando e para qual destino a mensagem de dados será enviada. Este tamanho foi definido porque limitou-se a rede em 16 nós sensores e com isso, necessita-se de 4 bits para seu endereçamento.

5.2 Conjunto de instruções (ISA) – Versão Inicial

O conjunto de instruções do Processador Específico para Redes de Sensores PERS segue o modelo RISC, pois possui um número pequeno de instruções. Atualmente com 20

instruções o PERS foi desenvolvido para efetuar transmissão de dados coletados por sensores e enviar para a rede utilizando a topologia ponto-a-ponto.

Importante salientar que as instruções do processador foram definidas após um estudo do Maté (LEVIS e CULLER 2002) que é uma máquina virtual para Redes de Sensores, analisando seus programas de envio e recebimento de mensagens e fazendo uma adaptação para a topologia Ponto-a-Ponto.

Foram necessárias instruções para montagem do cabeçalho de dados utilizando endereços de origem e destino dos nós que compõem os sensores da rede, para controlar a quantidade de saltos efetuados pelos dados entre os nós sensores da rede, para desmembrar a mensagem de dados, retirando o endereço de origem e destino e também a informação coletada, bem como instruções para efetuar testes caso os sensores não tenham coletado informações de um determinado local ou ambiente. Por fim, instruções para enviar a mensagem de dados para a rede. Todos esses estudos levaram a um conjunto de instruções do PERS que são mostrados na tabela 5.1.

As instruções Lógicas e Aritméticas (INC e DEC) são utilizadas para incrementar e decrementar o *hopcount* que é o controle necessário para saber quantas vezes um determinado pacote de dados foi transmitido de sensor para sensor.

A instrução LDID (*Load* ID) carrega o endereço do nó origem no registrador ID, necessário para compor a mensagem completa que será enviada pela rede. O endereço é encaminhado junto com a instrução.

As instruções de Desvio (JMP, JD, JPES e JPEW) são utilizadas para efetuar saltos no programa caso seja necessário. A instrução JMP faz um desvio incondicional, a instrução JD faz um desvio caso o nó seja o endereço destino do pacote e por fim, as instruções JPES (*Jump* de entrada dos sensores) e JPEW (*Jump* de entrada da Rede) fazem um desvio caso não exista informação disponível no sensor ou na rede respectivamente.

As instruções especiais/rede são utilizadas para o roteamento, recepção e transmissão de dados entre os registradores e portas de entrada e entrada e saída.

Tabela 5.1 Instruções implementadas do PERS

Lógicas e Aritméticas		
1	INC	Incrementa o HopCount
2	DEC	Decrementa o HopCount
Movimentação		
3	LDID	Atribui um valor para o registrador ID
Desvio		
4	JMP	Desvio incondicional
5	JD	Desvia se o nó for o destino
6	JPES	Desvia se não existe pacote de dados no sensor
7	JPEW	Desvia se não existe pacote de dados na Rede
Especiais/Rede		
8	SENSE	Atribui ao registrador I o conteúdo da porta PIO
9	WLAN	Atribui ao registrador IO o conteúdo da porta PIO
10	SET_ID	Seta no registrador MSG o conteúdo do registrador ID
11	SET_DEST	Seta no registrador MSG o endereço destino do pacote
12	SET_HOP	Seta no registrador MSG o hopcount inicial do pacote
13	SET_SENSE	Seta no registrador MSG o conteúdo do registrador I
14	SET_ORI	Seta no registrador ORI se conteúdo do registrador ID for maior
15	GET_DEST	Atribui ao registrador DEST o conteúdo do nó destino do registrador IO
16	GET_HOP	Atribui ao registrador HOP o conteúdo de saltos do nó do registrador IO
17	GET_ORI	Atribui ao registrador ORI o conteúdo do nó origem do registrador IO
18	CHK_DEST	Checa se o nó é o destino do pacote
19	PUT	Envia o pacote de dados do registrador MSG para a porta PIO
20	PUT_LAN	Envia a mensagem do registrador MSG para a porta PIO

A instrução SENSE atribui ao registrador I (nome atribuído por ser um registrador que vai armazenar informação apenas de entrada, ou seja, *Input*) os dados que estão na porta PI que foram coletados pelos sensores.

A instrução WLAN atribui ao registrador IO (nome atribuído por ser um registrador que vai armazenar informação de entrada e saída, ou seja, *Input* e *Output*) os dados que estão na porta PIO e que foi transmitida pela rede.

As instruções que começam com SET montam a mensagem de dados que será transmita na rede pela instrução PUT. As instruções que começam com GET desmembram as informações da mensagem de dados que foi transmitida na rede para saber se o nó é o destino

e também incrementar o *hopcount* para depois ser remontada e enviada pela instrução PUT_LAN.

Importante salientar que nesta versão foi definido que o endereço destino da mensagem de dados é sempre o endereço “0000”, pois a finalidade de todos os nós é enviar a mensagem de dados para a estação base que vai analisar as informações coletadas.

5.2.1 Programa Teste

Foi utilizada a topologia Ponto-a-Ponto para analisar o desempenho do programa representado na tabela 5.2.

Tabela 5.2 *Assembly* de uma comunicação

	PERS	Ciclos
1	SENSE	2
2	JPES, 08	2
3	LDID, 00	2
4	SET_ID	2
5	SET_DEST	2
6	SET_HOP	2
7	SET_SENSE	2
8	PUT	2
9	WLAN	2
10	JPEW, 00	2
11	LDID, 00	2
12	GET_DEST	2
13	CHK_DEST	2
14	JD, 20	2
15	GET_ORI	2
16	SET_ORI	2
17	GET_HOP	2
18	INC	2
19	PUT_LAN	2
20	JMP, 00	2
21	HALT	2
	Total	42

O objetivo desta topologia é condicionar a porta de saída de um nó na porta de entrada de outro nó de modo seqüencial, exemplificando: o nó 16 só pode enviar para o nó 15

que por sua vez recebe o pacote e envia para o nó 14. Cada nó desta topologia representa um processador PERS.

Como pode-se observar na tabela 5.2, o programa descreve um laço que fica capturando dados em sensores; caso não exista dados disponíveis na porta que é ligado aos sensores é efetuado um salto para buscar dados na porta que é ligada à rede, pois pode ser que outro sensor tenha transmitido dados pela rede. Interessante observar que após a prototipação em FPGAs desse processador, todas as instruções requerem dois ciclos para serem executadas, isso porque a Unidade de Controle trabalha com máquina de estado finita e a versão inicial da nossa implementação aqui apresentada possui apenas os estados de busca e execução.

5.2.2 Simulações na ferramenta Xilinx Foundation Series

Após a definição do conjunto de instruções e do programa teste que simula uma topologia ponto-a-ponto, foi implementado o processador na ferramenta Xilinx Foundation Series (Xilinx, 1998). A seguir são mostradas algumas simulações das instruções implementadas. Por exemplo, a figura 5.3 mostra a simulação da instrução LDID, que carrega no registrador ID a identificação do nó sensor na rede.

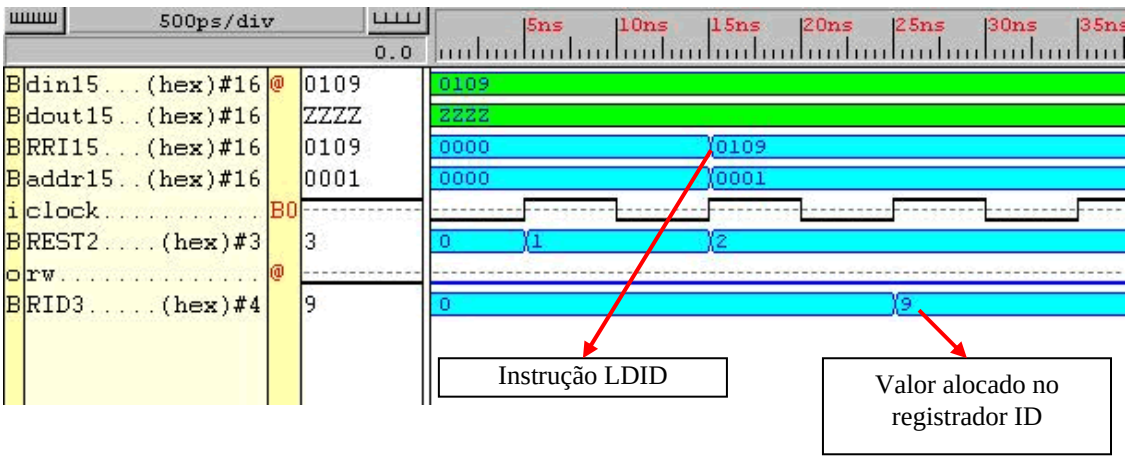


Figura 5.3. Simulação da instrução LDID

No exemplo é possível observar que a instrução contém o endereço do nó na qual é atribuído ao registrador ID após o ciclo de execução.

Na figura 5.4 é possível observar a simulação das instruções INC e DEC, que incrementa e decrementa o valor do registrador HOP. O incremento do registrador HOP acontece a cada transmissão de mensagem de dados, ou seja, a cada mudança de nó sensor.

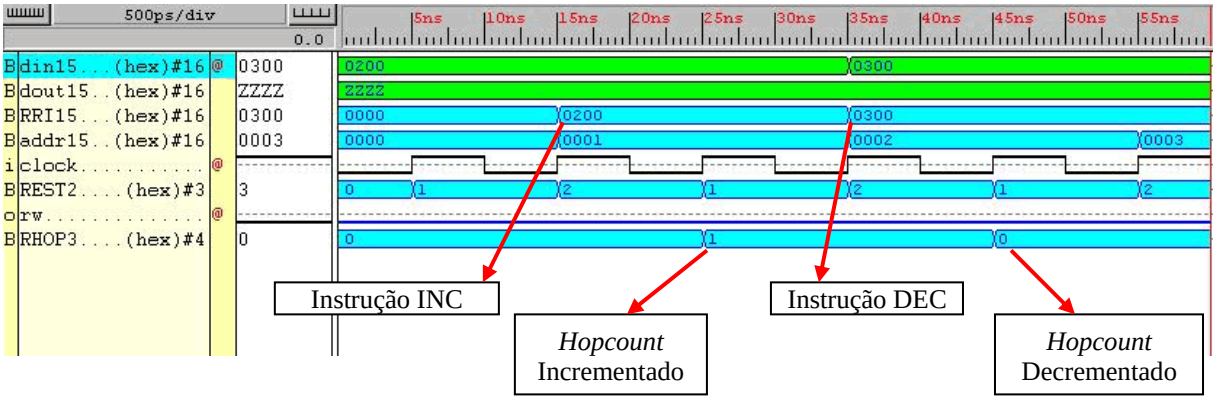


Figura 5.4. Simulação das instruções INC e DEC

Na figura 5.5 se observa o resultado da simulação da instrução JPES, que faz um salto no programa descrito na tabela 5.2, ou seja, caso a porta de entrada (PI) esteja vazia, é feito um salto para a instrução WLAN (que vai verificar se possui dados na porta de entrada e saída), pois o sensor não coletou dados. Este salto é feito no endereço de memória que contém o programa *Assembly*, pode-se notar o salto do endereço “0001” para o endereço “0015”.

