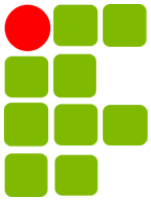


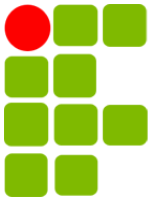
Instituto Federal de Educação, Ciência e Tecnologia da Paraíba
Campus Campina Grande
Coordenação do Curso Superior de Tecnologia em Telemática

Método para controle de acesso físico usando QR-Code e criptografia assimétrica

Dunfrey Pires Aragão

Orientador: Henrique do Nascimento Cunha

Campina Grande, novembro de 2015



Instituto Federal de Educação, Ciência e Tecnologia da Paraíba
Campus Campina Grande
Coordenação do Cursos Superior de Tecnologia em Telemática

Método para controle de acesso físico usando QR-Code e criptografia assimétrica

Dunfrey Pires Aragão

Monografia a ser apresentada à Coordenação do Curso de Telemática do IFPB - Campus Campina Grande, como requisito parcial para conclusão do curso de Tecnologia em Telemática.

Orientador: Henrique do Nascimento Cunha

Campina Grande, novembro de 2015

Método para controle de acesso físico usando QR-Code e criptografia assimétrica

Dunfrey Pires Aragão

Henrique do Nascimento Cunha
Orientador

Rhavy Maia Guedes
Membro da Banca

George Sobral Silveira
Membro da Banca

Campina Grande, Paraíba, Brasil
Novembro/2015

Agradecimentos

Agradeço a minha família que tanto acredita em mim e me ama de forma incondicional. A minha companheira Jhéssika Ângell, que me faz feliz e que me ajudou nesta caminhada universitária. A meu amigo Daniel Adônis, e aos professores que tive o prazer de conhecer: Henrique Cunha e Jerônimo Rocha, além da professora Daniella Dias, que tornaram minha vida acadêmica mais agradável e me ajudaram transmitindo conhecimento. Aos colegas de curso, que tornaram-se amigos nesta caminhada: Esdras, Anderson, Saulo, Teles e Camila. Além de todos que diretamente ou indiretamente me ajudaram nesta conquista.

Resumo

Em diversas ocasiões da vida cotidiana é necessário o sigilo ou proteção de informações, como também manter o controle de acesso físico de determinados ambientes, com objetivo de assegurar a ordem local, permitindo que apenas pessoas autorizadas sejam admitidas. Existem inúmeros sistemas de segurança, porém seus custos geralmente são elevados ou não podem garantir a segurança desejada e a alta velocidade de autenticação. Neste trabalho foi proposto a criação de um sistema de controle de acesso físico, utilizando a tecnologia *QR-Code* com criptografia assimétrica, baseando-se em pesquisas sobre as soluções existentes e os variados métodos adotados para autenticação, explorando as vulnerabilidades. O sistema criado fornece autenticidade, portabilidade e baixo custo, além de segurança e confidencialidade.

Palavras-chave: Criptografia assimétrica, QR-Code, Segurança, Controle de acesso.

Abstract

On several occasions of everyday life is necessary confidentiality and information protection, as well as maintain physical access control certain environments, in order to ensure the local order, allowing only authorized persons are admitted. There are numerous security systems, but their costs are generally high or can not ensure the desired safety and high-speed authentication. In this work we proposed the creation of a physical access control system using the *QR-Code* technology with asymmetric cryptography, based on research on existing solutions and the various methods used for authentication by exploiting vulnerabilities. The system created provides authenticity, portability and low cost, as well as security and confidentiality.

Keywords: Asymmetric encryption, QR-Code, Security, Access Control.

