Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Sistema de Controle de Acesso Utilizando Dispositivos Embarcados

Thiago Moratori Peixoto

JUIZ DE FORA
ABRIL, 2013

Sistema de Controle de Acesso Utilizando Dispositivos Embarcados

Thiago Moratori Peixoto

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Eduardo Pagani Julio

JUIZ DE FORA

ABRIL, 2013

Sistema de Controle de Acesso Utilizando Dispositivos Embarcados

Thiago Moratori Peixoto

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Eduardo Pagani Julio
Mestre em Computação pela Universidade Federal Fluminense (2007)

Eduardo Barrére
Doutor em Engenharia de Sistemas e Computação - COPPE/UFRJ (2007).

Luciano Jerez Chaves
Mestre em Ciência da Computação pela Universidade Estadual de Campinas - UNICAMP (2010)

JUIZ DE FORA

PRIMEIRO DE ABRIL, 2013

Aos meus amigos e irmãos por fazerem meus dias mais alegres.
Aos meus pais, pelo incentivo, força e presença em todas as situações.
Ao meu orientador Eduardo Pagani pela técnica, suporte intelectual e pela paciência.
E à todos que de alguma forma contribibuiram para o desenvolvimento desse trabalho.

Resumo

Os sistemas de controle de acesso à salas de aula, departamentos e laboratórios do Instituto de Ciências Exatas (ICE) da Universidade Federal de Juiz de Fora (UFJF) hoje em dia são feitos através do uso de claviculário manual e de porteiro eletrônico. Esses sistemas apresentam alguns problemas em relação a gerenciamento e portabilidade. O trabalho aqui proposto visa apresentar um sistema capaz de sanar esses problemas. Esse sistema consiste em duas entidades principais: um *middleware* e um servidor de autenticação. O *middleware* deve receber as informações de um usuário passadas através de uma *tag RFID*, montar uma mensagem de requisição e enviá-la de maneira segura via rede *ethernet* para o servidor de autenticação. Este, por sua vez, retorna uma mensagem ao *middleware* com a resposta à requisição. O sistema é centralizado em rede, de maneira que todos os locais em que o *middleware* for implementado fazem requisição ao mesmo servidor. Nesse trabalho são apresentados aspectos referentes ao desenvolvimento das duas entidades, à infraestrutura necessária para a implantação do sistema, uma estimativa de custos para ela e ainda um estudo sobre o *overhead* gerado pelo uso de criptografia para garantir segurança ao sistema.

Palavras-chave: Sistema de controle de acesso, *middleware*, segurança, RFID.

Abstract

The systems control access to classrooms, departments and laboratories of the Instituto de Ciências Exatas (ICE) da Universidade Federal Juiz de Fora (UFJF) today are made through the use of manual key storage cabinet and a doorphone. These systems present some issues regarding management and portability. The work proposed here aims to present a system capable of solving these problems. This system consists of two main entities: a middleware and an authentication server. The middleware must receive information that a user passes through a RFID tag, assemble a request message and send it securely over ethernet network to the authentication server. This, in turn, returns a message to the middleware with the response to the request. The system is centered on a network, so that all locations that the middleware is deployed to, makes requisitions to the same server. In this paper are presented aspects related to the development of the two entities, the necessary infrastructure for the deployment of the system, a cost estimate for it and also a study on the overhead generated by the usage of encryption to ensure system security.

Key-words: system for access control, middleware, security, RFID.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

ICE	Instituto de Ciências Exatas
DIP	*Dual In-line Package*
PCB	*Printed Circuit Board*
USB	*Universal Serial Bus*
LCD	*Liquid Crystal Display*
RFID	*Radio-Frequency IDentification*
SPI	*Serial Peripheral Interace*
I2C	*Inter-Integrated Circuit*
CAN	*Controller Area Network*
AES	*Advances Encrytion Standard*
ECB	*Electronic CodeBook*
CBC	*Cipher Block Chaining*
NIST	*National Institute of Standards and Technology*
DES	*Data Encryption Standard*
RAM	*Random Access Memory*
ROM	*Read Only Memory*
SSL	*Secure Sockets Layer*
IV	*Initialization Vector*
TCP	*Transmission Control Protocol*
ACK	*ACKnowledgement*
VLAN	*Virtual Local Arena Network*
FIPS	*Federal Information Processing Standard*
IP	*Internet Protocol*
SQL	*Structured Query Language*
ILP	*Intruction-Level Parallelism*
VoIP	*Voice over Internet Protocol*

1 Introdução

O uso de sistemas de controle de acesso vem se tornando algo comum nos mais diversos ramos de atividades e ambientes organizacionais. Tais tecnologias se fazem necessárias a fim de prover maior segurança tanto para usuários quanto para instituições. As universidades públicas são alvos constantes de investimentos em tecnologia de ponta e, consequentemente, são muito visadas por criminosos.

Neste ambiente surgem inúmeras ferramentas com o intuito de prover maior segurança e controle dos mais diversos tipos de usuários de uma instituição. Apesar da popularização destes sistemas, os mesmos ainda possuem alto custo de implantação e geralmente não possuem padronização das funcionalidades, principalmente em se tratando do sistema (*middleware*) que os gerenciam.

Este trabalho apresenta uma solução de controle de acesso de usuários às dependências do Instituto de Ciências Exatas (ICE) da Universidade Federal de Juiz de Fora (UFJF), mais precisamente salas de aula, laboratórios de computação e departamentos do instituto.

O ICE conta atualmente com 5 (cinco) laboratórios de informática, 40 (quarenta) salas de aula e 3 (três) anfiteatros. Tais dependências equipadas com mais de 300 computadores, 43 projetores de vídeo, entre outros dispositivos eletrônicos. Além disto, possui inúmeras salas de projetos e pós graduações, com equipamentos tecnológicos com alto valor monetário. Como já descrito, neste contexto surge a preocupação com o controle de acesso as dependências do instituto, principalmente àquelas detentoras de equipamentos de alto custo, uma vez que tendem a ser as mais visadas.

O controle de acesso utilizado atualmente para as salas de aula do instituto é feito por um claviculário manual. Neste sistema cada professor faz uso de um cartão (padrão a todos os usuários) para liberar a chave de determinada sala. Caso algum usuário esteja de posse de determinada chave ou se esqueça de devolvê-la, a sala fica bloqueada para uso, causando transtorno aos demais usuários.

Além disso, o acesso aos departamentos, por exemplo o DCC (Departamento de

Ciência da Computação), é restrito somente a pessoas autorizadas. Assim, um número alto de cópias de chaves deve ser feito para atender a esta demanda, que atualmente é próxima a 60 usuários. Para evitar gastos com chaves e eventuais gastos extras, caso uma fechadura tenha de ser trocada, por exemplo, foi instalado um porteiro eletrônico no departamento. O mesmo foi integrado a um sistema de voz sobre *IP* (*VoIP*) que permite tanto a abertura das portas através de senha, quanto a comunicação com os gerentes do sistema. Entretanto, este sistema opera de maneira isolada, ou seja, não permite nem o controle centralizado do acesso nem o registro de tentativas de acesso não autorizados. Outro problema está no retrabalho necessário para cadastramento de senhas para, por exemplo, este sistema ser implantado em outro local e, caso o porteiro apresente problema, todas as senhas deverão ser recadastradas.

A finalidade do sistema de acesso aqui proposto é sanar os problemas descritos. No trabalho desenvolvido, o sistema centralizado faz uso de um controle com granularidade fina, onde cada usuário estará autorizado a abrir uma determinada porta somente em horário permitido. Assim um controle de acesso ao instituto se torna mais efetivo permitindo inclusive registro de tentativas de acesso indevido. Com esse sistema o reaproveitamento da cartão de acesso será possível, de tal maneira que se possa diminuir o custo com chaves para diversas pessoas além do controle individual de acesso feito por cada indivíduo.

Uma interface amigável para cadastro dos usuários, cartões e horários autorizados em cada porta também está sendo desenvolvida de modo a facilitar a administração do sistema. Nessa interface será permitida a delegação de responsabilidades para administração de usuários e ambientes, aqui denotados por portas. Tal interface encontra-se em andamento, fará parte do sistema (iNtegra, 2010) e não é apresentada neste trabalho.

Ao trabalho, dá-se a organização descrita a seguir. Primeiramente, no Capítulo 2 são apresentados o *mbed* Holdings (2011), principal componente de *hardware* utilizado no sistema, a tecnologia usada para identificação por rádio frequência, *RFID* (*Radio-Frequency IDentification*) e ainda alguns conceitos de segurança pertinentes ao sistema proposto, incluindo a cifra de chave simétrica *AES* utilizada para conferir segurança à comunicação no projeto. No Capítulo 3 são apresentados os trabalhos relacionados a

este tema e um breve comparativo entre a plataforma aqui proposta e as encontradas em outros trabalhos. O Capítulo 4 trata o sistema em sua complexidade, na qual são apresentados os dados do ICE e o ambiente físico em que o sistema é utilizado levando em consideração a infraestrutura necessária para o funcionamento correto do sistema. É neste capítulo também que se apresentam o *middleware* do sistema e o servidor de autenticação, bem como todas as características que lhes são peculiares. No Capítulo 5 são mostrados os dados comparativos do uso ou não de criptografia *AES* (*Advanced Encryption Standard*)nas trocas de mensagens realizadas no sistema, o cálculo do *overhead* gerado pelo uso da mesma, tanto em relação ao tempo resposta quanto ao volume de dados trafegados na rede e uma análise sobre o impacto significativo (ou não) desses valores sobre o funcionamento adequado do sistema. Por fim, no Capítulo 6, são realizadas as considerações finais do sistema, a conclusão sobre a análise do *overhead* do *AES* sobre o sistema e alternativas para trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo são apresentados os conceitos relacionados ao desenvolvimento do trabalho. É nessa parte que se descreve o principal componente de *hardware* do sistema, o microcontrolador *mbed*, bem como a tecnologia por trás da processo de identificação dos usuários, a *RFID*, e ainda todos os conceitos de segurança pertinentes ao contexto do trabalho.

2.1 Sistemas Embarcados

Conforme Vahid and Givargis (2002), um sistema embarcado consiste em um sistema microprocessado com um propósito específico. Ao contrário de computadores que possuem um propósito mais abrangente, o sistema embarcado geralmente é focado para realizar um conjunto restrito de tarefas, porém o mesmo é capaz de englobá-las sem a utilização de demais dispositivos. Um exemplo clássico de sistemas embarcados são os chamados computador de bordo encontrados em veículos modernos. Estes sistemas geralmente desempenham o papel de calcular a média de combustível gasto em uma viagem, a temperatura ambiente e o tempo de viagem.

Uma vez que estes sistemas desempenham papéis específicos, podem ser otimizados para ter tamanho pequeno, portabilidade através de replicação de *middleware*, baixo custo, entre outros.

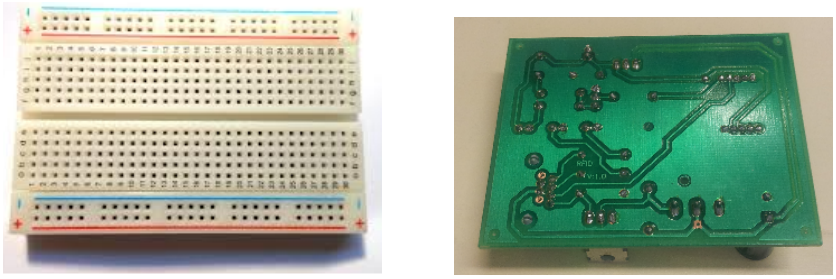
Dentro dessa vertente, diversas plataformas vem surgindo para facilitar a implementação de sistemas embarcados.

2.2 *mbed*

O microcontrolador *mbed* é o principal componente de *hardware* aqui descrito. Conforme (Holdings, 2011), ele pertence a uma série de microcontroladores elaborados para prototipagem rápida e é projetado com encapsulamento *DIP* (*dual in-line package*) para

montagem em *PCBs* (placas de circuito impresso) e/ou em *protoboards* (geralmente utilizadas na fase inicial do projeto para realização de testes).