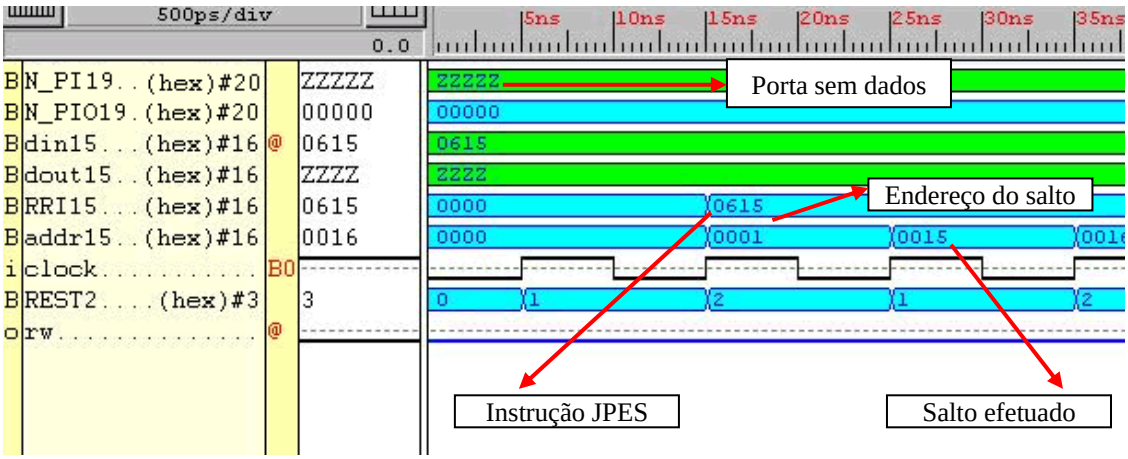


Figura 5.5. Simulação da instrução JPES

Na figura 5.6 se observa, através de simulação, o envio de um pacote de dados que foi coletado por um sensor. A primeira instrução simulada é a SENSE que atribui o valor da porta de entrada (PI) no registrador I, ou seja, armazena em um registrador específico os dados coletados pelo sensor. Depois se executa a instrução LDID que armazena no registrador ID o endereço do nó sensor origem, o endereço do ID vem acompanhado na instrução.

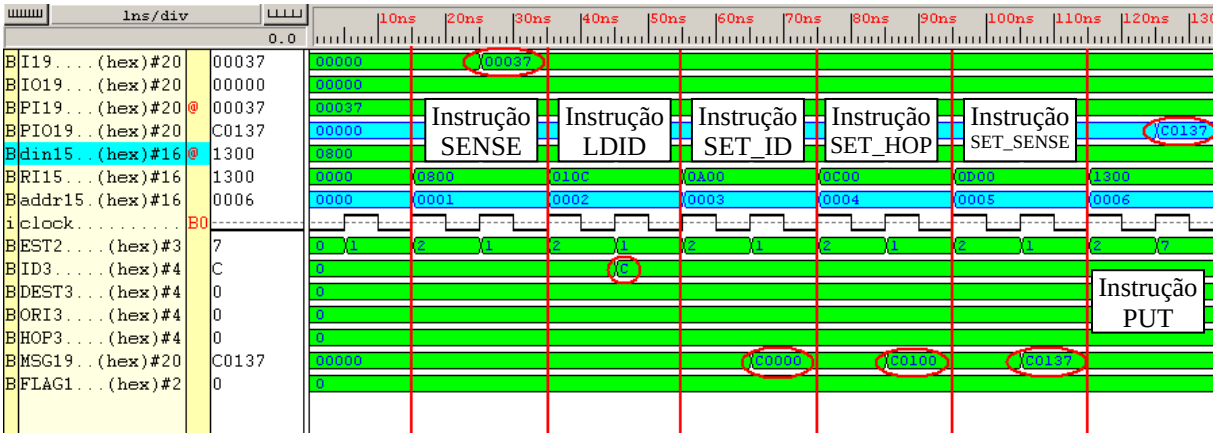


Figura 5.6. Simulação de envio de Mensagem de Dados

A partir da terceira instrução que é a SET_ID, se dá início à montagem da mensagem de dados (20 bits, conforme demonstrado na figura 5.2), pega-se o valor do registrador ID e coloca-se nos primeiros 4 bits do registrador MSG (que equivale a mensagem de dados). Nesta simulação não se colocou a instrução SET_DEST que informa o destino do pacote, pois conforme já descrito foi definido que será sempre para o nó zero, como o registrador MSG é inicializado com zero, não seria possível visualizar com êxito a simulação da instrução.

A instrução SET_HOP informa a quantidade de saltos que a mensagem de dados vai ter durante a transmissão, considerando que a primeira transmissão então vai começar com valor um. Depois a instrução SET_SENSE “pega” o conteúdo do registrador I e coloca-o na mensagem, terminando de montá-la. Com isso a mensagem fica completa, composta pelo endereço do nó origem, endereço do nó destino, *hopcount* da mensagem e os dados da

mensagem. Finalizando, a instrução PUT envia a mensagem de dados para a porta de entrada e saída (PIO), que é a porta da rede.

A seguir, apresenta-se um detalhamento das instruções que foram simuladas nas figuras 5.2-5.6, a saber: LDID, INC, DEC, JPES, SENSE, SED_ID, SET_HOP, SET_SENSE e PUT, e implementadas em VHDL:

```
when ldid =>    ID <= RI(3 downto 0);
                EST <= busca;
when inc =>     HOP <= HOP +1;
                EST <= busca;
when dec =>     HOP <= HOP -1;
                EST <= busca;
when jpes =>    if I = "0000000000000000000" then
                        PC(7 downto 0) <= RI(7 downto 0);
                        FLAG<="00";
                        EST <= busca;
                end if;
when sense => I <= PI;
                EST <= busca;
when set_id =>          MSG(19 downto 16) <= ID;
                EST <= busca;
when set_hop => MSG(11 downto 8) <= count;
                EST <= busca;
when set_sense => MSG(7 downto 0) <= I(7 downto 0);
                EST <= busca;
when put =>     PIO <= MSG;
                EST <= inicializa;
```

5.2.3 Estatísticas da prototipação em FPGAs

Após a implementação, síntese e teste do processador PERS foram analisados os resultados obtidos através da ferramenta Xilinx Foundation Series (Xilinx, 1998), que oferece informações quanto ao tempo de propagação do circuito e freqüência máxima do processador, assim como o espaço ocupado pelo circuito no FPGA. A prototipação foi realizada usando o FPGA Spartan 2 – XC2S200E da Xilinx, e os resultados estão contidos na tabela 5.3.

Tabela 5.3 Estatísticas espaciais do processador PERS1

Spartan 2 - XC2S200E	Total	Ocupado	%
CLBs	2352	175	7
Flip Flops	4704	121	2
Luts de 4 entradas	4704	321	6

É bom lembrar que a utilização dos recursos do FPGA são dados através de: (i) **CLB** *(Configurable Logic Bloc*k): Matriz de blocos lógicos configuráveis, unidade lógica de uma FPGA; (ii) **Flip Flop**: Circuito digital básico que armazena um bit de informação, sua saída só muda de estado durante a transição do sinal do *cloc*k; (iii) **Luts:** Muitos FPGAs modernos são modelados como tabelas de busca *(lookup table* - LUTs) programáveis. A LUT constitui a configuração de cada elemento da matriz.

Analisando os resultados mostrados na tabela 5.3 é possível verificar que a versão inicial do PERS utiliza uma porcentagem muito pequena do FPGA Spartan 2, o que possibilita implementar mais instruções específicas de roteamento para abranger outras topologias e também para inserir instruções de modo a incluir criptografia na transmissão dos dados.

As medidas temporais do processador PERS, quanto ao tempo de propagação no circuito é de 11,801 (ns) e a Freqüência Máxima do Processador é de 84,739 Mhz, medidas essas obtidas com a ferramenta Xilinx ISE.

5.3 PERS 2 – Arquitetura

Na segunda versão do PERS foram adicionados novos registradores e também uma nova porta de saída para efetuar alguma ação de acordo com a variável medida (ex. temperatura) que foi coletada pelo sensor. Na figura 5.7 pode-se observar a inclusão dos registradores A e B que utiliza-se para efetuar as novas operações Lógicas e Aritméticas do PERS 2, foram adicionados também os registradores BROAD, DADOS e CTRL (ver funções

na tabela 5.4) e por fim, a inclusão da porta de saída PO de 4 bits (nome atribuído por ser uma porta apenas de saída, ou seja, *Output*). Definiu-se este tamanho porque pode-se representar com 4 bits até 16 ações que a estação base pode executar de acordo com a variável (ex. temperatura) coletada pelos sensores.

Importante salientar que os nomes atribuídos aos registradores refletem ou tenta-se refletir a uma associação máxima à função que o registrador vai tomar. Atribuiu-se o nome BROAD para o registrador porque ele vai armazenar o endereço de broadcast da rede, mesma coisa para os registradores DADOS e CTRL, sendo que o primeiro armazena-se os dados que são coletados pela porta PIO (Porta de entrada e saída da rede ou *input* e *output*) e no segundo armazena-se a ação de controle (por isso o nome CTRL).

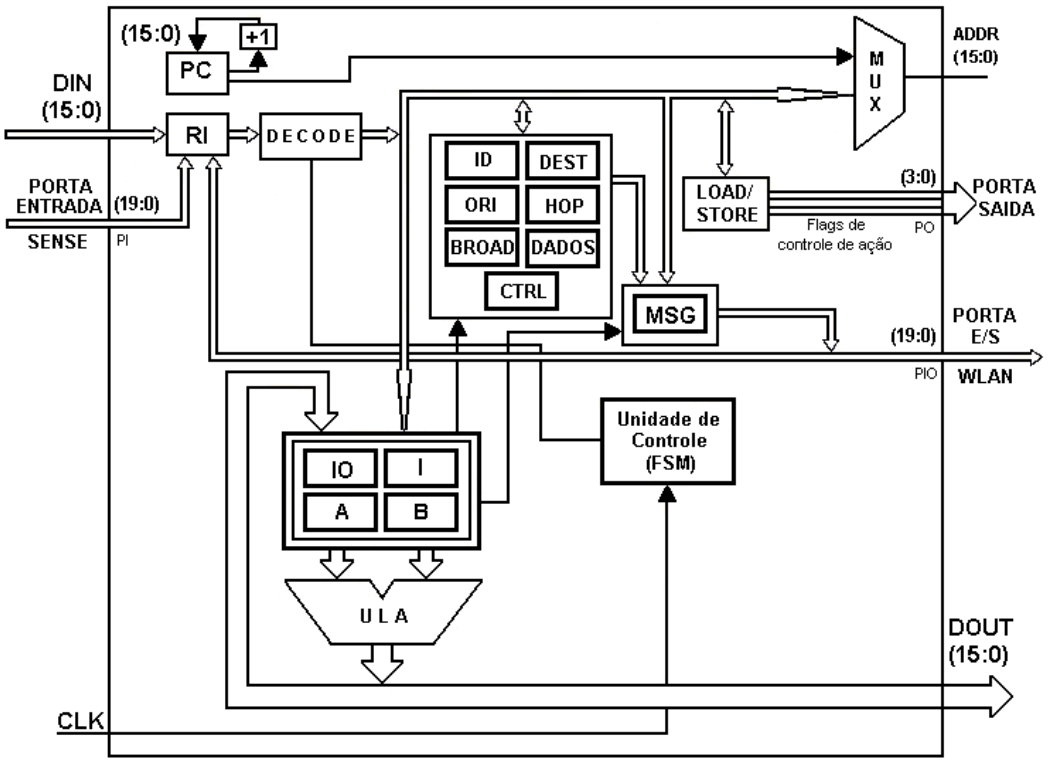


Figura 5.7. Arquitetura detalhada da 2ª versão do PERS

5.3.1 PERS 2 – Conjunto de instruções (ISA) – 2ª versão

A segunda versão do PERS inclui flags de controle de aplicação de 4 bits para efetuar alguma ação caso o nó seja o destino da mensagem de dados e também se adicionam

mais instruções lógicas e aritméticas, de movimentação e de desvio. Com essas inclusões, o PERS tem alterações na sua arquitetura inicial. Essas mudanças arquiteturais sugerem alterações no programa teste, novas simulações e também novas estatísticas de desempenho da prototipação em FPGAs.