Sumário

Lista de Abreviaturas

CAF	*Controle de Acesso Físico*
QR-Code	*Quick Response Code*
RSA	*Criptossistema de Chave Pública Criado por Rivest, Shamir e Adleman*
PU	*Chave Pública*
PK	*Chave Privada*
JPA	*Java Persistence API*
JSF	*JavaServer Faces*
UE	*Usuário Externo*
ADM	*Administrador do Sistema*
OTP	*One-Time Password*

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A ideia de haver um mecanismo formal para controle de acesso não é recente. Há relatos de que na corte imperial chinesa (há 3500 anos), onde nem todos os membros da corte poderiam acessar todas as áreas do palácio. Para acessar os locais, eram dados anéis de identificação que deveriam ser mostrados aos guardas de cada área para que o acesso fosse permitido [1].

O controle para o acesso físico é importante em qualquer instituição. Dentre as quais podemos citar: indústrias (de pequeno, médio e grande porte), condomínios, instituições governamentais, escolas, hospitais, shopping centers, bancos.

Há grande interesse da comunidade científica e da indústria em desenvolver novos métodos de controle de acesso que apresentem sempre um melhor desempenho.

Alguns trabalhos encontrados visam o controle de acesso a informação, como, por exemplo, o trabalho de Hoving [2] que provê um método de escolha e implantação de sistemas de controle de acesso para proteção de sistemas computacionais. É também comum encontrar trabalhos que necessitam de um esquema robusto de autenticação. O trabalho de Hans [3] por exemplo, introduz um sistema de biometria para acesso a veículos de shopping centers na África do Sul. O autor coloca como grande vantagem dessa abordagem, o baixo custo e o alto fator de proteção aos veículos. Há ainda trabalhos que buscam melhorar a autenticação em sistemas industriais, como o trabalho de Dzung [4].

É possível observar em trabalhos que abordam o controle de acesso físico vulnerabilidades comuns entre eles. Por eminencia, podemos citar o alto custo para desenvolvimento de equipamentos. Um exemplo são os *smartcards*, cujo investimento é de alto valor monetário para produção dos cartões e os equipamentos destinados a leitura de informações também costumam ser caros. O mesmo se aplica a gastos para utilização do método *RFID*. Ademais, o *RFID* tipicamente não utiliza criptografia, como citado por Lehtonen [5], portanto seus dados estão vulneráveis, e de fácil acesso a um atacante. Um outro caso são os métodos que utilizam senhas; Kowtko [6] argumenta que pessoas idosas tendem a apresentar dificuldades em sistemas de autenticação, pois podem esquecer senhas facilmente. Por fim, podemos citar o uso de biometria. Esta apresenta certos problemas em pessoas idosas devido a grande incidência de patologias que dificultam a autenticação (catarata, derrame cerebral, insuficiência

cardíaca congestiva).

Na análise, para criação do sistema proposto neste trabalho, denotaremos o sistema por *Code Control*, deve ser considerado a necessidade de ser modularizado em vários níveis e etapas, para que, o usuário que o utilizar posso escolher quais alternativas se encaixam melhor em seu negócio. Como exemplo é utilizar senhas e/ou adotar o método de reconhecimento biométrico. Entretanto, quanto mais recursos o sistema for composto, maior será a complexidade na implantação.

Além da modularização, para assegurar a proteção das informações existentes no sistema, foi utilizado o método da criptografia dos dados via *software*, mas poderia ser substituído por criptografia via hardware, através da utilização de um componente externo, como um dispositivo USB, que serviria como uma chave PU ou PK do criptossistema.

Sendo assim, neste trabalho foi proposto um método de controle de acesso físico, onde o elemento de autenticação é um código de barras bidimensional, conhecido como *QR-Code*. As informações deste *QR-Code* estão criptografadas para evitar que um usuário malicioso possa gerar um *QR-Code* válido em nome de outra pessoa.

1.1 Identificação do problema e Justificativa

As empresas tem grande preocupação em manter o controle do acesso de pessoas em seus ambientes e informações, por isso, buscam métodos que possam contribuir e assegurar tal sigilo. As empresas se deparam a diversas dificuldades, impedindo implementação das soluções existentes, ou desistência em utilizá-las.