A Figura 2.1 representa os tipos de placa com os quais o *mbed* é projetado para funcionar.



(a) Placa *Protoboard* (b) Placa de circuito impresso

Figura 2.1: Tipos de placa suportadas pelo *mbed*

A versão do *mbed* utilizada neste trabalho é o NXP LPC1768 uma vez que esse possui, em relação à outra versão disponível (NXP LPC11U24), maior capacidade de processamento e memória, além do suporte à rede *ethernet*, elemento essencial para o desenvolvimento do sistema.

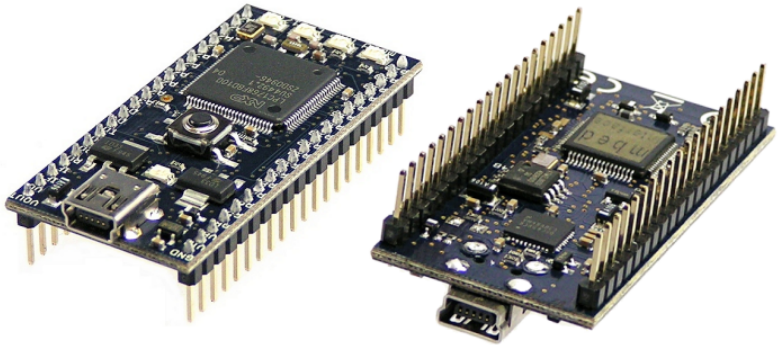


Figura 2.2: Visão frontal do *mbed* NXP LPC1768.

Esses microcontroladores incluem uma interface *USB* de acesso para programação. Para uso, basta se conectar o dispositivo a uma porta *USB* do computador, copiar o programa ARM (Holdings, 1983) binário para a memória *flash* para que o processador seja capaz de executar o código. Essa interface é compatível com os sistemas Windows, Mac OS X e Linux. E, desta forma, a plataforma é indiferente ao sistema do *mbed*. O programa binário pode ser obtido utilizando-se um compilador em "nuvem", disponível através do *site* do próprio fabricante (Holdings, 2011).

O *mbed* possui inúmeras portas de conexões. No total, são 25 pinos disponíveis para o LPC1768. Através destes pinos podem ser incorporados dispositivos e funcionalidades como interface de rede *Ethernet*, leitores e gravadores *RFID*, leitores biométricos, câmeras, teclados numéricos, telas *LCD touch screen*, entre outros. Além disto, esses microcontroladores podem ser alimentados através da interface *USB* de um computador ou através de fontes externas com voltagem operando entre 4.5V e 12V. A Tabela 2.1 ressalta as principais características deste modelo de microcontrolador.

Tabela 2.1: Principais características de *hardware* do NXP LPC1768.

Característica	**Valor**
Aplicações alvo	*Ethernet*, *USB* e de alto desempenho
Núcleo de processamento	*ARM Cortex-M3*
Frequência de operação	96 MHz
Memória *Flash*	512 KB
Memória *RAM*	32 KB
Alimentação	60-120 mA

Esse modelo também oferece as seguintes interfaces periféricas: suporte para controlador *Ethernet*, *USBHost* e *USBDevices*, *SPI*, *I2C*, *CAN*, *AnalogIn*, *PwmOut*, *AnalogOut*, *DigitalIn* e *DigitalOut*. Cada uma delas são detalhadas a seguir.

A Figura 2.3 pode ser utilizada como um guia para usuários do dispositivo, pois mostra a numeração dos pinos e suas funcionalidades.

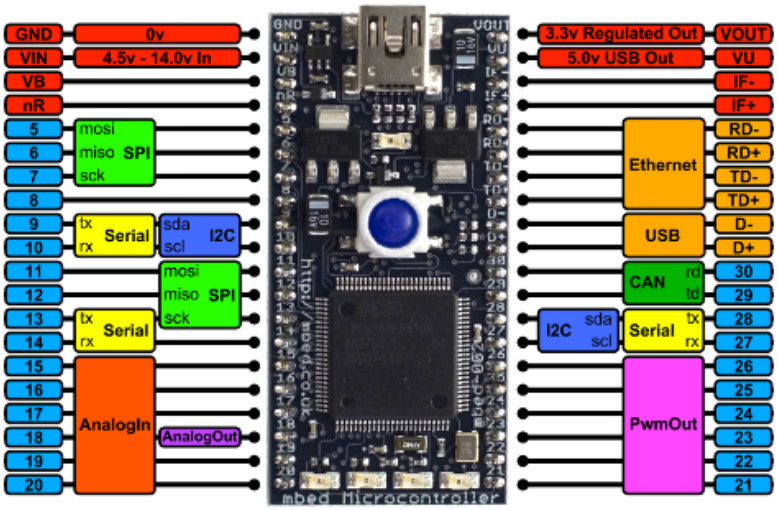


Figura 2.3: Diagrama de conexões do *mbed* NXP LPC1768.

- ***Ethernet***: permite que os microcontroladores *mbed* se conectem e se comuniquem com uma rede *Ethernet*, podendo ser utilizado para conversar com outros disposi-

tivos na rede, incluindo comunicação com computadores através da *web*, servidores de *email* ou atuar com um *webserver* (utilizada nos pinos p33/34/p35/p36);

- ***USB***: com essa interface é possível utilizar o *mbed* para, por exemplo, emular *mouse* e teclado, emular uma porta serial, enviar e receber mensagens MIDI, tocar músicas ao ser reconhecido como um dispositivo de áudio e receber pacotes de áudio vindos do computador (utilizada nos pinos p31 e p32);

- ***SPI***: fornece uma interface periférica serial mestre que permite a comunicação com dispositivos *SPI* escravos, como memória *flash*, telas *LCD* e outros módulos ou circuitos integrados (utilizada nos pinos p5/p6/p7 ou p11/p12/p13);

- ***I2C***: fornece funcionalidade mestre *I2C*, que por sua vez propicia a comunicação com dispositivos *I2C* escravos, como memória serial, sensores e outros módulos ou circuitos integrados (utilizada nos pinos p9/p10 ou p27/p28);

- ***CAN***: é um padrão de barramento projetado para permitir que os microcontroladores e dispositivos se comuniquem uns com os outros sem um computador *host* (utilizada nos pinos p9/p10 ou p29/p30);

- ***AnalogIn***: usada para ler a tensão aplicada a um pino de entrada analógica (pode utilizar qualquer pino de p15 a p20);

- ***PwmOut***: utilizada para controlar a frequência de uma sequência de pulsos digitais (pode utilizar qualquer pino de p21 a p26);

- ***AnalogOut***: utilizada para ajustar a tensão de um pino de saída analógica (só pode utilizar o pino p18);

- ***DigitalIn***: utilizada para ler o valor de um pino de entrada digital (pode utilizar qualquer pino de p5 a p30);

- ***DigitalOut***: utilizada para configurar e controlar um pino de saída digital (pode utilizar qualquer pino de p5 a p30).

Utilizou-se o *mbed* em detrimento à outros microcontroladores (como o Arduino por exemplo Arduino (2012)) uma vez que o mesmo possui vantagens no quesito custo [1] e facilidade de desenvolvimento de código [2].

2.3 *RFID*

O nome *RFID*, do inglês *Radio-Frequency IDentification* (Juels, 2006), ou Identificação por radiofrequência, é uma tecnologia para identificação automática de objetos e pessoas. Um dispositivo *RFID* é comumente conhecido como *tag* ou *token RFID*. Consiste de um pequeno *microchip* construído para transmissão de dados sem fio, em resposta a uma interação com o leitor *RFID* (emissor de rádio frequência). Esta interação geralmente se dá na aproximação do dispositivo ao leitor. Os dispositivos geralmente estão associados à uma antena em um conjunto muito similar a um adesivo. A Figura 2.4 representa um exemplo de *tag RFID* em formato adesivo.

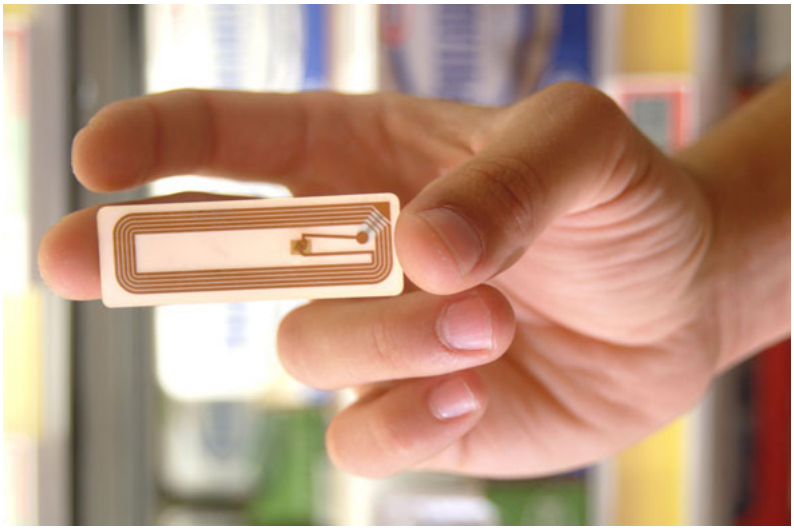


Figura 2.4: *Tag RFID* em formato adesivo.

Atualmente existem três tipos diferentes de *tags RFID* de acordo com Lahiri (2005); Weis (2007): as passivas, as semi-passivas (ou semi-ativas) e as ativas. As passivas, utilizadas nesse trabalho, utilizam a rádio frequência do leitor para transmitir o

[1] O *mbed*, que já possui interface de rede, custa cerca de US$50,00, já o conjunto do Arduino mais *shield* de rede (módulo necessário para integração dessa interface ao dispositivo), custa cerca de US$75,00, quase 50% mais caro.

[2] O código utilizado para desenvolvimento no *mbed* é C, uma linguagem mais comum com vasta documentação disposta na *web*, já o Arduino possui uma linguagem própria parecida com C, que também possui documentação na *web* Arduino (2012), mas que é menos difundida quando comparada ao C

seu sinal e normalmente têm suas informações gravadas permanentemente quando são fabricadas (algumas destas etiquetas são regraváveis). Apesar de terem o menor alcance dos três tipos, as *tags* passivas são as mais baratas de se fabricar e as mais fáceis de serem integradas à produtos além de serem as mais comuns de se encontrar.

As *tags* semi-passivas possuem bateria interna, diferem das ativas pelo fato de não serem capazes de iniciar comunicação. Devido à bateria interna, *tags* semi-passivas são capazes um alcance maior de leitura do que as passivas, porém possuem maior custo.

Já o terceiro tipo, *tags* ativas, possuem bateria interna e são capazes de iniciar a comunicação. São as mais interessantes do ponto de vista funcional. Uma aplicação de *tags* ativas é proposta em Weis (2007) e sugere o uso para controle de cargas em um navio. Tal sistema funcionaria da seguinte maneira: uma vez acoplada a um *container*, uma *tag* ativa, em conjunto com um sensor de movimento por exemplo, poderia enviar uma mensagem notificando o cargueiro caso o *container* apresente, por exemplo, uma movimentação estranha e esteja próximo de cair no mar.

De acordo com as publicações propostas em Ukkonen et al. (2007); Roberts (2006); Landt (2005); Sen et al. (2009); Weis (2007), pode-se descrever as faixas de leitura de *tags RFID* conforme a Tabela 2.2.