Nesta segunda versão foram incluídas 10 novas instruções: 6 instruções Lógicas e Aritméticas (AND, OR, XOR, ADD, SUB e INV), 3 instruções de Movimentação (LD, STR e MOV) e uma instrução Especial de Rede (CHK_DATA). Assim, o novo PERS, PERS2 possui 30 instruções, ver tabela 5.4.

Tabela 5.4 Instruções implementadas na 2ª versão do PERS

Lógicas e Aritméticas		
1	AND	Faz a operação AND entre dois registradores
2	OR	Faz a operação OR entre dois registradores
3	XOR	Faz a operação XOR entre dois registradores
4	ADD	Soma dois registradores
5	SUB	Subtrai dois registradores
6	INV	Nega a constante destino (0000) e coloca no registrador BROAD
7	INC	Incrementa o HopCount
8	DEC	Decrementa o HopCount
Movimentação		
9	LDID	Atribui um valor para o registrador ID
10	LD	Atribui ao registrador DADOS o conteúdo (dados) da porta PIO
11	STR	Envia uma ação de controle para a porta PO
12	MOV	Atribui ao registrador A o conteúdo do registrador B
Desvio		
13	JMP	Desvio incondicional
14	JD	Desvia se o nó for o destino
15	JPES	Desvia se não existe pacote de dados no sensor
16	JPEW	Desvia se não existe pacote de dados na Rede
Especiais/Rede		
17	SENSE	Atribui ao registrador I o conteúdo da porta PI
18	WLAN	Atribui ao registrador IO o conteúdo da porta PIO
19	SET_ID	Seta no registrador MSG o conteúdo do registrador ID
20	SET_DEST	Seta no registrador MSG o destino do pacote
21	SET_HOP	Seta no registrador MSG o hopcount inicial do pacote
22	SET_SENSE	Seta no registrador MSG o conteúdo do registrador I
23	SET_ORI	Seta no registrador ORI o se conteúdo do registrador ID for maior
24	GET_DEST	Atribui ao registrador DEST o conteúdo do nó destino do registrador IO
25	GET_HOP	Atribui ao registrador HOP o conteúdo de saltos do nó do registrador IO
26	GET_ORI	Atribui ao registrador ORI o conteúdo do nó origem do registrador IO
27	CHK_DEST	Checa se o nó é o destino do pacote
28	CHK_DATA	Atribui ao registrador CTRL uma ação de controle
29	PUT	Envia o pacote de dados do registrador MSG para a porta PIO
30	PUT_LAN	Envia a mensagem do registrador MSG para a porta PIO

As novas instruções Lógicas e Aritméticas são utilizadas para fazer operações básicas com registradores no processador e a instrução INV será utilizada para fazer *broadcast*, pois vai inverter o endereço destino para enviar para todos os nós da rede.

A instrução LD atribui direto ao registrador DADOS o conteúdo da porta de entrada e saída PIO (porta da rede onde os dados são coletados), a instrução STR atribui direto para a porta de saída PO o conteúdo do registrador CTRL e a instrução MOV atribui o valor do registrador B para o registrador A.

A seguir, se mostra o exemplo de medida de temperatura que a instrução CHK_DATA vai analisar de acordo com a variável coletada pelo sensor e irá tomar alguma ação de acordo com as regras estabelecidas (ver tabela 5.5 e 5.7).

Tabela 5.5 Tabela A de controle de ação da 2ª versão do PERS

Ação	Flag	
1	0000	LED0 (Luz Verde)
2	0001	LED1 (Luz Amarela)
3	0010	LED2 (Luz Vermelha)
4	0011	Ar Condicionado
5	0100	Alarme
6	0101	Alarme de Ação

Tabela 5.6 Tabela B de controle de ação da 2ª versão do PERS

Temperatura		Flag	
1	<= 20	0000	LED0 (Luz Verde)
2	> 20 ou <= 25	0001	LED1 (Luz Amarela)
3	> 25 ou <= 30	0010	LED2 (Luz Vermelha)
4	> 30 ou <= 35	0011	Ar Condicionado
5	> 35 ou <= 45	0100	Alarme
6	> 45	0101	Alarme de Ação

A tabela 5.5 mostra diferentes ações que podem ser tomadas de acordo com a temperatura analisada. O programa teste descrito na Tabela 5.7 mostra que o usuário é quem define o valor da temperatura de cada ação a ser tomada. Neste exemplo, a primeira ação é acender a luz verde, a segunda acender a luz amarela, a terceira acender a luz vermelha, a

quarta ligar o ar condicionado, a quinta ligar o alarme de emergência e por fim, a última ação enviar uma ação para ligar o alarme de ação (por exemplo, ligação dos dutos de água).

A tabela 5.6 fixa no código do programa os valores para efetuar alguma ação de acordo com a temperatura analisada, ou seja, não existe a interação do usuário. Até 20º graus acende a luz o verde, a amarela até 25º graus e a vermelha até 30º graus. Depois tomam-se ações, se a temperatura passar de 30º graus liga o ar condicionado, caso a temperatura passe de 35º graus liga o alarme de emergência e por fim, caso a temperatura passe dos 45º graus liga o alarme de ação (por exemplo, ligação dos dutos de água). Estas ações podem ser programadas pelo usuário.

5.3.2 Programa Teste

Para a segunda versão, foram adicionadas no programa descrito na tabela 5.2 as linhas contidas nas tabelas 5.7 e 5.8. O primeiro programa deixa a cargo do usuário a definição das ações de acordo com a variável desejada (ex. temperatura coletada). Já o segundo programa os valores foram definidos no programa de acordo com a tabela 5.6.

Tabela 5.7 Complemento A do A*ssembly* ponto-a-ponto

…	……..	…….
21	LD	2
22	CHK_DATA, 20	2
23	CHK_DATA, 25	2
24	CHK_DATA, 30	2
25	CHK_DATA, 35	2
26	CHK_DATA, 45	2
27	STR	2
28	JMP, 00	2

Tabela 5.8 Complemento B do A*ssembly* ponto-a-ponto

…	……..	…….
21	LD	2
22	CHK_DATA	2
23	STR	2
24	JMP, 00	2

5.3.3 Simulações usando a ferramenta Xilinx

Foram feitas novas simulações com a inclusão das novas instruções. A figura 5.8 mostra a simulação do programa descrito na tabela 5.8 com a checagem da variável e o envio da ação de controle pela porta de saída PO.

Na figura 5.8 se verifica a simulação do envio de uma ação de controle após analisar a variável (ex. temperatura) coletada por um sensor. A primeira é a instrução LD que atribui o valor da porta de entrada (PIO) no registrador DADOS, depois a instrução CHK_DATA analisa a variável (ex. temperatura) coletada e atribui no registrador CTRL um valor associado que consta na tabela 5.6, finalizando, a instrução STR envia o conteúdo do registrador CTRL para a porta de saída (PO).

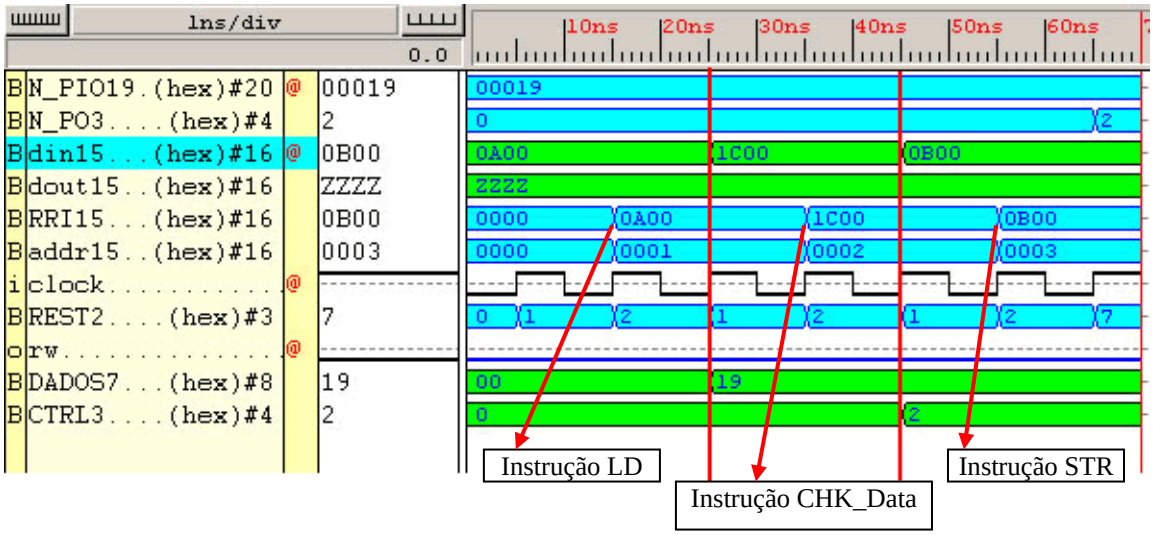


Figura 5.8. Envio de uma ação de controle

A seguir, apresenta-se o detalhamento das instruções LD, CHK_DATA e STR implementadas em VHDL:

```
when ld =>       DADOS <= PIO(7 downto 0);
                  EST <= busca;
when chk_data => if DADOS < "00010101" then
                 CTRL <= "0000";
                 elsif DADOS < "00011000" then
                        CTRL <= "0001";
```

```
                elsif DADOS < "00011010" then
                        CTRL <= "0010";
                elsif DADOS < "00011111" then
                        CTRL <= "0011";
                elsif DADOS < "00100100" then
                        CTRL <= "0100";
                elsif DADOS < "00101001" then
                        CTRL <= "0101";
                else
                        EST <= inicializa;
                end if;
                EST <= busca;
when str =>     PO <= CTRL;
                EST <= inicializa;
```

5.3.4 Estatísticas da prototipação em FPGAs

Na tabela 5.9 são apresentados os testes da 2ª versão do PERS, que deram informações quanto ao tempo de propagação do circuito e freqüência máxima do processador, assim como o espaço ocupado pelo circuito no FPGA Spartan 2.