A seguir, algumas tecnologias comumente utilizadas para controle de acesso físico e as dificuldades existentes:

- **Smartcards**: equipamentos leitores de *smartcards* e a produção de cartões são caros, além disto, quem adota esse tipo de tecnologia é constantemente atacado por leitores não autorizados, como Cívico [7] aborda em seu artigo;
- **RFID**: além do alto investimento, tipicamente não utilizam criptografia e, por isso, os dados ficam exposto, como exposto por Lehtonen [5] e Huang [8];
- **Biométrico**: custo comumente alto e, dependendo do usuário, a identificação é difícil, como por exemplo, a presença de suor nos dedos e pessoas com catarata [9] [3].

Outro problema que podemos considerar é a demora na autenticação. Bhanushali [10] afirma que, a usabilidade das soluções torna-se ineficiente quando o tempo de autenticação ultrapassa um tempo máximo de 2 segundos. Além disto, o tempo de autenticação adotado no mercado no qual insere-se esta ferramenta é de cerca de 3 segundos, como a solução recentemente apresentada pela empresa japonesa *Fujitsu* que afirma ser capaz de reconhecer a pessoa que está com a mão no leitor biométrico em apenas 2 segundos[1].

[1] http://www.fujitsu.com/global/solutions/business-technology/security/palmsecure

Nos códigos de barras EAN-13 (*European Article Number*), encontrados geralmente em unidades de consumo e supermercados, a velocidade de autenticação corresponde ao exigido pelo mercado, porém a quantidade de informações que este pode conter é muito limitado, sendo possível codificar apenas treze dígitos.

A partir de tais deficiências levantadas das soluções existentes, analisando os métodos de autenticação e suas vulnerabilidades, este trabalho tem o desafio de apresentar uma solução de baixo custo, alto grau de segurança e velocidade de autenticação satisfatória, desmitificando a ideia de que para obter segurança é necessário alto investimento monetário e complexidade na utilização do sistema.

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver um sistema de controle de acesso físico (CAF) utilizando *QR-Code* e criptografia assimétrica.

1.2.2 Objetivos Específicos

Dentre os objetivos específicos, pode-se destacar:

- Desenvolvimento de um módulo de criptografia assimétrica para codificação de dados de usuários e permissões de acesso;
- Desenvolvimento de um módulo de geração de *QR-Code* que contenham informações criptografadas de usuários;
- Desenvolvimento de um sistema de autenticação baseado nos *QR-Codes* gerados;
- Desenvolvimento de um sistema de informação para processamento de informações dos usuários;
- Testar da solução e comparar as outras ferramentas de CAF no tocante ao tempo de autenticação.

1.3 Organização do Documento

Os demais capítulos desta dissertação estão organizados conforme descrição a seguir.

- **No Capítulo 2:** são apresentados os conceitos relacionados a *QR-Code*, o criptossistema de chave pública RSA e trabalhos que utilizam *QR-Codes* e criptografia assimétrica;

- **No Capítulo 3:** destina-se a metodologia, descrição e esboço do comportamento do sistema proposto;
- **No Capítulo 4:** criação e implementação do sistema de controle de acesso proposto em um ambiente real. Criação e teste do funcionamento da tranca magnética a partir de acionamento realizado pelo sistema criado;
- **No Capítulo 5:** considerações finais e sugestões para trabalhos futuros.

Capítulo 2

Fundamentação Teórica

De acordo com Fanfara [11], a rápida expansão do uso da internet como meio de comunicação e, sobretudo, a conscientização dos benefícios do compartilhamento de dados comuns trouxe consigo a necessidade de construção de redes de comunicação segura. Cada organização ou indivíduo pode exigir níveis diferentes de segurança, mas para qualquer que seja o nível de segurança, a rede básica e de segurança de dados deve cumprir, pelo menos dois requisitos: o primeiro é a autenticação e autorização, e o segundo é transmitir qualquer que seja a informação de uma maneira que somente entidades habilitadas possam compreender, e para todo o resto, tal informação seja incompreensível.