Tabela 2.2: Tabela comparativa de frequências de operação *RFID*

Frequência	Tipo de *Tag*	Categoria	Alcance(cm)	Velocidade de Transferência
125 e 134,3 kHz	Passivas	Baixa	10 cm	Baixa
13.56 MHz	Passivas	Alta	100cm	Baixa à Moderada
860 - 960 MHz	Passivas	Ultra Frequência	100cm	Moderada à Alta
2450 - 5800 MHz	Ativas	Microondas	100cm a 200cm	Alta
3.1 - 10 GHz	Ativas ou Semi-ativas	Microondas	20000cm	Alta

2.4 Segurança

Os requisitos de segurança da informação são essenciais a um sistema de controle de acesso. Por possuírem constantes investimentos em tecnologia e equipamentos, as universidades públicas são muito visadas por criminosos. Nesse contexto surge a necessidade de combater e previnir ações que possam comprometê-las. Mecanismos de segurança como instalação de câmeras, contratação de seguranças para vigiar o *campus* e utilização de

fechaduras e cadeados para impedir o acesso indevido às dependências da universidade, já são utilizados. Porém, ainda há a necessidade de impedir ataques à rede interna, provedora de todo o serviço de comunicação do instituto, a fim de garantir maior segurança ao patrimônio público, uma vez que nessa rede trafegam informações sigilosas e que se descobertas podem comprometer o sistema e levar à situações de risco ao patrimônio público. Nesta seção são apresentados conceitos de criptografia, o algoritmo criptográfico utilizado para prover segurança à rede interna do sistema e o porquê da escolha do mesmo.

2.4.1 Criptografia

Na criptologia, estudo de técnicas que asseguram sigilo e autenticidade de informação (Stallings, 2008), existem dois ramos principais: a criptografia e a criptoanálise. A primeira trata do estudo de técnicas capazes de prover a segurança da informação através da transformação de texto puro em texto cifrado, a esse primeiro processo dá-se o nome de cifragem. A segunda se responsabiliza pela tentativa de obtenção do texto puro, a partir do texto cifrado, com conhecimento nulo ou limitado referentes ao primeiro processo, ou seja, um ataque ao sistema.

2.4.1.1 Modelo de Chave Simétrica

O modelo de chave simétrica é tido como o modelo convencional de criptografia (Stallings, 2008) e é adotado pelo sistema proposto nesse trabalho.

Esse modelo funciona de maneira tal que duas entidades são inicialmente definidas: emissor e receptor (que no sistema proposto tem o papel de cliente e servidor, respectivamente). As duas entidades devem ter conhecimento da chamada chave única, ou chave secreta, valor utilizado pelos algoritmos de criptografia e decriptografia para cifrar e recuperar o texto puro de uma mensagem.

Os eventos do modelo ocorrem na seguinte ordem:

1. Emissor aplica ao texto puro um algoritmo de criptografia (no caso desse trabalho, o *AES*) utilizando a chave única;

2. Emissor transmite a mensagem cifrada gerada para o receptor;

3. Receptor recebe a mensagem cifrada e aplica, usando a chave única, o mesmo algoritmo de criptografia executado de forma inversa a fim de obter o texto puro.

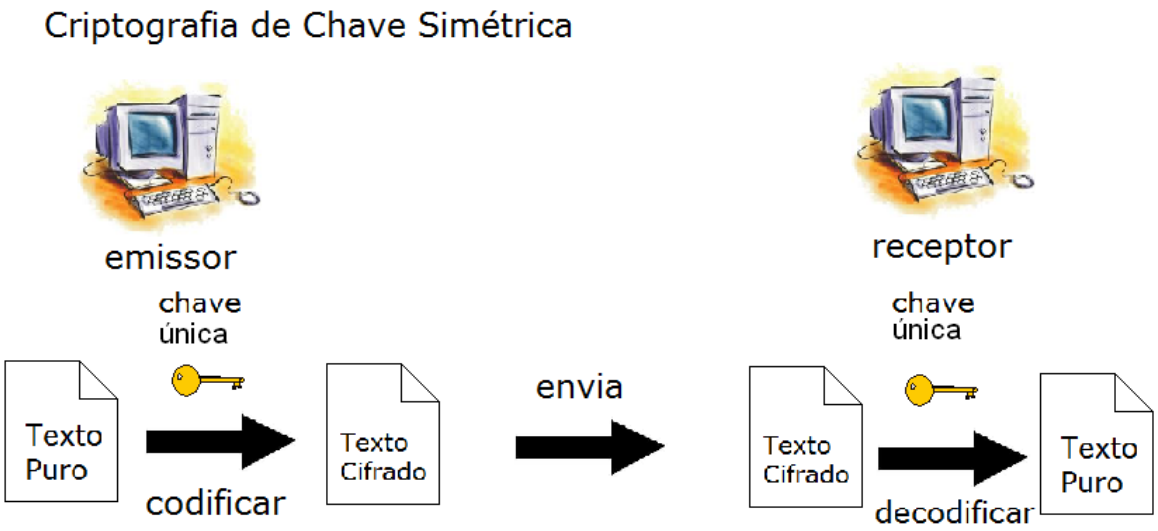


Figura 2.5: Criptografia de Chave Simétrica

De acordo com Stallings (2008), duas premissas básicas devem ser seguidas para a utilização do modelo de chave simétrica: o algoritmo de criptografia deve ser forte (requisito atendido pelo projeto visto que o mesmo utiliza o reconhecido padrão *AES* para realização das operações criptográficas), e emissor e receptor possuírem a chave única de maneira segura e mantendo-a protegida (critério também atendido pelo sistema dado que a chave é única, imutável e não circula na rede interna).

2.4.2 Criptoanálise

A intenção principal de um atacante ao tentar quebrar um sistema criptográfico é a obtenção da chave secreta atual, de maneira que não só a mensagem atual pode ser decifrada como todas as mensagens subsequentes que forem capturadas. Em Stallings (2008) diz-se que existem duas técnicas gerais a um sistema criptográfico de chave simétrica: a criptoanálise e o ataque por força bruta.

A criptoanálise consiste da utilização do algoritmo de criptografia, do texto cifrado e da análise de qualquer parte do texto puro disponível para tentar, ou decifrar o texto cifrado, ou recuperar a chave única (resultado primário pretendido). Já o ataque força bruta não faz análises sobre o texto cifrado ou o algoritmo, ele simplesmente testa

todas as chaves possíveis em um parte do texto cifrado até conseguir recuperar algum trecho de informação.

Caso algum dos ataques descritos acima obtenha êxito na obtenção da chave única, o sistema torna-se comprometido. A partir dessas informações nota-se a necessidade de utilização de um algoritmo que seja resistente a esses tipos de ataque.

2.4.3 Cifras de Fluxo e Cifras por Bloco

Em Stallings (2008) descreve-se que uma cifra de fluxo é tal que codifica e decodifica uma quantidade de dados sem tamanho específico caracter por caracter, diferentemente das cifras por bloco que possuem tamanho de texto de entrada específico (como por exemplo 16 bytes no caso do *AES*) e um bloco de texto de saída com tamanho correspondente à entrada (um bloco de 16 bytes de entrada gera outro bloco de 16 bytes de saída).

Muitas vezes, assim como o caso proposto nesse trabalho, informações a serem criptografadas por uma cifra de bloco não possuem o mesmo tamanho de entrada requisitado pelo algoritmo criptográfico. Para solucionar essa situação existem modos de operação de cifras de bloco que serão explicadas mais tarde, na seção 2.4.5, e que produzem o mesmo efeito gerado pelas cifras de fluxo, a saber, modo *ECB* e modo *CBC*.

2.4.4 *AES*

Segundo Stallings (2008), o padrão de criptografia avançado (*AES*) é uma cifra de bloco simétrica, publicada em 2001 pelo *NIST*, cujo objetivo é substituir o antigo *DES* como padrão aprovado para uma grande variedade de aplicações. Para escolher um algoritmo que representasse esse novo padrão, houve uma competição proposta pelo *NIST* e dentre os vários candidatos participantes, o algoritmo vencedor foi o Rijndael.

O *NIST* seguiu um protocolo de avaliação baseado nos seguintes critérios:

- Segurança: referente ao esforço computacional necessário pra criptoanalisar o algoritmo;
- Custo: refere-se a necessecidade de alta eficiência computacional do algoritmo dada a necessidade de poder usá-lo numa gama aplicações variadas incluindo as de alta

velocidade;

- Implementações em *Software*: diz respeito à velocidade na execução do algoritmo em diferentes plataformas e com variação no tamanho das chaves;
- Ambientes de espaço restrito: estudo sobre a possibilidade de execução do algoritmo em ambientes que rodam aplicações limitadas em recurso de memória RAM e ROM;
- Implementações em *Hardware*: assim como o critério supracitado, esse requisito se refere ao desempenho do algoritmo em espaços restritos e diz respeito ao estudo do impacto em *hardware* da variação do tamanho de código;
- Ataque nas implementações: referente ao estudo da estrutura do algoritmo e sua capacidade de defesa contra ataques de consumo de potência e temporização (a descrição desse tipo de ataque foge ao escopo do trabalho e não será realizada);
- Criptografia *versus* decriptografia: diz respeito ao estudo dos métodos de criptografia e decriptografia, se são diferentes ou não e o impacto de custo e temporização referentes ao tipo de implementação adotada;
- Agilidade da chave: diz respeito a capacidade e velocidade de alteração de chaves com utilização mínima de recursos;
- Outras versatilidades e flexibilidades: estudo referente à flexibilidade quanto ao tamanho de chaves e de bloco de entrada por exemplo;
- Potencial para paralelismo em nivel de instrução: referente à capacidade de aproveitamento de recursos de ILP (*instruction-level parallelism*) disponíveis pelos processadores *multi-core* atuais;

O Rijndael, ganhador do concurso para o *AES*, teve desempenho satisfatório em todos os critérios acima descritos, requeridos pelo *NIST*. Por esse fato, principalmente no que se refere ao funcionamento em ambientes de espaço restrito, o Rijndael/AES foi o algoritmo criptográfico escolhido para agregar segurança ao sistema proposto nesse trabalho, uma vez que no *middleware* implementado há restrições de processamento e armazenamento de dados.

2.4.4.1 Funcionamento

O Rijndael, de acordo com Stallings (2008), é um algoritmo que opera com tamanhos de chave e de bloco independentes e que podem ser definidos como de 128, 192 ou 256 bits, porém, a especificação do *AES* restringiu o tamanho de bloco para 128 bits, de maneira tal que todo o funcionamento descrito a partir de agora é dado seguinte essa propriedade.

A entrada do algoritmo é um bloco de 16 bytes representado por uma matriz quadrada 4 x 4. Ao iniciar o processo de criptografia, os valores desse bloco são copiados para um vetor ***State*** que será então modificado através de uma série de rodadas; ao final ele representará o texto de saída, seja ele o resultado de uma criptografia (texto cifrado) ou decriptografia (texto puro). A Figura 2.6 demonstra o processo em linhas gerais.

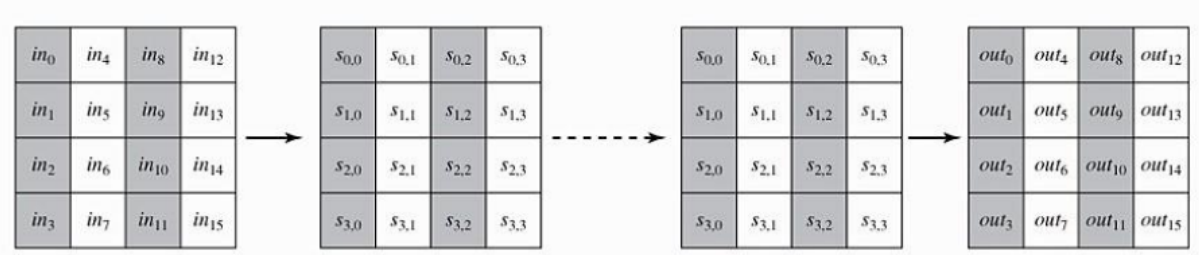


Figura 2.6: *Input*, vetor *State* e Saída. Fonte: (Stallings, 2008)

A estrutura de criptografia/decriptografia (para chave de tamanho 128 bits) é definida em 10 rodadas. Cada uma dessas rodadas utiliza quatro operações: ***SubBytes, ShiftRows, MixColumns e AddRoundKey***. Além dessas quatro operações (e de suas inversas para caso de decriptografia), o *AES* necessita necessita de uma função de extensão de chave que irá gerar, a partir da chave inicial, um vetor com 44 *words* que serão utilizadas na etapa ***AddRoundKey***. O vetor chave extendida é exemplificado na Figura 2.7.