Tabela 5.9 Estatísticas Espaciais da 2ª versão do PERS

Spartan 2 – XC2S200E	Total	Ocupado	%
CLBs	2352	165	7
Flip Flops	4704	135	2
Luts de 4 entradas	4704	305	6

As medidas temporais do processador PERS, quanto ao tempo de propagação no circuito é de 12,397 (ns) e que equivale a uma Freqüência Máxima do Processador de 80,665 Mhz, medidas essas obtidas com a ferramenta Xilinx ISE.

Analisando os resultados pode-se observar que não houve mudanças espaciais significativas, mas sim temporais em relação à primeira versão do processador PERS.

5.4 Inserção de funções de Criptografia

Existem diversos algoritmos de criptografia que podem ser utilizados, porém nesta versão inclui-se os algoritmos de criptografia DES (*Data Encryption Standard*) e AES (*Advanced Encryption Standard*). Com isso, será possível adicionar maior segurança na transmissão dos dados através da Rede dos Sensores sem Fio. Assim, como em todas as versões anteriores, essa inclusão ocasionará mudanças na arquitetura do PERS, e também serão necessários novos programas testes e novas simulações. Entretanto, é importante frisar que o ponto principal desta versão é o impacto desta inclusão no desempenho do PERS.

Tabela 5.10 Conjunto de Instruções na versão do PERS com criptografia

Lógicas e Aritméticas		
1	AND	Faz a operação AND entre dois registradores
2	OR	Faz a operação OR entre dois registradores
3	XOR	Faz a operação XOR entre dois registradores
4	ADD	Soma dois registradores
5	SUB	Subtrai dois registradores
6	INV	Nega a constante destino (0000) e coloca no registrador BROAD
7	INC	Incrementa o HopCount
8	DEC	Decrementa o HopCount
Movimentação		
9	LDID	Atribui um valor para o registrador ID
10	LD	Atribui ao registrador DADOS o conteúdo (dados) da porta PIO
11	STR	Envia uma ação de controle para a porta PO
12	MOV	Atribui ao registrador A o conteúdo do registrador B
Desvio		
13	JMP	Desvio incondicional
14	JD	Desvia se o nó for o destino
15	JPES	Desvia se não existe pacote de dados no sensor
16	JPEW	Desvia se não existe pacote de dados na Rede
Especiais/Rede		
17	SENSE	Atribui ao registrador I o conteúdo da porta PI
18	WLAN	Atribui ao registrador IO o conteúdo da porta PIO
19	SET_ID	Seta no registrador MSG o conteúdo do registrador ID
20	SET_DEST	Seta no registrador MSG o destino do pacote
21	SET_HOP	Seta no registrador MSG o hopcount inicial do pacote
22	SET_SENSE	Seta no registrador MSG o conteúdo do registrador I
23	SET_ORI	Seta no registrador ORI o se conteúdo do registrador ID for maior
24	GET_DEST	Atribui ao registrador DEST o conteúdo do nó destino do registrador IO
25	GET_HOP	Atribui ao registrador HOP o conteúdo de saltos do nó do registrador IO
26	GET_ORI	Atribui ao registrador ORI o conteúdo do nó origem do registrador IO
27	CHK_DEST	Checa se o nó é o destino do pacote
28	CHK_DATA	Atribui ao registrador CTRL uma ação de controle
29	PUT	Envia o pacote de dados do registrador MSG para a porta PIO
30	PUT_LAN	Envia a mensagem do registrador MSG para a porta PIO
31	CRIPTA	Cifra o conteúdo da porta PI e atribui no registrador MSG
32	DECRIPTA	Decifra o conteúdo da porta PIO e atribui no registrador DADOS

Com a inclusão de criptografia, foram incluídas duas novas instruções Especiais de Rede (CRIPTA e DECRIPTA), totalizando assim 32 instruções no PERS. A tabela 5.10 mostra as novas instruções inclusas que serão utilizadas pelos dois algoritmos de criptografia.

A instrução CRIPTA será utilizada para cifragem dos dados que são coletados pelos sensores de rede. Após a cifragem ele é enviado para o próximo nó até chegar ao destino, onde o dado será decifrado.

A instrução DECRIPTA será utilizada para decifragem dos dados que são coletados na rede, caso o nó sensor seja o destino. Após a decifragem será tomada alguma ação de acordo com o dado coletado.

5.4.1 Algoritmo DES

O algoritmo DES é composto de operações simples, como permutações, substituições, XOR e deslocamentos. O DES criptografa informações por meio do processo de cifra de bloco com tamanho de 64 bits e retorna blocos de texto cifrado do mesmo tamanho usando uma chave de 56 bits.

O processo principal do algoritmo é executado 16 vezes (iterações) e em cada iteração se utiliza uma subchave diferente derivada da chave original.

Para cifrar um determinado bloco de texto (64 bits), é necessário utilizar as subchaves na ordem crescente, isto é, na iteração 1 do processamento principal se utiliza a subchave 1, na iteração 2, a subchave 2, na 3, a subchave 3 e assim por diante. Para decifrar um bloco de texto, as subchaves são aplicadas na ordem inversa ou decrescente. Neste contexto: para decifrar a iteração 1, se utiliza a subchave 16, na iteração 2, se utiliza a subchave 15, na iteração 3, a subchave 14 e assim por diante.

A figura 5.9 mostra a representação do processamento principal do algoritmo DES, onde é possível verificar que o processamento principal é dividido em três etapas executadas em seqüência. A primeira etapa é a permutação inicial (PI), a segunda corresponde às iterações e a terceira e última etapa corresponde à permutação final (PF). Maiores detalhes das operações do algoritmo e da implementação podem ser obtidas em MORENO et al. (2005).

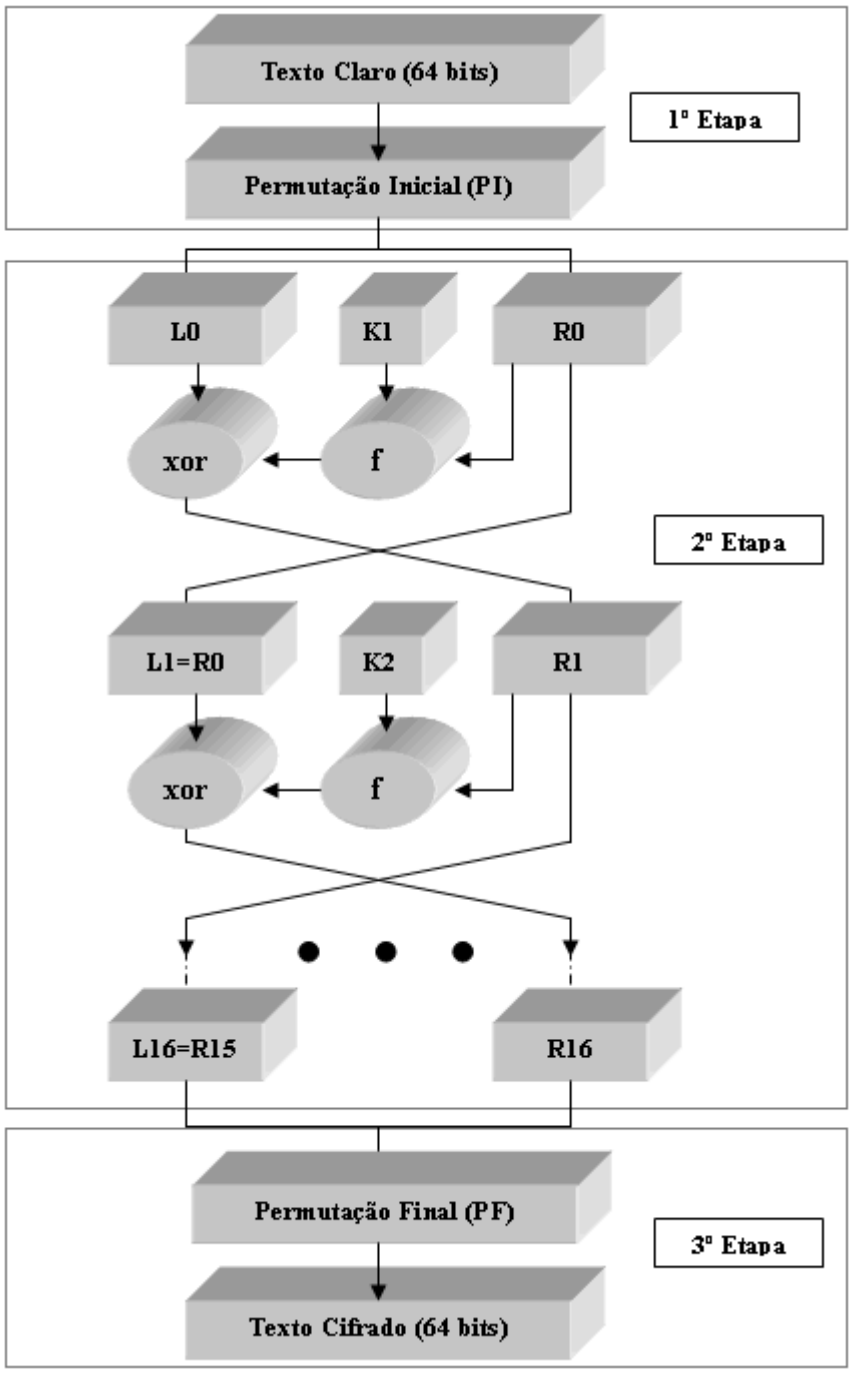


Figura 5.9. Operações do Algoritmo DES (MORENO et al., 2005)

5.4.2 PERS 3 - Arquitetura com DES

Nesta versão do PERS com criptografia, foi necessário adicionar novos registradores específicos para criptografia, uma vez que o algoritmo DES necessita de registradores para armazenar o bloco de dados, a cifra que é o dado cifrado, as iterações (16 no total) e a chave criptográfica que muda a cada iteração. Na figura 5.10 pode-se observar a inclusão dos registradores BLOCO, CIFRA, Rodada e CHAVE, que são utilizados pelo algoritmo DES.

Além da inclusão dos registradores, foram necessárias mudanças na estrutura de alguns registradores e nas portas do processador PERS, pois o algoritmo DES utiliza 64 bits.

O registrador MSG era composto de 4 bits para o endereço destino, 4 bits para o endereço origem, 4 bits para o *hopcount* e 8 bits para os dados, totalizando 20 bits. Considerando a possibilidade de criptografia usando o DES , passou a possuir 64 bits para os dados devido à necessidade de cifragem e decifragem, totalizando assim 76 bits.