Como forma de suprir a necessidade por segurança de informações, podemos usar a cifragem, que segundo Stallings [12] seria o uso de algoritmos matemáticos para transformar os dados em um formato que não seja prontamente decifrável. Essa abordagem possibilita propriedades importantes sobre os dados manipulados, tais como: confidencialidade (sigilo de informações), integridade (garantia que o dado não sofrerá mudança indesejada), autenticidade (garantindo de quem produziu o pacote criptografado é realmente quem diz ser) e irretratabilidade.

A criptografia pode ser de duas formas:

- Simétrica, que usam apenas uma chave segredo como artifício tanto de criptografia como descriptografia;
- e Assimétrica, também conhecida como criptografia de chave pública.

Na criptografia **Assimétrica**, é existente a presença de pelo menos dois elementos importantes para o êxito. O primeiro elemento é a chave pública (PU), utilizada por terceiros que serve para tornar um dado legível, indecifrável. O segundo elemento é a chave privada (PK), utilizada por um usuário específico para decifrar o dado, antes cifrado, em legível. Porém, o processo também pode ser realizado de forma inversa. Este processo garante confidencialidade da informação e/ou autenticidade.

De acordo com Stallings [12], o criptossistema de chave pública mais utilizado é o Rivest, Shamir e Adleman (RSA), proposto depois de um desafio lançado através do artigo de Diffie

e Hellman [13] que tinha como objetivo a busca por um algoritmo criptográfico que atendesse aos requisitos para sistemas de chaves públicas.

A construção das chaves(PU e PK) é realizada através de uma série de procedimentos a partir de dois números primos e, é neste fator que reside a grande dificuldade de atacar o RSA, fatorar um número em seus componentes primos.

Incorporando ao método de criptografia que tem como objetivo segurança, é notório a importância, em dias atuais, da praticidade que os objetos possam carregar consigo informações complexas de maneira mais simples. Um exemplo clássico é o código de barras, que têm como aplicação a eficiência em identificar objetos através de um conteúdo codificado em uma etiqueta passível de leitura. Esta abordagem é amplamente adotada em diversos contextos, como a identificação de produtos em supermercados, que possui informações como o nome do produto, datas de fabricação e validade, preços, distribuidora, entre outras informações. O código de barra EAN13, por exemplo, pode conter apenas 13 dígitos (0-9), sendo o último um dígito verificador, por isso, a quantidade de dados é limitado.

O *Quick Response Code* (**QR-Code**), é um símbolo bidimensional(2D), criado em 1994 pela empresa japonesa *Denso-Wave* com o principal objetivo de ser um código rapidamente interpretado pelos equipamentos de leitura, não difere muito de um código de barras, porém agrega uma maior densidade de informação e por isso, tem sido usado nas mais diversas ocasiões diárias por empresas de comércio e publicidade. A sua alta capacidade de armazenamento de informação é devido a possibilidade de conter informações tanto em posição vertical bem como horizontal. Atualmente, os *QR-Codes* são utilizados por inúmeras organizações e empresas nos mais diversos setores e o tipo de informação embutida pode ser desde um texto simples até uma URL ou mensagens SMS.

Tendo como principais pontos neste trabalho os requisitos mínimos a autenticação, irretratabilidade e alta performance, este capítulo objetiva-se abordar sobre o que são *QR-Codes* e como são formados, além de abordar ainda sobre fundamentos da criptografia assimétrica RSA.

2.1 Quick Response Code

Embora a popularização dos *QR-Codes* serem recentes, eles surgiram há cerca de 20 anos, na década 90. Criados pela empresa *Denso Wave*, com a finalidade de facilitar a identificação de partes dos carros em fábricas e também o processo de logística como um todo [14]. Podem ser considerados como uma otimização dos códigos de barras tradicionais, ordenando as informações em matrizes de duas dimensões, com conteúdo tanto na horizontal quanto vertical, como podemos observar na Figura 2.1. Portanto, são capazes de armazenar até 100 vezes mais dados e caracteres do que os tradicionais códigos de barras de uma dimensão(1D).