Figura 2.7: Chave e Chave Extendida. Fonte: (Stallings, 2008)

As 4 operações são descritas a seguir:

- **Substituição de bytes ou *SubBytes***: realiza uma substituição byte a byte do bloco de entrada utilizando uma estrutura matricial 16 x 16 chamada caixa-S, que contém todos os 256 possíveis valores de 8 bits permutados. Tal substituição é feita da seguinte maneira: cada um dos 16 valores hexadecimais pertencentes ao bloco de entrada são mapeados em um outro valor da caixa-S, a posição correspondente para a realização dessa troca é dada de forma tal que linha e coluna são relacionados aos dígitos do hexadecimal, por exemplo, dado o seja o número 52^3 o alvo para realização da troca; dessa forma, a linha correspondente para substituição na caixa-S é "5"e a coluna é "2";

- **Deslocamento de linhas ou *ShiftRows***: realiza uma permutação no conteúdo do vetor ***State***, de tal forma que, a primeira linha se mantém inalterada, a segunda sofre um deslocamento cíclico de 1 byte à esquerda, a terceira um deslocamento 2 bytes à esquerda e a quarta 3 bytes à esquerda (ou um à direita);

- **Embaralhamento de colunas ou *MixColumns***: realiza uma substituição coluna a coluna de tal forma que cada byte de uma coluna é mapeado em um valor que é uma combinação linear de todos os bytes dessa mesma coluna;

- **Adição de chave da rodada ou *AddRoundKey***: realiza um XOR bit a bit no bloco atual com parte da chave extendida.

A representação das operações de **ShiftRows** e **MixColumns** são encontradas na Figura 2.8.

A Figura 2.9 exemplifica as ordens de aplicação das funções descritas para os processos de criptografia e decriptografia.

A Figura 2.10 representa uma sequências de cada operação descrita anteriormente sobre o vetor ***State*** em uma rodada do algoritmo.

[3] A notação entre chaves para representação de um valor hexadecimal é a adotada na *FIPS* PUB 197 e será a notação adotada nesse trabalho

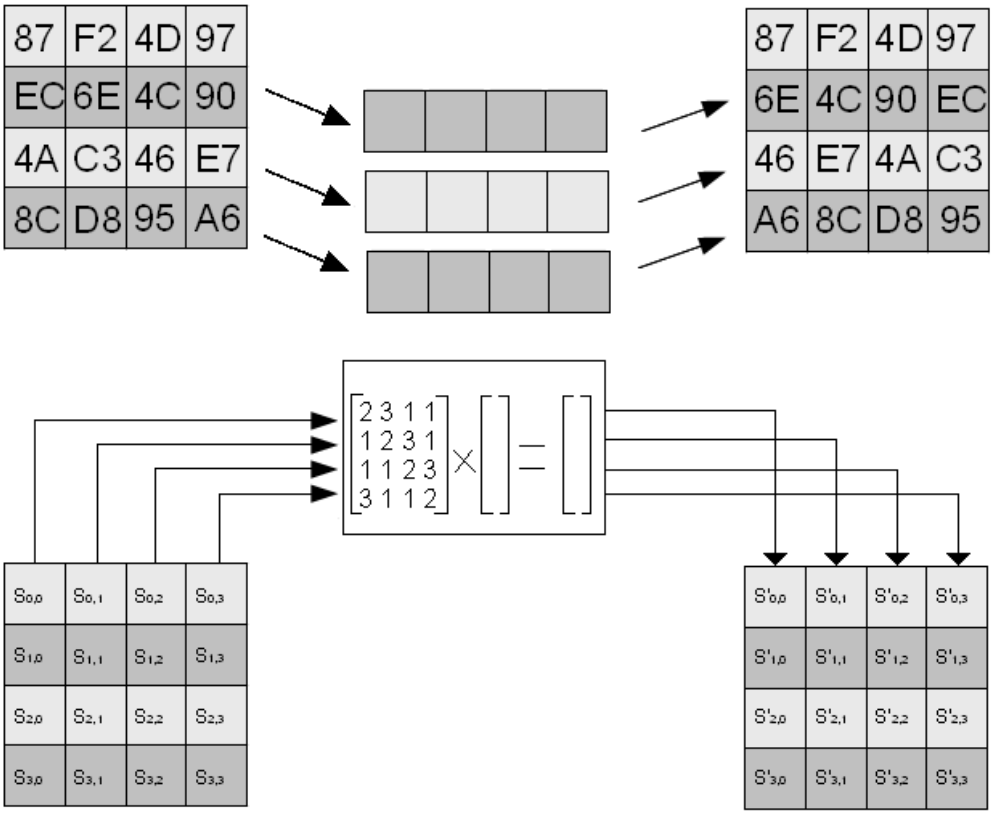


Figura 2.8: Operações ShiftRows e MixColumns. Fonte: (Stallings, 2008)

2.4.5 Modos de Operação de Cifra de Bloco

Como descrito em Stallings (2008), um algoritmo de cifra de bloco é um elemento básico para agregar segurança a um conjunto de dados, porém nem sempre esses algoritmos em sua forma pura são suficientes para suprir a necessidade de certas aplicações (como é o caso do *AES* em relação a esse trabalho). Dado este fato, foram definidos quatro modos de operação para cifras de bloco, publicados pelo *NIST* na *FIPS* 81, com intuito de adequar os algoritmos às aplicações. Para o sistema proposto nesse trabalho, foi necessária uma implementação com dois desses modos de operação em conjunto: o *Electronic CodeBook* e o *Cipher Block Chaining*, que serão descritos a seguir.

2.4.5.1 Modo *Electronic Codebook* (*ECB*)

É o modo mais simples de criptografia, faz o tratamento do texto puro um bloco de b bytes por vez e utilizando sempre a mesma chave. Para mensagens com tamanho maior que b

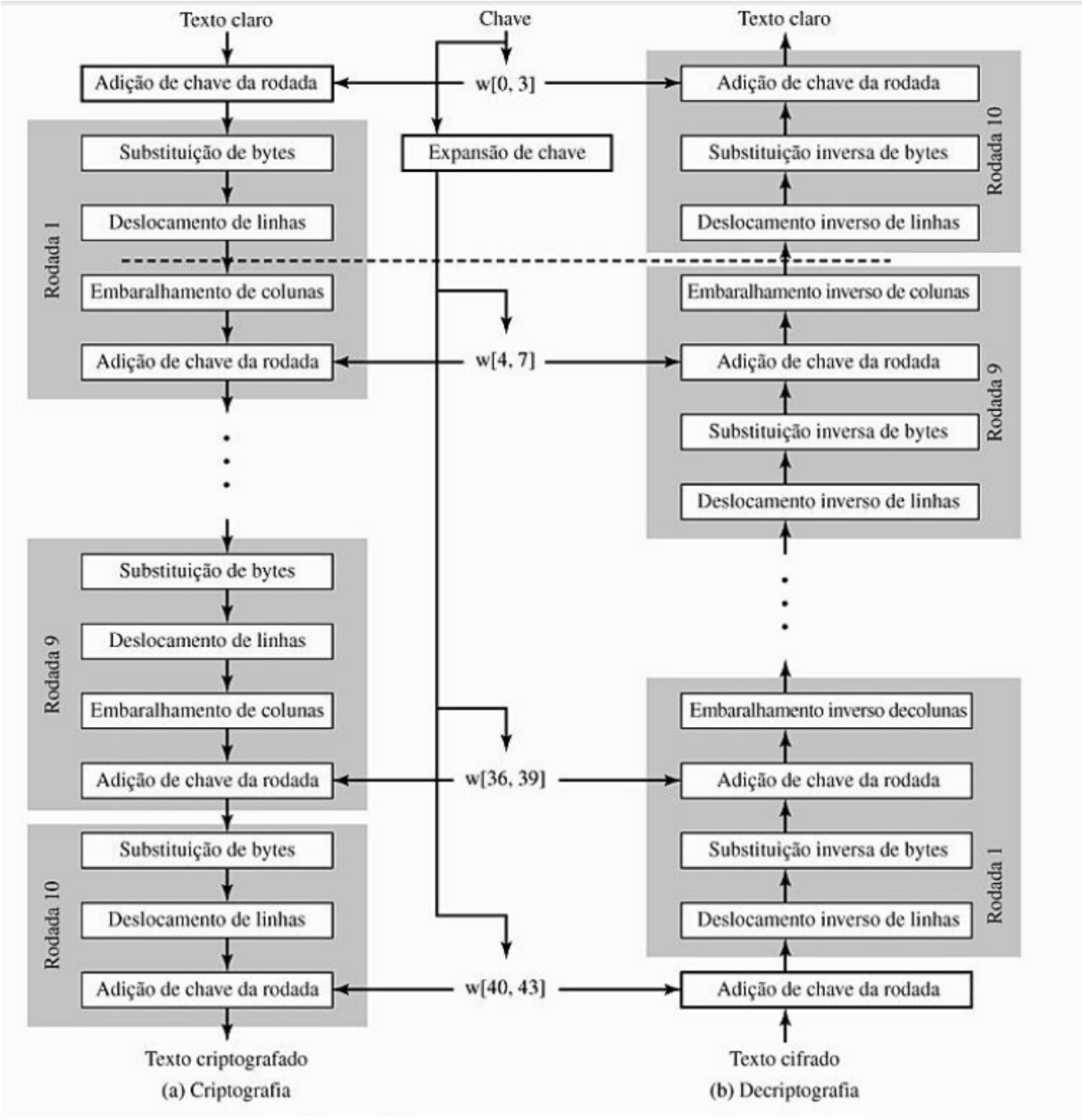


Figura 2.9: Fluxos de criptografia e decriptografia. Fonte: (Stallings, 2008)

bytes, o método as trata através da divisão da mensagem original em blocos de b bytes, caso o último bloco não contenha os b bytes necessários, realiza-se então um procedimento chamado *padding* que consiste em completar o bloco com informações arbitrárias até que o mesmo possua o tamanho adequado.

Um dos problemas encontrados nesse modo de criptografia é que mesmos blocos de texto puro sempre gerarão um mesmo texto cifrado, uma vez que a chave utilizada e única nesse processo. Tal fato gera problemas de segurança e deixa o código vulnerável à ataques criptoanalíticos uma vez que o atacante pode usufruir do padrão encontrado para tentar decifrar o texto.