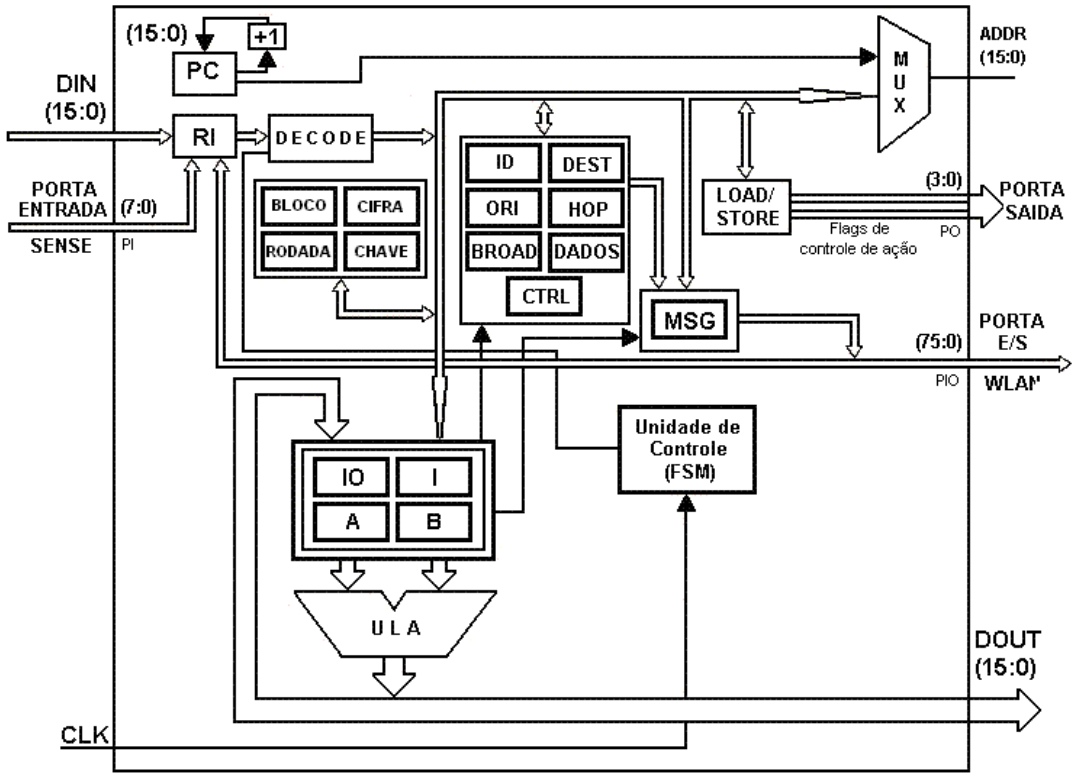


Figura 5.10. Arquitetura detalhada do PERS com opções de criptografia - DES

A cifragem dos dados ocorrerá no momento que o sensor coletar a informação, ou seja, o nó origem vai coletar e cifrar os dados. Os dados permanecerão cifrados até que chegue à estação base onde será decifrado para análise da variável coletada. Com isso, a porta (PIO) e os registradores (IO e MSG) que utilizam a rede precisaram se adaptar para receber a informação cifrada, com isso foram aumentados de 20 bits para 76 bits.

5.4.3 Programa Teste

Para a terceira versão, foram adicionadas linhas que estão contidas na tabela 5.11. Pode ser notada a inclusão de duas linhas no programa que se refere às instruções CRIPTA e DECRIPTA (ver linha 2 e linha 21 no programa de instrução da tabela 5.11).

Tabela 5.11 *Assembly* ponto-a-ponto do processador PERS com Criptografia

	PERS	Ciclos
1	SENSE	2
2	CRIPTA	19
3	JPES, 08	2
4	LDID, 00	2
5	SET_ID	2
6	SET_DEST	2
7	SET_HOP	2
8	PUT	2
9	WLAN	2
10	JPEW, 00	2
11	LDID, 00	2
12	GET_DEST	2
13	CHK_DEST	2
14	JD, 20	2
15	GET_ORI	2
16	SET_ORI	2
17	GET_HOP	2
18	INC_HOP	2
19	PUT_LAN	2
20	JMP, 00	2
21	DECRIPTA	19
22	CHK_DATA	2
23	STR	2
24	JMP, 00	2
	Total	82

5.4.4 Simulações usando a ferramenta Xilinx

Na figura 5.11 se observa a simulação da cifragem de um dado após ser coletado pelo sensor de um nó da rede. O dado cifrado é armazenado no registrador MSG que depois será preparado com o cabeçalho (endereço do nó origem, endereço do nó destino e *hopcount*) para ser enviado ao próximo nó da rede. Já na figura 5.12 se observa a continuação da simulação da figura 5.11 onde acontece a decrifragem do dado. O bloco recebe o dado cifrado do registrador MSG, depois se executa as 16 iterações necessárias para no final, o registrador DADOS receber a informação decifrada. Com o dado decifrado será tomada alguma ação de controle de acordo com as regras contidas na tabela 5.5.

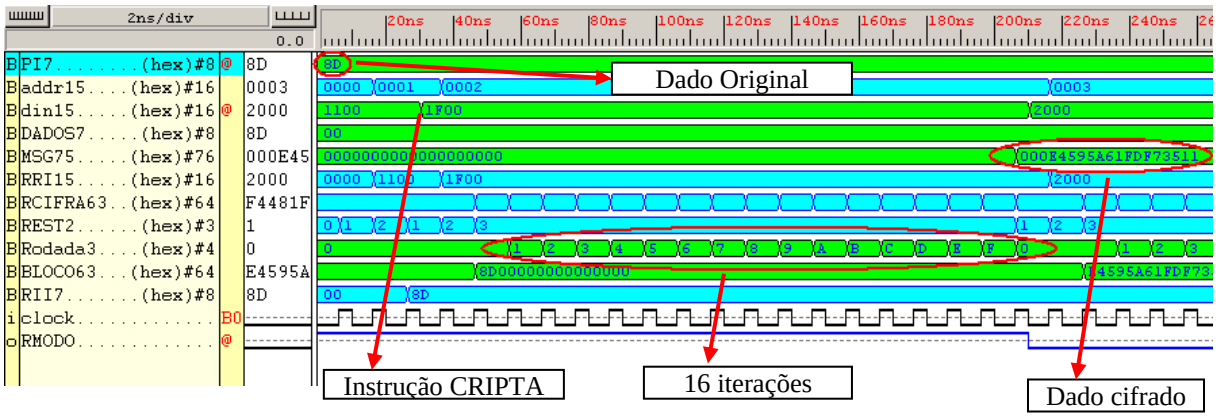


Figura 5.11. Cifragem de um dado no PERS 3

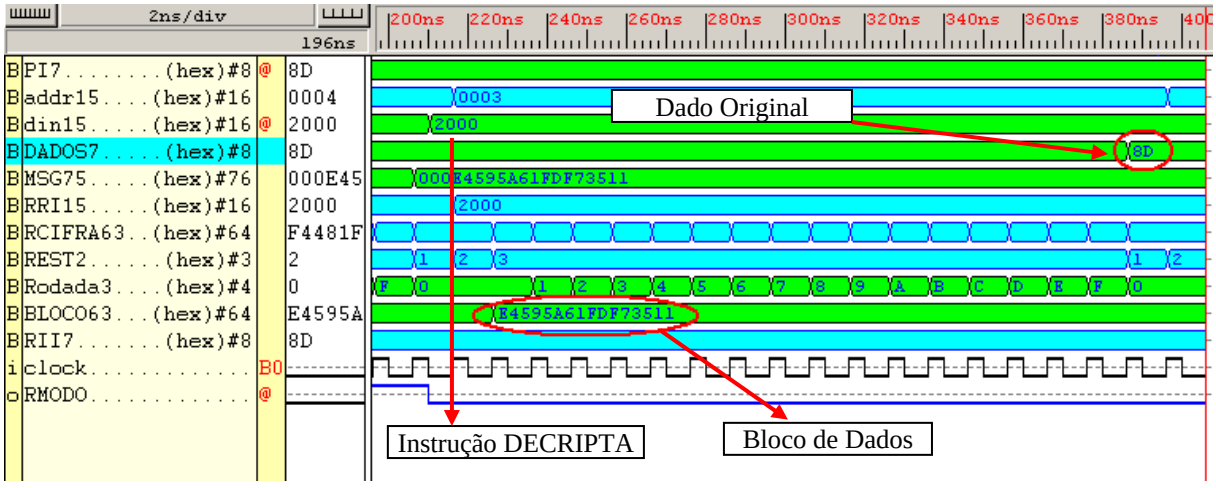


Figura 5.12. Decifragem de um dado no PERS 3

Pode-se observar que o dado original “8D” contido na figura 5.11, aparece novamente ao final da figura 5.12 após passar pelas instruções cripta e decripta comprovando o sucesso na cifragem e decifragem dos dados. As simulações mostradas nas figuras 5.11 e 5.12 correspondem às instruções CRIPTA e DECRIPTA implementadas em VHDL, que segue a seguinte estrutura:

```
when cripta => CHAVE <= "00000000000000000000000000000000000000000000000000000000";
               MODO  <='1';
               BLOCO <= I&PAD;
               EST <= CRIPTO;
when decripta => MODO  <='0';
               BLOCO <= IO(63 downto 0);
               EST <= CRIPTO;
```

Pode-se notar que foi incluído no processador um novo estado, chamado CRIPTO. Este estado descrito a seguir foi necessário para efetuar as 16 iterações necessárias no algoritmo do DES e depois cifrar ou decifrar o dado de acordo com o valor atribuído para o registrador MODO, sendo valor ‘1’ para cifrar e ‘0’ para decifrar (PEREIRA, 2004):

```
when CRIPTO =>      Rodada<=Rodada+1;
                    if Rodada="1111" then
                        Rodada<="0000";
                            if MODO='1' then
                                MSG(63 downto 0) <= CIFRA;
                            elsif MODO='0' then
                                DADOS <= IO(63 downto 56);
                            end if;
                    EST <= busca;
                    end if;
```

Como o algoritmo DES já estava implementado em VHDL (PEREIRA, 2004), foi adicionado como um componente do processador PERS, usando o código que o declara como um componente:

```
component des
```

```
                port (clk      : in  std_logic;
                modo     : std_logic;
                iteracao : std_logic_vector(3 downto 0);
                K        : std_logic_vector(55 downto 0);
                din      : std_logic_vector(63 downto 0);
                saida    : out std_logic_vector (63 downto 0));
end component;
```

Para finalizar, é preciso chamar o componente declarado no código do PERS, usando a seguinte declaração:

```
U1:DES port map (Clock, MODO, Rodada, CHAVE, BLOCO, CIFRA);
```

5.4.5 Estatísticas de prototipação em FPGAs

Na tabela 5.12 apresentam-se os testes da terceira versão do PERS, que deram informações quanto ao tempo de propagação do circuito, freqüência máxima do processador, assim como o espaço ocupado pelo circuito no FPGA.

Tabela 5.12 Estatísticas Espaciais com algoritmo DES

Spartan 2 – XC2S200E	Total	Ocupado	%
CLBs	2352	757	32
Flip Flops	4704	88	1
Luts de 4 entradas	4704	1386	29

Analisando os resultados, salientando que foi usado a mesma placa de prototipação nas três versões do PERS (Spartan 2 – XC2S200E da Xilinx), é possível verificar um aumento significativo de CLBs e Luts de 4 entradas devido a inclusão de criptografia. Este aumento em relação à segunda versão levando-se em consideração ao espaço ocupado no FPGA foi de 25% e 23% respectivamente.

As medidas temporais da terceira versão do processador PERS, quanto ao tempo de propagação no circuito é de 15,969 (ns) e a Freqüência Máxima do Processador é de 62,621 Mhz. Pode-se observar também que a inclusão de criptografia influenciou diretamente no desempenho do PERS, pois apresentou em relação à primeira versão uma queda significativa

de 26 % na freqüência do processador. Estes resultados permitiram a publicação de um artigo no evento WSCAD (*Workshop* em Sistemas Computacionais de Alto Desempenho) (OLIVEIRA et al., 2006).