O *QR-Code* foi aprovado como padrão internacional em junho de 2000, pela ISO/IEC 18004, que define os requisitos para sua simbologia. Especifica suas características, métodos

Figura 2.1: *Exemplo de QR-Code*

de codificação de caracteres de dados, formatos de símbolo, características dimensionais, as regras de correção de erros, algoritmo de referência de decodificação, requisitos de qualidade de produção e parâmetros de aplicação selecionadas pelo usuário.

O processo de criação do *QR-Code* passou por 4 etapas básicas. Tais etapas fazem parte do processo evolutivo dos *QR-Codes* e padronização.

- Modelo 1: especificação original e descrito em AIM ITS 97-001 *International Symbology Specification-QR Code*.
- Modelo 2: atualização e melhorias em funcionalidades como adição de padrões de alinhamento. Base para primeira edição da ISO/IEC 18004.
- Modelo 3: similar ao Modelo 2, porém difere na adição de símbolos, com tons claros e *background* mais escuros, além da opção para especificar conjuntos de caracteres alternativos para o padrão.
- Modelo 4: o formato Micro *QR-Code* (também especificada na segunda edição da norma ISO/IEC 18004), é uma variante do *QR-Code*, que permite pequena ou moderada quantidade de dados a serem representado em um pequeno símbolo, particularmente adequado para marcação em peças e componentes automotivos, e para aplicações onde o espaço disponível para o símbolo é severamente restrita.

Segundo a *Denso Wave* [14], enquanto os códigos de barras convencionais são capazes de armazenar um máximo de cerca de 20 dígitos, *QR-Code* é capaz de lidar com dezenas, até mesmo centenas, de vezes mais informação. *QR-Code* é capaz de lidar com todos os tipos de dados, tais como caracteres numéricos e alfabéticos, códigos Kanji, Kana, hiragana, símbolos binários e comandos de controle. Até 7089 caracteres podem ser codificado em um símbolo.

Uma vantagem da utilização de *QR-Codes* é o fato destes serem digitalizados a partir de diferentes ângulos de 360 graus. Seu atual uso personalizado, com diversas cores e logótipos embutidos apenas são possíveis devido à propriedade de correção de erros.

Para padrão QR podemos ter quatro modos de codificação de texto: numérico, alfanumérico, byte e códigos Kanji/Kana, para cada temos as seguintes capacidades de armazenamento:

- Numéricos - quantidade máxima de 7089 caracteres;
- Alfanuméricos - quantidade máxima de 4296 caracteres;
- Binário (8 bits) - quantidade máxima de 2953 bytes;
- Kanji/Kana - quantidade máxima de 1817 caracteres.

2.1.1 Estrutura do *QR-Code*

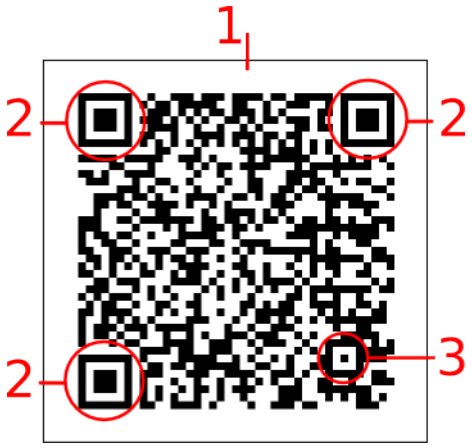


Figura 2.2: *Formatação de um QR-Code*

O *QR-Code* consiste de um conjunto de módulos quadráticos, dispostos em um padrão geral também formulado em quadrado (ponto 1 da Figura 2.2), que delimita o *QR-Code*. O ponto 1, também é comumente conhecida como "margem silenciosa".

A codificação inclui ainda um padrão localizador único, posicionados em três cantos do símbolo (ponto 2 da Figura 2.2) e que destina-se a ajudar na localização, tamanho e inclinação do código, e um quadrado menor que os dois anteriores, porém ainda maior do que todo o resto, que serve como referência para a lida do código (ponto 3 da Figura 2.2). Este ponto 3, é conhecido também como padrão de alinhamento e pode ser usado para corrigir distorções não lineares ao longo da área do símbolo, têm maior importância em proporção à informação codificada.