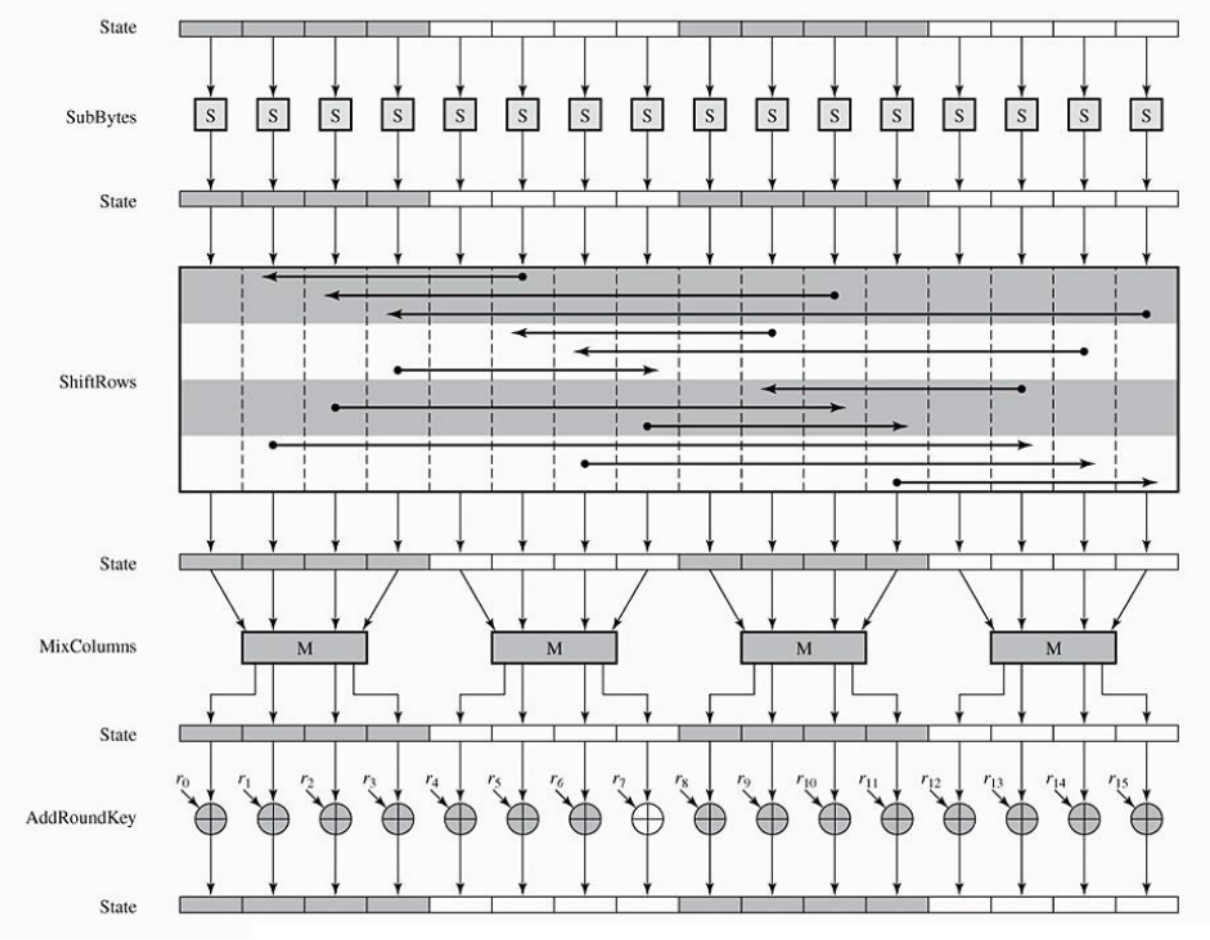


Figura 2.10: Rodada de Criptografia. Fonte: (Stallings, 2008)

A Figura 2.11 demonstra o funcionamento do modo *ECB*, K, P e C representam a chave, o texto puro (*plaintext*) e o texto cifrado(*ciphertext*), respectivamente.

2.4.5.2 Modo *Cipher Block Chaining*(*CBC*)

Com intuito de contornar as falhas de segurança do modo *ECB*, de maneira a garantir que textos puros, mesmo que reincidentes em uma mensagem, gerem textos cifrados diferentes, utiliza-se o método *CBC*. Esse modo funciona da seguinte maneira: ao invés de se entrar com um texto puro no algoritmo de criptografia, entra-se com um XOR desse texto com o texto cifrado anterior (ou dele com um vetor de inicialização (IV), caso seja o primeiro bloco criptografado de uma mensagem). Dessa forma, mesmo utilizando-se uma mesma chave para criptografia de blocos subsequentes, o texto cifrado gerado será diferente, ainda que possua texto puro repetido no corpo da mensagem.

A Figura 2.11 demonstra o funcionamento do modo *ECB*, P e C representam o texto puro (*plaintext*) e o texto cifrado(*ciphertext*), respectivamente.

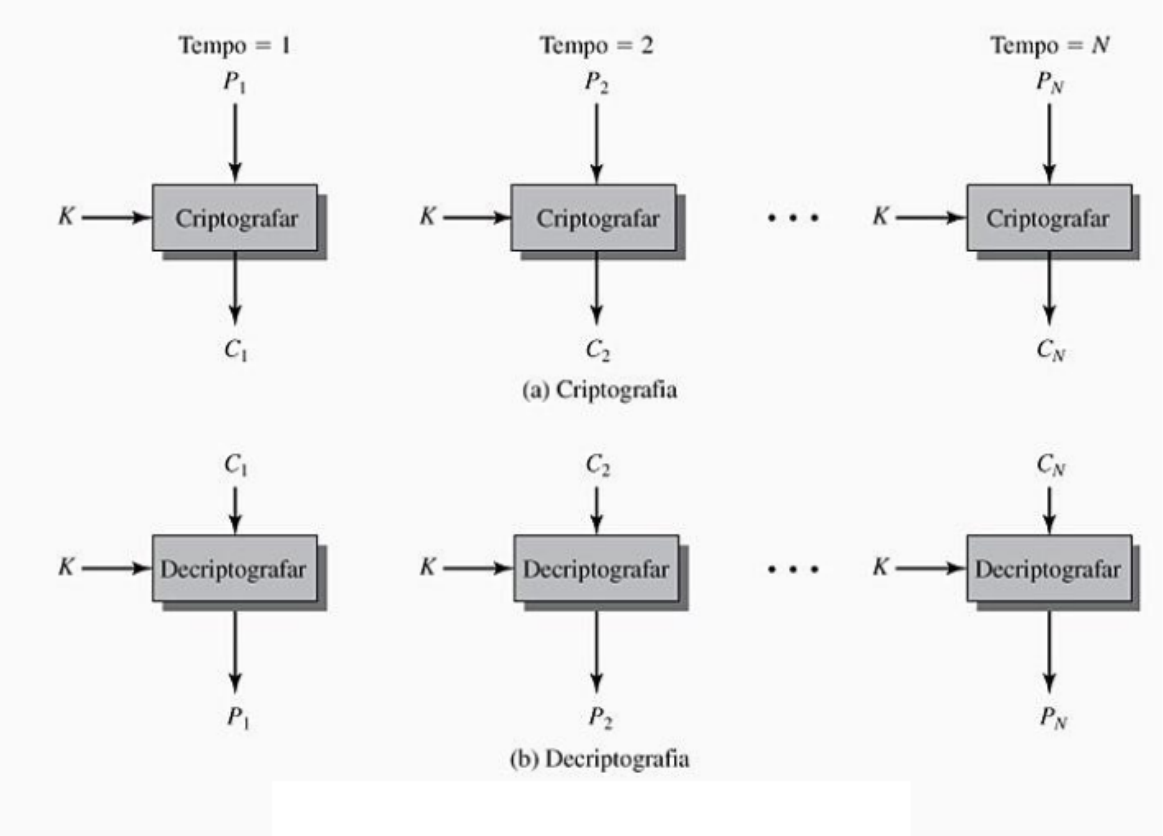


Figura 2.11: Modo *ECB*. Fonte: (Stallings, 2008)

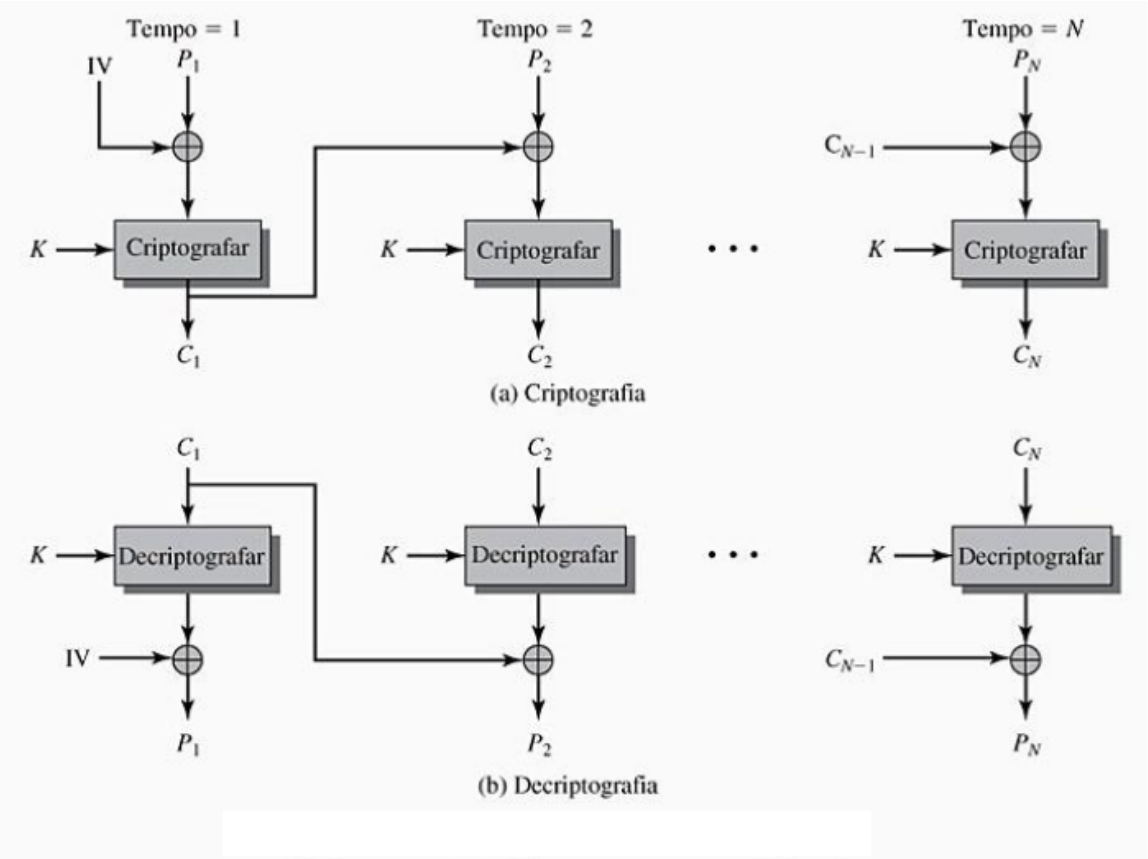


Figura 2.12: Modo *CBC*. Fonte: (Stallings, 2008)

3 Trabalhos Relacionados

Existem atualmente no mercado inúmeros sistemas que possuem propósito semelhante ao aqui apresentado. Por se tratarem de sistemas proprietários, por vezes patenteados, não é comum encontrar na bibliografia descrição detalhada de tais plataformas.

As diferenças entre os sistemas de controle de acesso se dão em vários quesitos: na complexidade do sistema, no propósito do mesmo, na estrutura de *hardware* e nos mecanismos de identificação.

Um sistema de controle de acesso centralizado via *RFID* é proposto por Peusaari et al. (2009). Neste sistema são utilizados dispositivos *RFID* passivos de 13.56MHz, e leitores de *RFID*, compatíveis com essa faixa de frequência. Estes leitores são conectados a computadores por interface *USB* e transmitem os dados de leitura aos mesmos. Semelhante ao trabalho aqui proposto, esse sistema utiliza integração com banco de dados MySQL e comunicação cliente-servidor através do protocolo *TCP* (*Transmission Control Protocol*). No entanto, o trabalho foca tão e somente na aplicação que irá integrar os sistemas *RFID* conectados a computadores em rede.

Outro trabalho interessante no quesito comparação de métodos de autenticação, é encontrado em Santos et al. (2007). Nesse trabalho é também desenvolvido um sistema de controle de acesso, sem segurança e que utiliza como meio de identificação a entrada de senha digitada em um teclado numérico.

O trabalho proposto por (Mooney et al., 2002), apresenta um sistema de controle de acesso muito parecido com o aqui proposto, porém, o propósito de utilização de seu sistema é a autenticação de usuários para utilização de um computador, diferentemente da autenticação para acesso à dependências proposto aqui nesse trabalho. O sistema de Mooney, auto-intitulado *"ACESS CONTROL / CRYPTO SYSTEM"*, utiliza *smart-cards*, tecnologia muito similar à de *RFID* utilizada aqui, para realizar a identificação de usuários e padrão *DES* para garantir a segurança dos dados, diferentemente do *AES* aqui utilizado.

Em Dong et al. (2006), é proposto um outro tipo de sistema de controle de acesso

duas vias através de *RFID*. Nesse trabalho o controle é dado através da identificação e detecção de pessoas ao passar por um portal dotado de um leitor de *RFID* para a identificação e de dois sensores infra-vermelhos para avaliar a direção em que se passou.

Um processo de identificação biométrico utilizando impressões digitais para autenticação de usuários é proposto por Burger (2001), nesse sistema é utilizado um *smart-card* contendo informações fisiológicas dos usuários armazenadas num *chip* e autenticação dessas informações através de um leitor de impressões digitais. Diferentemente do sistema aqui proposto, essa abordagem faz todo o processo de autenticação localmente, não requerendo consulta a um banco de dados externo e evitando assim a necessidade do uso de criptografia nas mensagens. Essa aplicação, não tem propósito específico (como controle de abertura de portas por exemplo) e dessa maneira se faz portável à qualquer sistema em que a autenticação de um usuário seja necessária.