5.4.6 Algoritmo AES

O algoritmo AES é um cifrador de blocos com tamanho de bloco e chave variáveis entre 128, 192 e 256 bits. O que significa que se pode ter tamanho de blocos com tamanhos de chaves diferentes. Em função do tamanho de bloco e chaves é determinada a quantidade de rodadas necessárias para cifrar/decifrar.

O AES opera desta forma com um determinado número de blocos de 32 bits, que são ordenados em colunas de 4 bytes, as quais são chamadas de *Nb*. Os valores de *Nb* possíveis são de 4, 6 e 8 equivalentes a blocos de 128, 192 e 256 bits.

Assim sempre que *Nb* for referido, significa que se tem *Nb* x 32 bits de tamanho de bloco de dados. A chave é agrupada da mesma forma que o bloco de dados, isto é, em colunas, sendo representado pela sigla *Nk*. Com base nos valores que *Nb* e *Nk* podem assumir é que se determina a quantidade de rodadas a serem executadas, identificada pela sigla *Nr*. Por meio dos dados contidos na tabela 5.13, pode-se verificar as possíveis combinações e o número de rodadas necessárias à execução do algoritmo AES.

Tabela 5.13 *Nr* em função do tamanho de bloco e chave no AES

Nr	Nb=4	Nb=6	Nb=8
Nk=4	10	12	14
Nk=6	12	12	14
Nk=8	14	14	14

O processo de cifrar e decifrar no AES não são funções idênticas, como ocorre na maioria dos cifradores.

O processo de cifrar do AES envolve uma aplicação seqüencial de funções. Na figura 5.13 pode-se analisar a estrutura do processo de cifragem do AES.

Ao analisar essa visão macro do AES, podem-se perceber os indicadores *Nb*, *Nk* e *Nr* que possuem seus valores de acordo com o tamanho de bloco e chave a serem utilizados. Observa-se que a última iteração é diferente das demais.

O processo de decifrar no AES consiste na execução de diferentes operações, em virtude de sua essência matemática. Diferentemente do DES que tem sua estrutura baseada em Feistel (FIPSS46-3, 1999), que possui a característica de ser reversível apenas se invertendo a seqüência das chaves para decifrar, o AES necessita de inversas matemáticas de suas transformações para realizar o processo de decifrar.

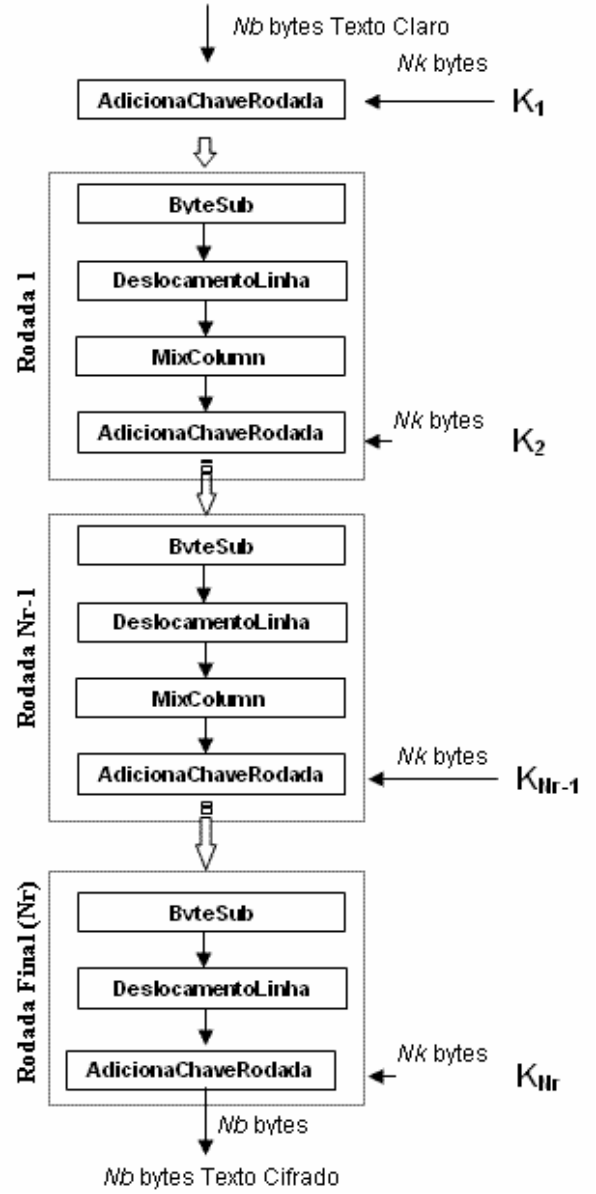


Figura 5.13. Algoritmo de cifragem AES (MORENO et al., 2005)

Observando o AES em uma visão macro, o algoritmo poderia ser considerado como uma seqüência de transformações matemáticas e o seu processo reverso consiste na aplicação da seqüência inversa à da original.

O processo de expansão de chaves continua sendo o mesmo descrito anteriormente, porém as funções de SubByte, ShiftRow e MixColumn necessitam as suas inversas matemáticas para realizar o processo de decifrar.

Analisando o fluxo demonstrado na figura 5.14, podem-se perceber as diferenças em função do processo utilizado para cifrar.

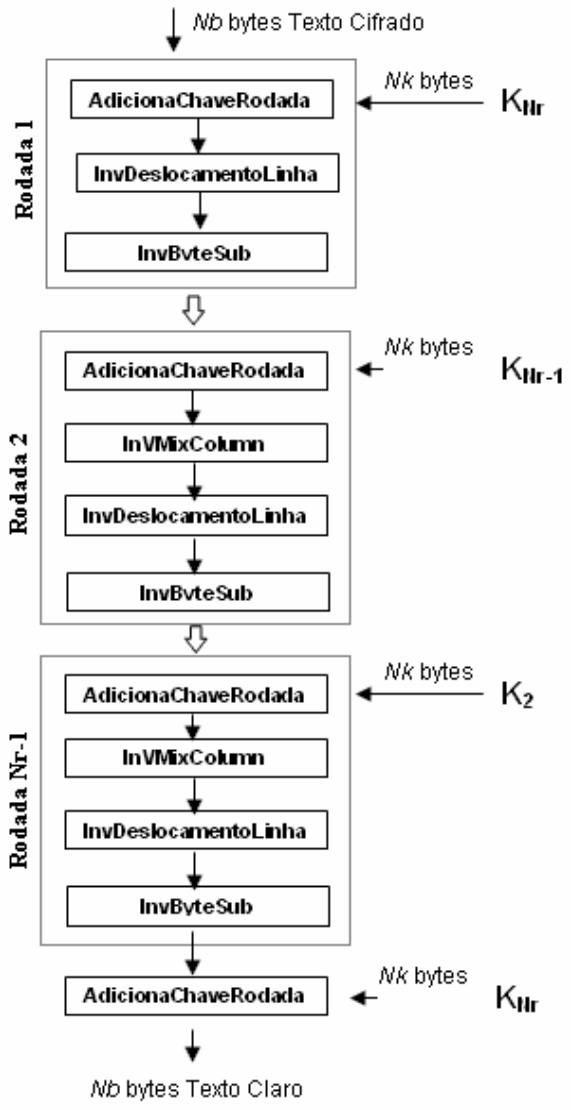


Figura 5.14. Algoritmo de decifragem AES (MORENO et al., 2005)

Realiza-se a operação inversa da ShiftRow efetuando-se rotacionamento cíclico à direita na mesma quantidade de *bytes* da operação de cifrar, a operação inversa mais complexa é a MixColumn. Maiores detalhes das operações e da implementação do algoritmo AES podem ser obtidas em MORENO et al. (2005).

5.4.7 PERS 4 - Arquitetura com AES

Nesta versão do PERS com o algoritmo criptográfico AES não foi necessário incluir novos registradores específicos para criptografia, pois utilizamos os mesmos registradores da versão com o algoritmo DES.

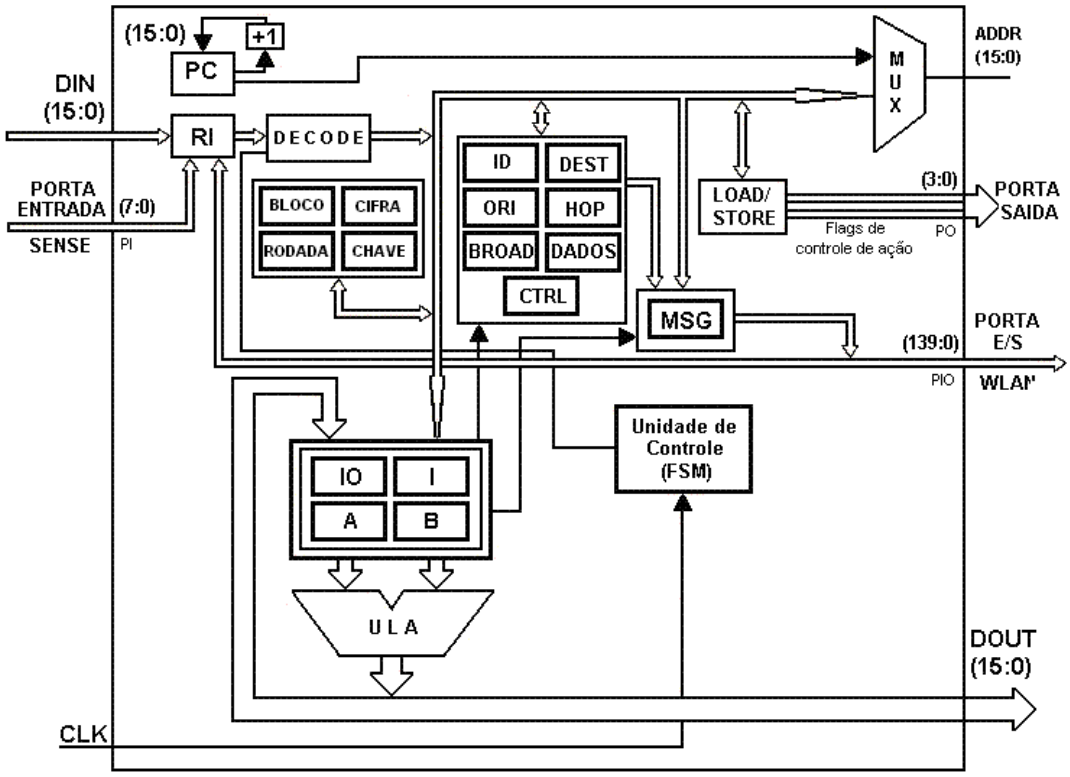


Figura 5.15. Arquitetura detalhada do PERS com opções de criptografia - AES

Houve alterações no tamanho de alguns registradores devido ao algoritmo AES nesta versão, utilizarem 128 bits para cifragem e decifragem dos dados coletados pelos sensores.

Desta forma os registradores BLOCO (64 bits), CIFRA (64 bits) e CHAVE (56 bits) foram aumentados para 128 bits, que são utilizados pelo algoritmo DES.