2.1.2 Gerando *QR-Code*

A formação do *QR-Code* passa basicamente por 7 passos até sua finalização.

1. É realizado uma análise dos dados submetidos, para posteriormente codificar em sequência de símbolos. Cada dado sob análise corresponde a um tipo codificação do texto distinto, e cada forma possui uma sequência de bits e tamanho também distintas uma das outras. Portanto, o primeiro passo deve ser destinado na realização da análise de dados para determinar se o texto pode ser codificado em numérico, alfanumérico,

byte, ou código Kanji, e então, selecionado o modo de sequencia de bits mais ideal para o seu texto.

2. Realizado a análise dos dados, vêm a codificação dos dados. O resultado desta etapa é uma sequência de bits que é dividido em palavras-código de 8 bits;

3. *Error Correction Coding*. Os *QR-Codes* podem usar correção de erros. Isso significa que, depois de criar a sequência de bits de dados que representam o texto, são gerados bits para palavras-código de correção de erro usando um processo chamado de *correção de erros Reed-Solomon*. Os *scanners*(leitores) de *QR-Codes* podem ler tanto as palavras-código dos dados quanto as palavras-código de correção de erros. Ao comparar os dois, o *scanner* poderá determinar se houve sucesso na leitura dos dados e poder corrigir possíveis erros;

4. Nesse passo, as palavras-código dos dados e as palavras-código de correção de erros gerados em passos anteriores devem agora ser dispostos em ordem apropriada. Para grandes *QR-Codes*, as palavras-código dos dados e correção de erros são gerados em blocos e, estes blocos, devem ser intercaladas de acordo com a especificação ISO/IEC 18004;

5. Depois de gerar as palavras-código dos dados e palavras-código de correção de erros e organizá-las na ordem correta, os dados são finalmente dispostos na matriz do *QR-Code*. As palavras-código são dispostas na matriz de uma forma específica. Durante esta etapa, é utilizado os padrões que são comuns a todos os *QR-Codes*, como as caixas quadráticas do ponto 2 da Figura 2.2;

6. Certos padrões na matriz de *QR-Codes* podem ser difíceis de serem lidos pelos *scanners*. Para compensar, é definido oito padrões de máscaras, cada uma das quais altera o *QR-Code* de acordo com um padrão em particular. Isso define o menor número de características indesejáveis em um *QR-Code*. Em outras palavras, significa simplesmente alternar a cor de um módulo (branco para preto, vice-versa).

7. O ultimo passo, define o formato e informações da versão. Neste caso é adicionar o formato e (se necessário) informações sobre a versão do *QR-Code*, acrescentando *pixels* em áreas específicas do código que foram deixados em branco nas etapas anteriores. Isto serve para identificar o nível de correção de erros e padrão de máscara a ser utilizada no *QR-Code*.

2.2 Criptografia Assimétrica RSA

Na criptografia assimétrica, há a existência de pelo menos dois elementos: uma chave pública(PU) e uma chave privada(PK). É possível distribuir PU a qualquer usuário ou sistema

para poder utilizá-la, criptografando a mensagem e por fim enviar ao seu destino. Por sua vez, o usuário destino ou sistema, descriptografa a mensagem usando a PK, mantida em segredo. Atualmente chaves com 2048 bits dão uma certa tranquilidade em referência a segurança por algum tempo, porém a medida que os computadores se tornam mais poderosos este número tende a aumentar. Além disso, para sucesso do algoritmo, segundo Stallings [12], mesmo que um criptoanalista tenha conhecimento do algoritmo e de uma das chaves, é insuficiente para o atacante determinar a outra chave do sistema.