Dentre os trabalhos de Santos et al. (2007) e Peusaari et al. (2009) mais parecidos com o aqui proposto, algumas diferenças podem ser notadas, tanto em questão de *hardware* quanto as relacionadas ao *software*.

Diferentemente de Peusaari et al. (2009) e de Santos et al. (2007), o enfoque deste trabalho se dá na construção de um sistema embarcado independente, que não necessita conexão direta com um computador para funcionar e que irá se comunicar diretamente com um servidor centralizado, via interface de rede. Além disto, o mesmo se preocupa com questões de segurança, e a confere ao sistema através do uso de criptografia *AES* para a troca de informações entre cliente e servidor, aspecto não levado em consideração nos outros trabalhos.

4 Ambiente

Neste capítulo são feitas as descrições do ambiente de desenvolvimento do trabalho, apresentação de características de infraestrutura, uma proposta de avaliação de custos de implementação do sistema e ainda as estruturas referentes ao desenvolvimento do *middleware* e do servidor de autenticação.

Foi construído um protótipo para realização de testes do sistema em um ambiente isolado. Desta forma a comunicação entre o dispositivo cliente e o servidor que recebe, trata a mensagem e retorna o aceite ou não de acesso, não sofrem interferências de variáveis externas.

Para implementação do sistema ativo, fora do protótipo, usa-se uma estrutura de redes *VLAN* para isolar a comunicação entre clientes com o servidor. Ao utilizá-la permite-se aumentar tanto a segurança do sistema quanto o desempenho do mesmo, uma vez que, dessa forma, ele não compartilha o canal de dados utilizado com outros usuários.

Uma infraestrutura também centralizada de *nobreaks* é necessária para o sistema; uma vez que o processo de abertura e fechamento de portas é elétrico, caso aja falta de energia no ICE, o sistema não pode falhar. Estudos referentes a maneira de implementação dessa estrutura ainda devem ser realizados.

Os equipamentos utilizados foram: o *middleware* composto por um *mbed*, uma interface de rede RJ-45, três *LEDs* coloridos (verde, amarelo e vermelho) e um leitor *RFID* (ID-12) (Innovations, 2005); um servidor com sistema operacional Ubuntu versão 12.04 com servidor *Web Apache* e servidor de banco de dados *MySQL*, 4Gb de memória *RAM* e processador Intel *Quad Core* Q8400; 2 cabos de rede do tipo *patch cord* categoria 6 de três metros de comprimento cada; e um *switch Extreme Networks Summit* X450e operando em 2 de suas portas a 10/100 Mbps. Além disto foram utilizados 50 cartões *RFID* de frequência 125Khz para os testes.

Para a abertura de portas, pode-se usar três possíveis variações: fechaduras elétricas, fechos elétricos e fechaduras de eletroímã. A Figura 4.1 mostra os três tipos de fechadura possíveis.

(a) Fechadura elétrica (b) Fecho elétrico

(c) Fechadura com eletroímã

Figura 4.1: Tipos de Fechadura

Para exemplificar os custos do projeto, para cada porta a ser aberta, tem-se o valor de R$370,00, incluindo todos os componentes(*mbed*, leitor *RFID*, controle de potência para abertura da fechadura), cabos, fontes de alimentação, bateria auxiliar de energia e fechadura eletromecânica de sobrepor. O sistema está preparado para controlar até duas portas com a mesma *mbed*, neste caso o custo para operar duas portas próximas chega a R$530,00. Isso permite uma redução de custo de aproximadamente 39,6% para uma segunda porta. Como é comum no cenário de implantação do sistema a ocorrência de salas próximas, e, consequentemente, portas próximas, haverá otimização no custo de implantação do sistema.

O custo do cartão *RFID* também deve ser considerado. Por exemplo, se duas portas tem uma fechadura do tipo tetra-chave, e são necesárias 60 pessoas com acesso a essas portas, 120 cópias devem ser feitas. Considerando um custo médio de R$10,00 por cópia de chave, um total de R$1200,00 seriam gastos nesse ambiente.

Com a implantação do sistema de controle acesso proposto, seria necessário um total de aproximadamente R$650,00 (considerando o custo do cartão *RFID* em torno de R$2,00) para o mesmo número de usuários. Para substituir o sistema de acesso em uma porta, há economia de R$550,00 no comparativo sistema de chaves *versus* sistema com cartão *RFID*. Se considerarmos ainda que esses usuários podem acessar outras portas,

como laboratórios, com o mesmo cartão, o custo do sistema fica claramente justificado.

4.1 *Middleware*

O *middleware* desenvolvido consiste em um sistema de autenticação centralizado via rede que libera ou não o acesso de usuários, através do recebimento de uma mensagem de aceite de acesso o mesmo envia um pulso para abertura da porta.

De maneira geral, toda vez que um usuário desejar usufruir dos recursos disponíveis de laboratórios, utilizar salas de aula, ou entrar em um departamento, o mesmo deverá aproximar seu cartão *RFID* de um leitor para que esse possa iniciar o processo de autenticação.

Ao contrário do trabalho de (Peusaari et al., 2009), onde são utilizados cartões de 13.56MHz, a frequência utilizada para o sistema aqui proposto é a de 125 kHz (tanto para os leitores quanto para as *tags*). O alcance real varia de 2,5cm a 4 cm, dependendo do tipo de cartão utilizado e de incidência ou não de obstáculos entre o cartão e o leitor (por exemplo uma carteira de bolso). Optou-se por essa faixa de frequência pelo fato de não haver necessidade de grande alcance de leitura, pela facilidade de encontrar cartões com esta frequência no mercado e pelo menor custo do leitor/cartão.

Além disto, por se tratar de um sistema de controle de acesso, não é desejável um grande alcance na leitura das *tags*. É fácil perceber que caso fosse utilizado um leitor de longo alcance (1m a 2m) ,por exemplo, poderia ocasionar a abertura de uma porta ou o acionamento indesejado de um dispositivo pelo simples fato de caminhar com um cartão no bolso por um corredor de salas, cenário que não se deseja obter dada a proposta de segurança do sistema.

Na construção do *middleware* foi utilizada a plataforma *mbed* com suporte para um ou dois leitores de *RFID* (ID-12 / ID-20), uma interface de rede *ethernet* com saída para conector padrão RJ-45 e ainda três *leds* coloridos (verde, vermelho e amarelo) os quais darão retorno ao usuário quando da interação com o sistema. As cores que representam o estado do sistema são apresentadas conforme a Tabela 4.1.

O Fluxograma da Figura 4.2 mostra a estrutura do processo que ocorre no *middleware*.

Tabela 4.1: Cores que representam o estado do sistema *RFID*

Cor	Estado	Mensagem
Amarelo	Piscando 1 vez	Realiza a Leitura do Cartão (*tag*)
Verde	Piscando 3 vezes	Acesso liberado
Vermelho	Piscando 3 vezes	Acesso negado
Vermelho e Verde	Piscando 3 vezes	Cartão não cadastrado no sistema

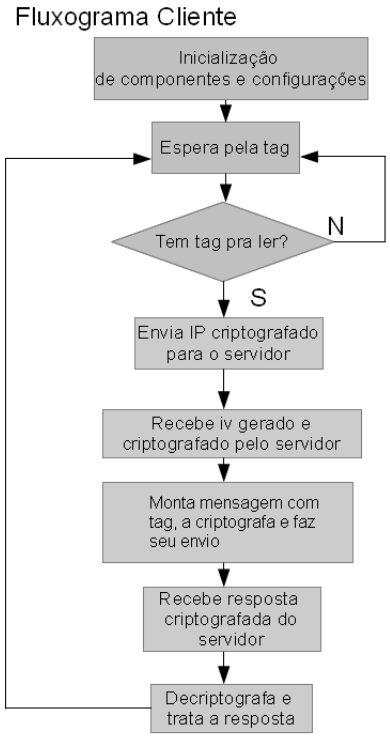


Figura 4.2: Fluxograma correspondente ao cliente na comunicação Cliente-Servidor

Os passos detalhados de cada etapa da execução do código no *middleware* são:

1. Ao iniciar o programa, faz-se toda a configuração necessária ao dispositivo através da leitura de um arquivo ".cfg"próprio para essa finalidade, além disso o programa extrai o identificador único da *mbed* e o atribui à chave que será utilizada para a criptografia *AES*;

2. Leitor *RFID* recebe uma *tag* (número referente ao cartão *RFID*);

3. O cliente então envia uma mensagem criptografada contendo seu número de *IP* ao servidor;

4. Cliente espera pelo retorno do servidor; uma mensagem criptografada contendo um outro vetor de inicialização gerado pelo servidor para a troca subsequente de mensagens;

5. Constrói-se uma mensagem para o servidor utilizando-se quatro parâmetros: o tipo de identificação que será feita (porta, chamada de classe, catraca do restaurante universitário)[4], o número da porta a qual se deseja obter acesso, o número correspondente ao método de autenticação (atualmente apenas via *RFID*) e a *tag* correspondente ao usuário;

6. Faz-se a inclusão de criptografia *AES* à mensagem (utilizando o identificador da *mbed* como chave, e o novo vetor de inicialização transmitido pelo servidor);

7. Envia-se a mensagem ao servidor via rede *ethernet*;

8. Espera-se a resposta do servidor, uma de três possíveis mensagens "OD", caso o usuário esteja liberado, "NA", caso o acesso não seja autorizado ou "NC", caso o cartão não esteja cadastrado;

9. A partir da resposta do servidor a porta será aberta (resposta "OD") ou permanecerá fechada (resposta "NA"ou "NC"). Em ambos os casos far-se-á um registro (*log*) do evento no banco de dados.

É importante ressaltar que a primeira mensagem é criptografa com o algoritmo *AES* usando um vetor de inicialização padrão conhecido pelo cliente e pelo servidor e com a chave do algoritmo sendo o identificador único de cada *mbed*, tal identificador está armazenado no banco de dados do sistema pareado com o número de *IP* atribuído de maneira fixa a cada *mbed*, dessa forma, ao receber uma conexão, o servidor faz uma consulta ao banco de dados e consegue descobrir qual chave deverá ser utilizada nos processos criptográficos. Além disso, vale dizer que a utilização de novos vetores de inicialização permite agregar ainda mais segurança ao sistema. Ao utilizar os modos de criptografia *ECB* em conjunto com o *CBC*, o sistema garante que não haverá repetição de texto cifrado para determinados trechos de texto puro dentro de uma mesma mensagem, porém, a priori não há uma garantia de que mensagens subsequentes, que possuam um

[4]Para o sistema descrito nesse trabalho, levar-se-á em consideração apenas a autenticação correspondente à portas. No capítulo 6, como trabalhos futuros, serão discutidos aspectos de implementação para controle de chamada de alunos em sala de aula, assim como o controle de catraca e de crédito a ser utilizado no restaurante universitário (RU).

mesmo texto puro, gerem textos cifrados diferentes. Para solucionar essa questão, utiliza-se uma atualização do vetor de inicialização cada vez que o cliente faça uma requisição para tentativa de acesso, agregando assim, mais segurança ao sistema.

A versão utilizada do *AES* (Rundgren, 2009) no sistema proposto por esse trabalho opera com conjunto de modos *ECB* e *CBC*, dessa forma, o ***AddRoundKey*** será feito sempre com a mesma chave inicial em conjunto com um XOR do bloco atual com o bloco cifrado anterior. O primeiro ***AddRoundKey*** realiza um XOR com um vetor de inicialização.

Antes da utilização da criptografia *AES*, houve uma tentativa de utilização de *SSL* no sistema, porém o conjunto de bibliotecas disponíveis no site do microcontrolador *mbed* (Holdings, 2011) para a comunicação via rede *ethernet* encontrava-se desatualizada e com funcionamento não-determinístico, de maneira tal que todas as bibliotecas a utilizavam deparavam-se com esse problema, inclusive a utilizada para *SSL* (CyaSSL, 2011), de tal maneira que foi necessária a busca por outro método de agregação de segurança ao sistema.