Foi necessário também aumentar o tamanho da porta (PIO) e dos registradores (IO e MSG), com isso, foi aumentado de 20 bits para 140 bits, conforme figura 5.15.

5.4.8 Programa Teste

Nesta quarta versão do PERS foi utilizado o mesmo programa da versão anterior, ou seja, aproveitaram-se as instruções CRIPTA e DECRIPTA usadas também pelo algoritmo DES. Houve alterações na quantidade de ciclos das instruções de cifragem e decifragem, devido o algoritmo AES de 128 bits ter menos iterações que o algoritmo DES (10 contra 16).

A quantidade de ciclos das instruções CRIPTA e DECRIPTA diminuíram de 19 para 17 ciclos, consequentemente diminuindo a quantidade total de ciclos do programa mostrado na tabela 5.11 que de 82 passou para 78.

5.4.9 Simulações usando a ferramenta Xilinx

Na figura 5.16 se observa a simulação da cifragem de um dado utilizando o algoritmo AES. Após as 10 iterações necessárias, o dado cifrado é armazenado no registrador MSG que depois será preparado com o cabeçalho (endereço do nó origem, endereço do nó destino e *hopcount*) para ser enviado ao próximo nó da rede.

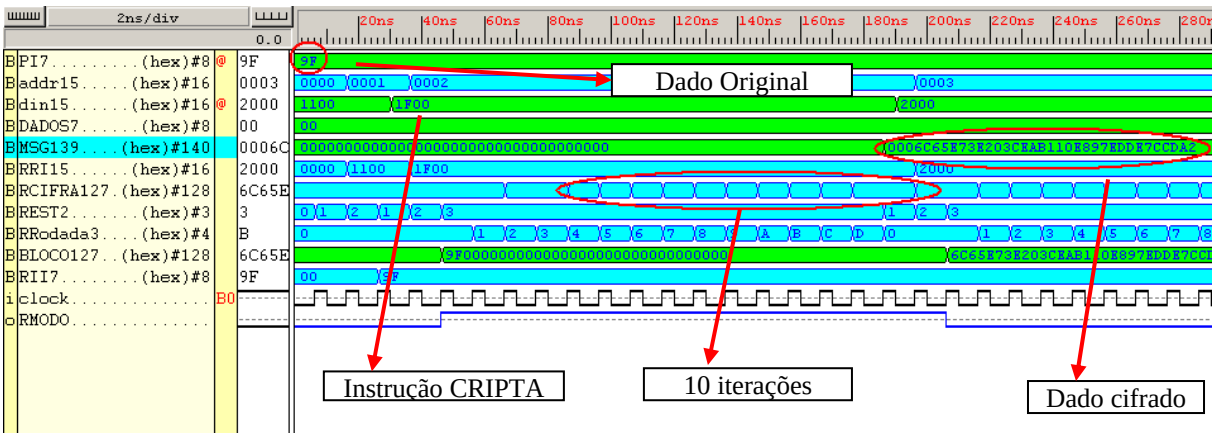


Figura 5.16. Cifragem de um dado no PERS 4

A simulação mostrada na figura 5.16 corresponde à instrução CRIPTA implementada em VHDL, na seguinte estrutura:

```
when cripta =>   RESET <= '0';
                 CHAVE <= "00000000000000000000000000000000000000000000"&
                          "00000000000000000000000000000000000000000000"&
                          "00000000000000000000000000000000000000000000000000";
                 MODO  <='1';
                 BLOCO <= I&PAD;
                 EST <= CRIPTO;
```

O estado CRIPTO descrito a seguir foi necessário um laço para aguardar as 10 iterações necessárias do algoritmo do AES e depois cifrar ou decifrar o dado de acordo com o valor atribuído para o registrador MODO, sendo valor ‘1’ para cifrar e ‘0’ para decifrar (PEREIRA, 2004):

```
when CRIPTO =>          Rodada<=Rodada+1;
                        if Rodada="1101" then
                            Rodada<="0000";
                                if MODO='1' then
                                    MSG(127 downto 0) <= CIFRA;
                                elsif MODO='0' then
                                    DADOS <= CIFRA(127 downto 120);
                                end if;
                        EST <= busca;
                        end if;
```

O algoritmo foi adicionado como um componente do processador PERS, pois já estava implementado em VHDL (PEREIRA, 2004), usando o código que o declara como um componente:

```
component aes
                port (clk       : in std_logic;
                reset       : in std_logic;
                plaintext : in  std_logic_vector(127 downto 0);
```

```
                user_key   : in  std_logic_vector(127 downto 0);
                ciphertext : out std_logic_vector(127 downto 0);
                encrypt    : std_logic);
end component;
```

Finalizando, chama-se o componente declarado no código do PERS, usando a seguinte declaração:

```
U1:AES port map (  clk      => Clock,
                   reset    => RESET,
                   plaintext  => BLOCO,
                   user_key   => CHAVE,
                   ciphertext => CIFRA,
                   encrypt    => MODO);
```

5.4.10 Estatísticas de prototipação em FPGAs

As estatísticas de desempenho são apresentadas na tabela 5.14, vale ressaltar a robustez do algoritmo AES, pois foi necessária a troca da placa para a prototipação em FPGA, sendo utilizada a placa Virtex – V800FG680-5 Xilinx.

A tabela 5.14 apresenta-se os testes que deram informações quanto ao tempo de propagação do circuito, freqüência máxima do processador, assim como o espaço ocupado pelo circuito no FPGA não só da quarta versão do PERS com o algoritmo AES, como de todas as versões anteriores.

Tabela 5.14 Desempenho do PERS de todas as versões

Versão	**Tempo de propagação**			**Freqüência máxima**					
PERS 1	13,190 ns			75,815 Mhz					
PERS 2	13, 938 ns			71,746 Mhz					
PERS 3 - DES	17,485 ns			57,192 Mhz					
PERS 4 - AES	29,884 ns			33,463 Mhz					
Taxa de ocupação FPGA - Virtex - V800FG680-5									
Versão	**CLBs**			**Flip Flops**			**Luts de 4 entradas**		
	Total	**Ocupado**	**%**	**Total**	**Ocupado**	**%**	**Total**	**Ocupado**	**%**
PERS 1	9408	164	1,74	18816	110	0,58	18816	302	1,60
PERS 2	9408	160	1,70	18816	125	0,66	18816	297	1,58
PERS 3 - DES	9408	757	8,04	18816	92	0,49	18816	1386	7,36
PERS 4 - AES	9408	2072	22,02	18816	269	1,43	18816	4109	21,84

Em relação ao tempo de propagação, a versão do PERS com o algoritmo AES apresentou uma queda significativa de 41 % na freqüência do processador em relação à versão do PERS com o algoritmo DES.

Pode-se basear na quantidade de bits cifrados para justificar esta queda de desempenho temporal (TP) mostrado nas estatísticas de desempenho, pois comparando o desempenho pela quantidade de *Megabits* cifrados o AES é superior, pois com 10 iterações e um total de 17 ciclos leva 508,03ns para cifrar 128 bits, fazendo as conversões o AES cifra 240,28 Mbits/seg, sendo que o DES com 16 iterações e um total de 19 ciclos leva 332,21ns para cifrar 64 bits, resultando em 183,72 Mbits/seg. Fica claro que não se tem apenas uma ou duas métricas para avaliar o desempenho de um algoritmo, e sim, uma série de fatores para definir qual é o melhor algoritmo a ser utilizado.

6. Conclusões

A grande dificuldade encontrada neste trabalho foi obter um nível de segurança que impeça os diversos tipos de ataques existentes nas redes de sensores sem que isso prejudique a vida útil dos mesmos, devido às suas limitações de recursos, principalmente, a limitação de energia. Com os estudos feitos neste trabalho sobre conceitos de redes e redes de sensores, conceitos de segurança e conceitos de processadores de rede e as dificuldades encontradas durante este trabalho serviram de estímulo e contribuíram para a o desenvolvimento e implementação do PERS um Processador Específico para Redes de Sensores com Primitivas de Segurança, na qual, contribuirá para a pesquisa científica, pois esse trabalho visa aumentar o desempenho e segurança de aplicações específicas para redes de sensores.

O PERS é um processador que aproveitou as funções Lógicas e Aritméticas, de Movimentação e de Desvio do NPSoC (PRADO, 2004) e também, as instruções de segurança do Criptoprocessador VLIW de (PEREIRA, 2004), além de incluir as instruções específicas para as redes de sensores.

O processador PERS é uma arquitetura RISC e foi implementado em quatro versões, na primeira e segunda versão tinham respectivamente 20 e 30 instruções e não continham segurança na transmissão dos dados. O desempenho quanto ao tempo de propagação foi de 13,190ns (75,815 Mhz) na primeira versão e 13,938ns (71,746 Mhz) na segunda versão, bem superior às terceira e quarta versão que, com a inserção de criptografia, aumentaram para 32 instruções.

Comparando o desempenho dos algoritmos utilizados neste trabalho que correspondem a terceira e quarta versão do processador PERS, pode-se notar que na terceira versão com o algoritmo DES o tempo de propagação foi de 17,485ns (57,192 Mhz), bem

superior a quarta versão com o algoritmo AES que apresentou um tempo de propagação de 29,884ns (33,463 Mhz).

Pode-se basear na quantidade de bits cifrados para justificar esta queda de desempenho temporal (TP) mostrado nas estatísticas de desempenho, pois o DES consome 19,55ns para cifrar 64 bits enquanto que o AES consome 28,789ns para cifrar 128 bits. O número de iterações é um fator importante, pois o AES executa em 10 ciclos enquanto que o DES processa em 16.

No mesmo contexto, comparando o desempenho pela quantidade de *Megabits* cifrados o AES é superior, pois cifra 240,28 Mbits/seg, enquanto que o DES cifra apenas 183,72 Mbits/seg. Fica claro que não se tem apenas uma ou duas métricas para avaliar o desempenho de um algoritmo, e sim, uma série de fatores para definir qual é o melhor algoritmo a ser utilizado.

Os dados de desempenho demonstrados e analisados mostraram que o processador proposto (PERS) sofreu um grande impacto com a inclusão dos algoritmos de criptografia (DES e AES), porém a inclusão da criptografia é necessária para que os dados transmitidos estejam seguros de algum tipo de ataque. Este impacto correspondeu na diminuição de 24,56% (DES) e 55,86% (AES) na freqüência do processador, porém a inserção de segurança na transmissão dos dados torna esta queda irrelevante.

Tem-se vários trabalhos futuros que visam o melhoramento do PERS, porém a seguir são listados os mais importantes: (i) inclusão de outros algoritmos de segurança (por exemplo, RC5 e Twofish) no PERS para avaliar seus desempenhos; (ii) suporte do processador para outras topologias existentes (por exemplo, árvore binária e anel unidirecional), com isso, será necessário incluir novas instruções de roteamento para efetuar *multicast* e *broadcast* na rede; (iii) diminuição do tamanho da mensagem enviada com os dados de 76 bits para 20 bits. Para isso será necessário quebrar a mensagem em 4 ou 5 pacotes, pois em cada pacote de dados

sempre deverá conter o endereço de origem e destino. Com esta divisão precisa-se ter uma atenção especial para o sincronismo dos pacotes de dados; (iv) inclusão de instruções que analisam o consumo de energia dos sensores e possíveis perdas de nós na rede de sensores; (v) adaptação do processador PERS aos padrões de barramentos industriais como aqueles vistos no capítulo 2; (vi) inclusão do processador PERS para fazer comunicação com uma outra rede de sensores e rede de sensores sem fio.