A Figura 4.3 mostra o esquema do *middleware* construído em um protótipo.

Figura 4.3: Protótipo do sistema utilizado para testes

4.2 Servidor de Autenticação

Conforme dito anteriormente, um servidor centralizador das informações foi utilizado para execução do projeto. O mesmo tem por finalidade receber via rede *ethernet* as mensagens

enviadas pelo *middleware* e tratá-la, ou seja, verificar se o usuário está ou não autorizado a acessar determinada porta no horário requisitado, registrar a tentativa de acesso (com sucesso ou não) no banco de dados do sistema e retornar a mensagem de aceite ou recusa ao sistema. Nesse momento, a implementação está baseada em linguagem C rodando sobre um servidor *Web Apache* e os dados são gravados em um banco de dados *MySQL*. Futuramente a proposta é que todo o controle seja feito pelo administrador no sistema iNtegra (iNtegra, 2010), que apoia a administração geral do instituto.

O Fluxograma da Figura 4.4 mostra a estrutura do funcionamento do código utilizado no servidor.

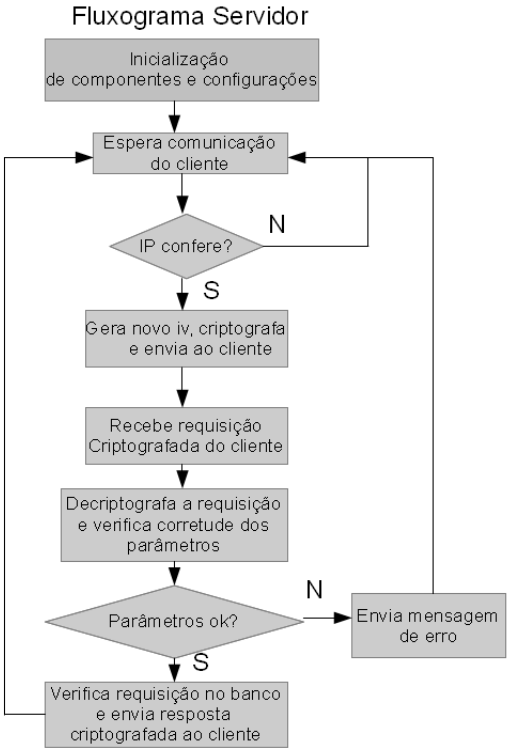


Figura 4.4: Fluxograma correspondente ao servidor na comunicação Cliente-Servidor

O processo detalhado da implementação em C no servidor pode ser descrito nas seguintes etapas:

- Servidor espera por uma conexão com o cliente;
- Servidor recebe uma mensagem criptografada do cliente contendo (teoricamente)[5] o *IP* do mesmo;

[5]Teoricamente uma vez que pode ser que o sistema esteja sendo atacado fazendo dessa forma com que o conteúdo da mensagem criptografada e o *IP* propriamente dito de quem a esteja enviando sejam diferentes

- Faz-se então consulta *SQL* pela chave de criptografia que deve ser usada na comunicação com o *IP* recebido, utiliza-se então a chave para decriptografar a mensagem recebida;

- Faz-se então uma comparação do texto com o *IP* recebido com o *IP* do cliente propriamente dito, caso a informação corresponda ao esperado, o processo continua normalmente, caso contrário, o servidor monta uma mensagem de erro e a envia criptografada ao cliente;

- Servidor gera um novo vetor de inicialização a ser usado nas próximas trocas de mensagens, o criptografa usando como chave o identificador recuperado através de consulta *SQL* e um vetor de inicialização padrão conhecido pelos dois lados da comunicação;

- Recebe-se então a mensagem referente à requisição de autenticação, faz-se a decriptografia da mesma e uma consulta *SQL* para obter a resposta esperada pelo cliente;

- Faz-se a criptografia da resposta e então envia-se a mesma ao cliente.

A Figura 4.5 representa a troca de mensagens em uma requisição.

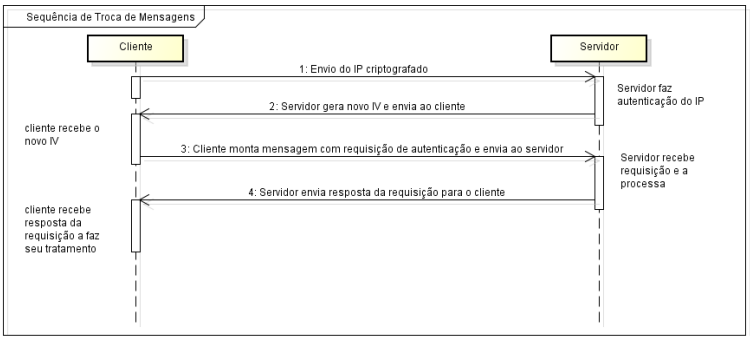


Figura 4.5: Sequência de mensagens de uma requisição na comunicação Cliente-Servidor

5 Resultados Experimentais

Para avaliar o *overhead* gerado pela inclusão da criptografia *AES* na comunicação cliente-servidor foram feitos testes do sistema com e sem o uso do processo criptográfico.

Para a bateria de testes foram levados em consideração dois cenários.

O primeiro cenário caracteriza-se pela execução do sistema no protótipo e comparação do tempo de resposta do sistema a partir da coleta de dados da implementação dos códigos do *middleware* e do servidor com e sem *AES*. Ele tem por objetivo fazer cálculo do *overhead* de tempo gerado pela utilização do *AES*.

Já o segundo cenário caracteriza-se pela comparação da quantidade de dados transmitida em cada requisição e subsequente cálculo do *overhead* gerado em relação ao tamanho (em bytes) de informação útil transmitida.

5.1 Cenário 1

No cenário 1, foram coletados dados na execução do sistema com e sem *AES*. Foram utilizados 50 (cinquenta) cartões, passados sequencialmente em um mesmo leitor com intervalo de 0.5 (meio) segundo entre uma leitura e outra.

Para a execução do sistema sem *AES* temos a Tabela 5.1 com os tempos obtidos:

Tabela 5.1: Cenário 1: Tempos de resposta (em milissegundos) para o sistema sem *AES*.

	Número de Cartões Testados				
Tipo de Retorno	10	20	30	40	50
NA	3,0996	3,1829	2,9278	2,8845	2,8741
NC	1,7060	1,6600	1,6035	1,6258	1,6359
OD	2,7596	2,6960	2,5649	2,5348	2,4843
Misto	4,0019	3,9886	3,8693	3,8614	3,8573
Média	2,8917	2,8818	2,7394	2,7266	2,7129

Para a execução do sistema com *AES* temos a Tabela 5.2 com os tempos obtidos:

A Tabela 5.3 representa o percentual de aumento de tempo do sistema com *AES* em relação ao sistema sem *AES*:

Os resultados obtidos na Tabela 5.3 mostram um aumento percentual significativo

Tabela 5.2: Cenário 1: Tempos de resposta (em milissegundos) para o sistema com *AES*.

	Número de Cartões Testados				
Tipo de Retorno	10	20	30	40	50
NA	4,2100	4,1191	4,1866	4,1829	3,9724
NC	2,3423	2,3866	2,3399	2,3455	2,3412
OD	4,3992	3,8881	3,9833	4,0832	4,0821
Misto	4,7617	4,3082	4,1270	3,9061	3,8766
Média	3,9283	3,6755	3,6592	3,6294	3,5680

Tabela 5.3: Cenário 1: Percentual de aumento de tempo entre os sistemas com e sem *AES*.

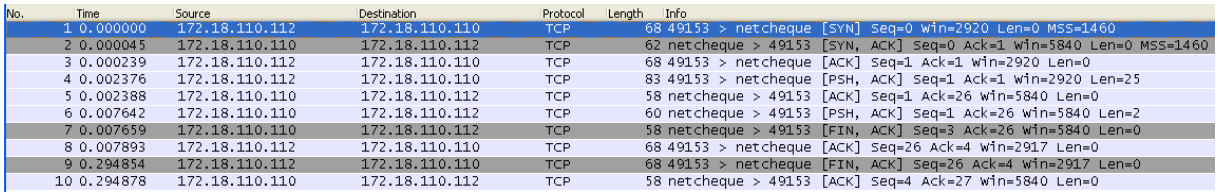
	Número de Cartões Testados				
Tipo de Retorno	10	20	30	40	50
NA	35,82%	29,41%	42,99%	45,01%	38,21%
NC	37,30%	43,77%	45,92%	44,27%	43,11%
OD	59,41%	44,22%	55,30%	61,09%	64,32%
Misto	18,99%	8,01%	6,66%	1,16%	0,50%
Média	37,88%	31,35%	37,71%	37,88%	36,53%

no tempo de resposta do sistema com *AES* em relação ao sistema sem *AES*. Contudo, ao se levar em consideração que o aumento de tempo de resposta não é sensível ao sistema (uma vez que a alteração está na faixa dos milisegundos e o sistema não requer velocidades maiores para funcionamento adequado), pode-se, por essa análise, inferir que a utilização do *AES* para agregação de segurança ao sistema é um recurso válido.

5.2 Cenário 2

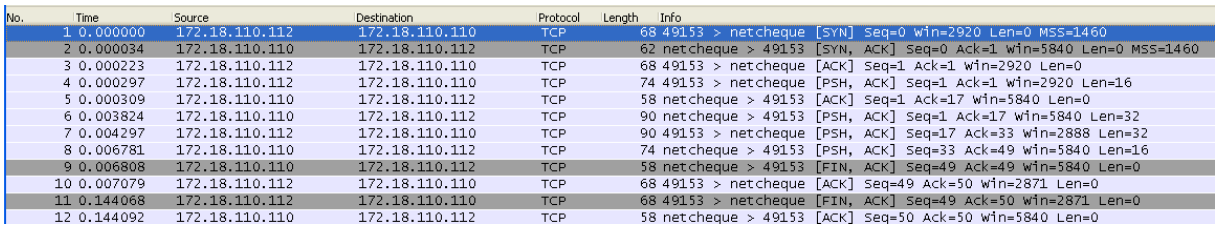
A coleta de dados para o cenário 2 foi realizada através da utilização do *sniffer* de rede *Wireshark* (Combs et al., 1998). As Figuras 5.1 e 5.2 mostram, respectivamente, a captura de uma requisição do cliente sem, e com a utilização do *AES* (no total foram capturadas 50 requisições, mas uma vez que todas possuem o mesmo formato e trafegam a mesma quantidade de dados, so se faz necessária a análise de uma delas para cada tipo de requisição).

As Tabelas 5.4 e 5.5 descrevem as mensagens trocadas na comunicação e seus tamanhos. Ao analisar seu conteúdo, podemos constatar que oito mensagens se mantém constantes, as três primeiras do *three way handshake* do *TCP*, a quinta mensagem de reconhecimento de conexão (*ACK*) e as quatro últimas mensagens de finalização de co-



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.18.110.112	172.18.110.110	TCP	68	49153 > netcheque [SYN] Seq=0 Win=2920 Len=0 MSS=1460
2	0.000045	172.18.110.110	172.18.110.112	TCP	62	netcheque > 49153 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
3	0.000239	172.18.110.112	172.18.110.110	TCP	68	49153 > netcheque [ACK] Seq=1 Ack=1 Win=2920 Len=0
4	0.002376	172.18.110.112	172.18.110.110	TCP	83	49153 > netcheque [PSH, ACK] Seq=1 Ack=1 Win=2920 Len=25
5	0.002388	172.18.110.110	172.18.110.112	TCP	58	netcheque > 49153 [ACK] Seq=1 Ack=26 Win=5840 Len=0
6	0.007642	172.18.110.110	172.18.110.112	TCP	60	netcheque > 49153 [PSH, ACK] Seq=1 Ack=26 Win=5840 Len=2
7	0.007659	172.18.110.110	172.18.110.112	TCP	58	netcheque > 49153 [FIN, ACK] Seq=3 Ack=26 Win=5840 Len=0
8	0.007893	172.18.110.112	172.18.110.110	TCP	68	49153 > netcheque [ACK] Seq=26 Ack=4 Win=2917 Len=0
9	0.294854	172.18.110.112	172.18.110.110	TCP	68	49153 > netcheque [FIN, ACK] Seq=26 Ack=4 Win=2917 Len=0
10	0.294878	172.18.110.110	172.18.110.112	TCP	58	netcheque > 49153 [ACK] Seq=4 Ack=27 Win=5840 Len=0

Figura 5.1: Imagem do Wireshark para o tráfego de uma requisição sem utilização do *AES*



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.18.110.112	172.18.110.110	TCP	68	49153 > netcheque [SYN] Seq=0 Win=2920 Len=0 MSS=1460
2	0.000034	172.18.110.110	172.18.110.112	TCP	62	netcheque > 49153 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
3	0.000223	172.18.110.112	172.18.110.110	TCP	68	49153 > netcheque [ACK] Seq=1 Ack=1 Win=2920 Len=0
4	0.000297	172.18.110.112	172.18.110.110	TCP	74	49153 > netcheque [PSH, ACK] Seq=1 Ack=1 Win=2920 Len=16
5	0.000309	172.18.110.110	172.18.110.112	TCP	58	netcheque > 49153 [ACK] Seq=1 Ack=17 Win=5840 Len=0
6	0.003824	172.18.110.110	172.18.110.112	TCP	90	netcheque > 49153 [PSH, ACK] Seq=1 Ack=17 Win=5840 Len=32
7	0.004297	172.18.110.112	172.18.110.110	TCP	90	49153 > netcheque [PSH, ACK] Seq=17 Ack=33 Win=2888 Len=32
8	0.006781	172.18.110.110	172.18.110.112	TCP	74	netcheque > 49153 [PSH, ACK] Seq=33 Ack=49 Win=5840 Len=16
9	0.006808	172.18.110.110	172.18.110.112	TCP	58	netcheque > 49153 [FIN, ACK] Seq=49 Ack=49 Win=5840 Len=0
10	0.007079	172.18.110.112	172.18.110.110	TCP	68	49153 > netcheque [ACK] Seq=49 Ack=50 Win=2871 Len=0
11	0.144068	172.18.110.112	172.18.110.110	TCP	68	49153 > netcheque [FIN, ACK] Seq=49 Ack=50 Win=2871 Len=0
12	0.144092	172.18.110.110	172.18.110.112	TCP	58	netcheque > 49153 [ACK] Seq=50 Ack=50 Win=5840 Len=0

Figura 5.2: Imagem do Wireshark para o tráfego de uma requisição utilizando *AES*

nexão. As outras mensagens, que possuem alteração de tamanho, são o alvo da análise para cálculo do *overhead* de dados trafegados ao utilizar criptografia *AES* no sistema.

Tabela 5.4: Cenário 2: Análise dos dados trafegados em um requisição sem *AES*.

Tamanho em Bytes	Descrição
68 62 68	*TCP three way handshake*
83	envio do cliente com requisição da *tag* (25 bytes de dados)
58	*ACK*
60	resposta do servidor (2 bytes de dados)
58 68 68 58	fim da conexão
total 651 bytes	

Ao analisar as supracitadas tabelas, podemos verificar que:

- Em uma requisição sem *AES*, encontrada na Tabela 5.4, duas mensagens de dados são trocadas, uma correspondente à requisição de autenticação da *tag* pelo cliente, e a resposta enviada pelo servidor;
- Já na requisição com *AES*, encontrada na Tabela 5.5, temos a troca de quatro mensagens de dados, o envio do *IP* criptografado, a resposta do servidor contendo o novo *IV* e em seguida as mesmas mensagens trocadas na requisição sem *AES* (pedido de autenticação e resposta do servidor).

Tabela 5.5: Cenário 2: Análise dos dados trafegados em um requisição com *AES*.

Tamanho em Bytes	Descrição
68 62 68	*TCP three way handshake*
74	envio *IP* criptografado (16 bytes de dados)
58	*ACK*
90 90 74	envio do servidor com novo *IV* gerado (32 bytes de dados) envio do cliente com requisição da *tag* (32 bytes de dados) resposta do servidor (16 de dados)
58 68 68 58	fim da conexão
total 836 bytes	

Dessa forma, já temos duas mensagens extras na comunicação ao utilizar *AES*, o que já gera um *overhead* de 164 bytes na quantidade total de bytes trafegados.

Analisando as outras duas mensagens, que possuem a mesma função nas duas comunicações, obeservamos que:

- Na comunicação sem *AES*, 83 bytes são enviados para a requisição de autenticação, desses 83 bytes, 25 correspondem à informação útil transmitida;
- Ainda na comunicação sem *AES*, 60 bytes são enviados para a resposta do servidor, desses 60 bytes, 2 correspondem à informação útil transmitida;
- Na comunicação com *AES*, 90 bytes são enviados para a requisição de autenticação, desses 90 bytes, 32 correspondem à informação útil transmitida;
- Também na comunicação com *AES*, outros 74 bytes são enviados para a resposta do servidor, desses 74 bytes, 16 correspondem à informação útil transmitida;

Para analisar a continuação do cálculo do *overhead*, leva-se em consideração apenas o aumento da quantidade de informação útil transmitida em cada mensagem, dessa forma, temos um aumento de 21 bytes (7 bytes da requisição, diferença dos 32 bytes com *AES* contra os 25 sem *AES*; mais 14 bytes de resposta, diferença dos 16 bytes com *AES* contra 2 bytes sem ele).

Somando os 164 bytes das mensagens extras, com os 21 bytes do aumento de informação útil nas mensagens de requisição e resposta, temos um total de 185 bytes de *overhead* de dados transmitidos na comunicação, que representa um aumento de 25,19% na quantidade de dados envidos. Tal valor pode ser também obtido ao se subtrair da quantidade total de dados enviados com *AES* (836 bytes), o total de bytes enviados sem *AES* (651 bytes).

Apesar do que aparenta, esse aumento percentual de 25,19% na quantidade dados trafegados não é significativo a ponto de influenciar no desempenho do sistema de tal forma que também no quesito volume de dados, o *AES* também tem seu uso justificado.

6 Considerações Finais

A utilização de sistemas embarcados tem sido amplamente realizada no dia-a-dia. Isso faz com que novas técnicas de otimização dos sistemas sejam utilizadas. O controle de acesso é um mecanismo que deve ser tratado com cuidado em qualquer instituição, balanceando o custo de implantação com o benefício de assegurar o patrimônio institucional.

Nesse contexto, esse trabalho apresenta uma de implementação de um sistema de controle de acesso centralizado com *tags RFID*, de baixo custo, onde todo o controle de acesso às salas, laboratórios e departamentos do ICE da UFJF utilizarão este sistema. O trabalho proposto encontra-se em versão *beta* disponível para uso. Também está sendo desenvolvido e aprimorado o sistema de controle por parte do servidor *web*, visando uma flexibilidade maior no gerenciamento do ambiente.

Após a análise dos resultados, constatou-se que mesmo gerando *overhead* tanto em relação ao tempo quanto ao volume do fluxo de dados, o uso do *AES* é válido pelo fato de agregar segurança ao sistema impactando pouco no tempo de transferência das informações no ambiente. O sistema pode, ainda, ser utilizado em qualquer tipo de rede local, mesmo sem isolamento e/ou segurança.

Como trabalhos futuros, novos mecanismos de identificação e entrada de dados serão usados, tais como: um teclado numérico para acesso por senhas, um leitor biométrico de impressões digitais e uma microcâmera para identificação de tentativas de acesso não autorizado. Múltiplos mecanismos combinados ao *middleware* poderão ser utilizados, aumentando ainda mais a segurança no ambiente. Um exemplo disso é utilização de criptografia via *hardware* ao se utilizar o *chip* ATSHA204 (ATMEL, 2013).

O sistema poderá ser utilizado ainda, quando disposto na entrada de salas de aula, para, por exemplo, controle de frequência de alunos no decorrer do período letivo. Ou como meio de se substituir o uso de *tickets* no restaurante universitário pelo cartão que armazenaria o crédito de cada usuário; ou ainda como alternativa para controle de empréstimo de livros na biblioteca.

Para o servidor, uma implementação de uma *interface* de gerenciamento do sis-

tema está sendo desenvolvida, onde grupos de usuários serão definidos, sendo possível a administração compartilhada entre diversos gerentes de setor. Além disso ainda é necessário um estudo relativo ao que fazer quando houver queda do servidor, ao cálculo da quantidade simultânea de acessos suportada pelo sistema.

Referências Bibliográficas

ATMEL. **Atmel atsha204**. http://dlnmh9ip6v2uc.cloudfront.net/datasheets/BreakoutBoards/Atmel-8740-CryptoAuth-ATSHA204-Datasheet.pdf, 2013. [Online; acesso em Fevereiro de 2013].

ARDUINO. **Arduino**. http://www.arduino.cc/, 2012. [Online; acesso em Fevereiro de 2013].

BURGER, P. M. **Biometric authentication system**, Abr. 17 2001. US Patent 6,219,439.

COMBS, G. **Wireshark**. http://www.wireshark.org/, 1998. [Online; acesso em Fevereiro de 2013].

OUSKA, T. **Cyassl for mbed**. https://mbed.org/users/toddouska/code/CyaSSL, 2011. [Online; acesso em Fevereiro de 2013].

DONG, Y.-S.; Zuo, B.-R.; Mackenzie, P. D. ; Salvo, J. J. **Analysis of state transition diagrams for rfid-based two-way access control**. In: Cybernetics and Intelligent Systems, 2006 IEEE Conference on, p. 1–4. IEEE, 2006.

HOLDINGS, A. **Arm**. http://www.arm.com/, 1983. [Online; acesso em Fevereiro de 2013].

HOLDINGS, A. Rapid prototyping for microcontrollers. **Cambridge, UK.[Online] Available: http://www.mbed.org**, 2011.

INNOVATIONS. **Innovtions id-12**. https://www.sparkfun.com/datasheets/Sensors/ID-12-Datasheet.pdf, 2005. [Online; acesso em Fevereiro de 2013].

JUELS, A. Rfid security and privacy: a research survey. **Selected Areas in Communications, IEEE Journal on**, v.24, n.2, p. 381 – 394, feb. 2006.

LAHIRI, S. **RFID sourcebook**. IBM press, 2005.

LANDT, J. The history of rfid. **Potentials, IEEE**, v.24, n.4, p. 8–11, 2005.

MOONEY, D. M.; Kimlinger, P. J. ; Bradley, J. V. **Access control/crypto system**, Fev. 26 2002. US Patent 6,351,813.

PEUSAARI, J.; Kelkka, R. ; Ikonen, J. **An access control and time management software solution using rfid**. In: Proceedings of the International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing, p. 50. ACM, 2009.

ROBERTS, C. Radio frequency identification (rfid). **Computers amp; Security**, v.25, n.1, p. 18 – 26, 2006.

RUNDGREN, A. **Crypto**. https://mbed.org/users/andersrundgren/code/crypto/, 2009. [Online; acesso em Fevereiro de 2013].

dos SANTOS, D.; de OLIVEIRA JUNIOR, J.; LOPES, J. ; SANTOS, K. Um sistema de controle de acesso utilizando: Apis javacomm e jdbc, microcontrolador pic e banco de dados relacional. 2007.

SEN, D., S. P.; Das, A. **RFID for energy and utility industries**. Pennwell Corp, 2009.

STALLINGS, W. **Criptografia e Segurança de Redes**. Pearson Prentice Hall, 2008.

UKKONEN, L.; Sydanheimo, L. ; Kivikoski, M. **Read range performance comparison of compact reader antennas for a handheld uhf rfid reader**. In: RFID, 2007. IEEE International Conference on, p. 63–70. IEEE, 2007.

VAHID, F.; Givargis, T. **Embedded system design: a unified hardware/software introduction**. Wiley, 2002.

WEIS, S. A. Rfid (radio frequency identification): Principles and applications. 2007.

INTEGRA. **Sistema integra**. `http://integra.ice.ufjf.br/integra`, 2010. [Online; acesso em Fevereiro de 2013].