E principalmente, colocá-lo em funcionamento em uma rede para efetuar a comunicação entre PERS em conjunto com os transceptores de rádio, sensores e bateria, coletando informações reais para verificar seu real desempenho em uma rede de sensores.

Referências

AKYILDIZ, I., SU, W., SANKARASUBRAMANIAM, Y., CAYIRCI, E. (2002) “A Survey on Sensor Networks”, IEEE Communications Magazine, Agosto.

CARVALHO, D. B. (2001), “Criptografia Métodos e Algoritmos”. 2ª Edição. Rio de Janeiro. Editora Book Express.

CHAN, H., PERRIG, A. e SONG, D. (2003) “Random Key Predistribution Schemes for Sensor Networks”, IEEE Symposium on Security and Privacy, Maio.

CHONG, C. Y., KUMAR, S. P. (2003) Sensor Network: evolution, opportunities and challenges Proceedings of the IEEE, IEEE, v.91, nº. 8, p. 1247-1256, Aug.

CORREIA, H., CRETU, E., BARTEK, M., WOLFFENBUTTEL, R. F. (1997) A microinstrumentation system for industrial applications. In: International Symposium Industrial Electronics – ISIE, 1997. Proceedings of the..., IEEE, p. 846-840.

DENG, J., HAN, R. e MISHRA, S. (2002) “INSENS: Intrution-Tolerant Routing in Wireless Sensor networks,” TR CU-CS-939-02, Dept of Computer Science, University of Colorado.

DIFFIE, W. e HELLMAN, M. E. (1976) New Directions in Cryptography, IEEE Transactions on Information Theory, Vol. IT22, N. 6, November.

DOUCEUR, J. R. (2002) “Tha Sybil Attack”, 1st International Workshop on Peer-to-Peer Systems (IPTS`02), Março.

FIPS46-3. (1999) Federal Information Processing Standards Publication 46-3, Data Encryption Standard, Dezembro.

FREITAS, H. C., MARTINS, C. A. P. S., (2000) “Projeto de Processador com Microarquitetura Dedicada para Roteamento em Sistemas de Comunicação de Dados” – WSCAD.

FREITAS, H. C., MARTINS, C. A. P. S., (2001) “Processador de Rede com Suporte a Multi-protocolo e Topologias Dinâmicas” – WSCAD.

FREITAS, H. C., MARTINS, C. A. P. S., (2002) “R2NP: Processador de Rede RISC Reconfigurável” – WSCAD.

GARDNER, J. W. (1994) Microsensors Principles and Applicatios. NY: John Wiley & Sons, Inc. 331 p.

HEIDEMANN, J. SILVA, F., INTANAGONWIWAT, C., GOVINDAN, R., ESTRIN, D. e GANESAN, D. (2001) “Building efficient wireless sensor networks with low-level naming”, In Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles}, Banff, Alberta, Canada, ACM Press, pp 146-159. Disponível em www.isi.edu/~johnh/PAPERS/Heidemann01c.pdf.

HU, F., TILLETT, J., ZIOBRO, J., SHARMA, N. K. (2003) “A Survey on Securing Wireless Sensor Networks”, Submetido ao IEEE Communications Surveys, Janeiro.

HU, Y. C., PERRIG A., JOHNSON, D. B. (2002) “Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks”, MobiCOM.

HU, L. e EVANS, D. (2003) “Secure aggregation for wireless networks”, In Workshop on Security and Assurance in Ad hoc Networks, January, disponível em http://www.cs.virginia.edu/~evans/pubs/wsaan-abstract.html.

HUIJSING, J. H., (1992) “Integrated Smart Sensors,” *Sensors and Actuators A*, 30(1-2), pp. 167-174.

IEEE. Institute of Electrical and Electronics Engineers, Computer Society. (1985) IEEE Standard for Local Area Networks – Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications (std. 802.3). IEEE. NY: ANSY/IEEE, 146 p.

IEEE. Institute of Electrical and Electronics Engineers, Instrument and Measurement Society. (1999) IEEE Standard for a Smart Transducer Interface for Sensor and Actuators – Network Capable Application Processor (NCAP) Information Model, (std. 1451.1) Standards Board. NY: IEEE, 341 p.

IEEE. Institute of Electrical and Electronics Engineers, Instrument and Measurement Society. (1997) IEEE Standard for a Smart Transducer Interface for Sensor and Actuators – Transducer to Microprocessor Communication Protocols and Transducer Dara Sheet (TEDS) Formats, (Std. 1451.2) Standards Board.NY: IEEE. 114 p.

KARLOF, C., WAGNER, D. (2003) “Secure Routing in Sensor Networks: Attacks and Countermeasures”, 1st IEEE International Workshop on Sensor Network Protocols and Applications, Maio, disponível em http://webs.cs.berkeley.edu/papers/sensor-route-security.pdf

KESTER, W., CHESTNUT, B., KING, G. Smart Sensors. (2005) Papers in PDF. Disponível em: http://www.analog.com/technology/amplifiersLinear/training/pdf/Sensor_ Sect9.pdf.

LAW, Y. W., CORIN, R., ETALLE, S., HARTEL, P. H. (2003) “A Formally Verified Decentralized Key Management Architecture for Wireless Sensor Networks”, Personal Wireless Communications (PWC 2003), IFIP WG 6.8, Mobile and Wireless Communications, Setembro.

LEE, K. B., SCHNEEMAN, R. D. (1999) Implementing a standard-based distributed measurement and control application on the Internet . Papers in PDF format, June. Disponível em: <ieee1451.nist.gov/framework.pdf>

LEE, K. B. Sensor networking and interface standardization. (2001) In: Instrumentation and Measurement Technology Conference – IMTC, 18, 2001, Budapest. Proceedinf of the…, Budapest: IEEE, v.1, p. 147 – 152.

LEVIS, P., CULLER, D., (2002) Mate' – a Virtual Machine for Tiny Networked Sensors, ASPLOS, Oct., 85-95.

LIU, D. e NING, P. (2003) "Efficient Distribution of Key Chain Commitments for Broadcast Authentication in Distributed Sensor Networks", In Proceedings of the 10th Annual Network and Distributed System Security Symposium, p. 263-276, Fevereiro.

LOPEZ, M. J. L., (2004) "Criptografía y Seguridad en Computadores". Tercera Edición, Marzo.

LOUREIRO, A. A. F., NOGUEIRA, J. M. S., LINNYER B. R. e RAQUEL, A. F. (2002) Redes de Sensores Sem Fio. Mini Simpósio Brasileiro de Computação, Jornada de Atualização à Informática. Florianópolis, Santa Catarina - Brasil.

MALLADI, R. e AGRAWAL, D. P. (2003) "Current and future applications of mobile and wireless networks", Communications of the ACM, ACM Press, ISSN 0001-0782, n. 10, pp 144-146, 2002, disponível em http://doi.acm.org/10.1145/570907.570947.

MEGERIAN, S., KOUSHANFAR, F., QU, G., VELTRI, G. e POTKONJAK, M. (2002) "Exposure in wireless sensor networks: theory and practical solutions", Wireless Networks, Kluwer Academic Publishers, ISSN 1022-0038, vol. 8, no. 5, pp 443-454. Disponível em : http://www.ece.wisc.edu/~megerian/papers/exposure_journal.pdf.

MENEZES, A., OORSCHOT, P. V., VANSTONE, S., (1996) "Handbook of Applied Cryptogrphy". Editora CRC Press.

MORENO, E. D., PEREIRA, F. D., PENTEADO, C. G., PERICINI R. A., (2003) "Projeto, Desenvolvimento e Aplicações de Sistemas Digitais em Circuitos Programáveis (FPGAs)". Marília: Editora Bless.

MORENO, E. D., PEREIRA, F. D., CHIARAMONTE, R. B., (2005) "Criptografia em Software e Hardware – Implantação e Desempenho". Marília. Editora Novatec.

NAJAFI, N., WISE, K. D. (1990) An organization and interface for sensor-driven semiconductor process control systems. IEEE Trans. Semiconductor Manufacturing, IEEE, v.3, n.º 4, p. 230-238, Nov.

OLIVEIRA, A. P., MORENO, E. D., BRANCO, K. R. L. J. C., (2006) PERS – Um Processador Específico para Redes de Sensores com Primitivas de Segurança – WSCAD.

PALLÁS-ARENY, R., WEBSTER, J. G. (1994) Sensors and Signal Conditioning. 1 ed. NY: John Wiley & Sons, Inc., 398 p.

PEREIRA, F. D., (2004) "Um Criptoprocessador VLIW para Algoritmos Criptográficos Simétricos". Dissertação de Mestrado em Ciência da Computação do PPGCC da UNIVEM, Marília.

PERRIG, A., SZEWCZYK, R., WEN, V., CULLER, D. e TYGAR, J. D. (2001) "SPINS: security protocols for sensor networks", In Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking, Rome, Italy, ACM Press, pp 189-199.

PRADO, R. P., (2004) "NPSoC – Arquitetura e Protótipo de um Novo Processador de Rede". Dissertação de Mestrado em Ciência da Computação do PPGCC da UNIVEM, Marília.

ROSSI, S. R. (2004) Implementação de um nó IEEE 1451, baseado em ferramentas abertas e padronizadas, para aplicações em ambientes de instrumentação distribuída. Tese (Doutorado). Universidade Estadual Paulista, Departamento de Engenharia Elétrica. Ilha Solteira. p. 15 – 51.

SILVA, G. E. F., (2003) "Análise comparativa entre os quatro Algoritmos de Chave Simétrica submetidos ao Projeto NESSIE – Segunda Etapa". 2003. Trabalho de Conclusão de Curso em Ciência da Computação da Universidade Luterana do Brasil, Gravataí.

STANKOVIC, J. A. (2002) “A network virtual machine for real time-coordination”, The Real-Time Computing Laboratory, University of Virginia, disponível em: http://www.cs.virginia.edu/nest.

SZE, S. M. (1994) Semiconductor Sensors NY: John Wiley & Sons, Inc., 550 p.

TANENBAUM, A. S. (1997) Redes de Computadores. 3 ed. Rio de Janeiro: Campus, 884 p.

TILAK, S., ABU-GHAZALEH, N.B. e HEINZELMAN, W. (2002) “A taxonomy of wireless micro-sensor network models”, In Proceedings of the ACM Workshop on Wireless Security, ACM Press, pp 28-36. Disponível em: http://www.cs.binghamton.edu/~sameer/pubs /wcnc02-draft.pdf.

TORRES, G. (2001) Redes de Computadores: Curso Completo. Rio de Janeito: Axcel Books do Brasil, 664 p.

WARRIOR, J. (2005) Smart sensor networks of the future. Sensors Magazine ,1997. Disponível em: http://www.sensorsmag.com/articles/1097/ieee1097/main.shtml.

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

YAMASAKI, H. (1996) Intelligent Sensors. Serie: Handbook of Sensors and Actuators, v. 3. Amsterdam, NY: Elsevier. 297 p.