Observando a Figura 2.3 e considerando o que Stallings [12] afirma, existe uma mensagem em texto claro, denotado por X, produzido por uma origem A e que será destinado a B. Ao chegar em B, deverá chegar de forma criptografada. Portanto B deverá gerar duas chaves: PU_b e PK_b. A chave PU_b, B fornecerá a quem deseja se comunicar com ele. Já a chave PK_b ficará com B para que este possa descriptografar a informação e compreender o conteúdo.

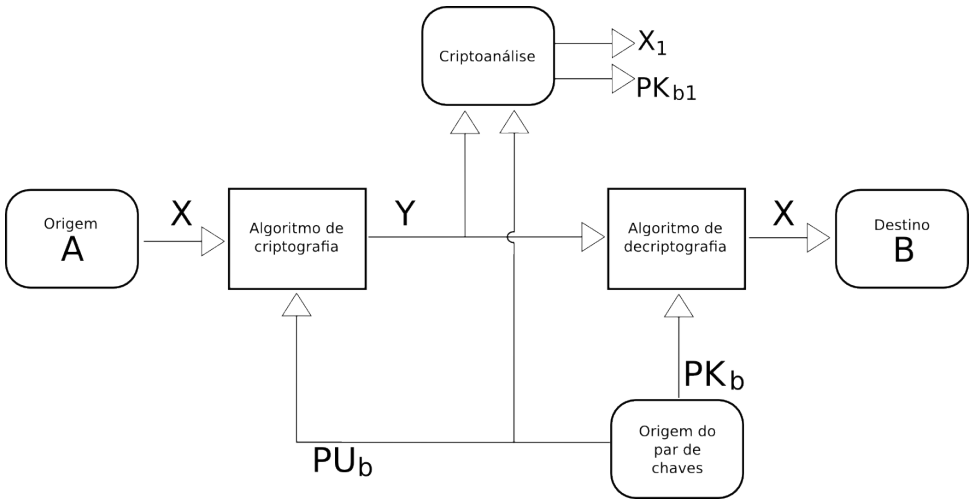


Figura 2.3: *Criptossistema de chave pública*

Com a mensagem X e a chave PU_b como entrada, a origem A forma o texto cifrado Y:

$$Y = E^1(PU_b, X)$$

O receptor a quem se destina a mensagem, de posse da chave privada correspondente, é capaz de inverter a transformação:

$$Y = D^2(PK_b, Y)$$

Porém, pode haver um adversário entre a comunicação da origem A e o destino B. Este adversário pode ter acesso a PU_b e a mensagem Y, mas não tem acesso a PK_b e a X. Seu esforço estará em recuperar a mensagem X, gerando uma estimativa de texto claro X_1. Frequentemente, o adversário também está interessado em poder ler mensagens futuras, gerando um PK_{b1} estimado.

[1] Encrypt = criptografia
[2] Decrypt = descriptografia

2.2.1 Algoritmo RSA

Buscando então a complexidade e impossibilidade para um criptoanalista atacar, o RSA é baseada na **Teoria dos Números**, com dificuldade em fatorar um número em seus componentes primos, envolvendo exponenciação de números inteiros e módulos de n, usando um formato de blocos, e cada bloco subsequente possui um valor binário menor que algum número n.

Seja M texto claro e C bloco cifrado, a criptografia e descriptografia possuem a seguinte forma:

$$C = M^e(\mod n)$$

$$M = C^d(\mod n)$$

Tanto a origem quanto o destino conhecem o valor n, o emissor conhece e e somente o destino conhece o valor de d. Assim, o algoritmo de chave pública é $PU = \{e, n\}$ e de chave privada é $PK = \{d, n\}$.

Como requisito de sucesso, devemos considerar o seguinte caso:

- Encontrar valores de e, d e n, tal que $M^{ed} \mod n = M$ para todo $M < n$

Isso implica que o relacionamento se mantém caso e e d forem inversos multiplicativos módulo $\phi(n)$, em que este é o *tociente* de Euler. Ou seja, temos que considerar a seguinte expressão:

$$ed \mod \phi(n) = 1 \tag{2.1}$$

Isto é equivalente a: $d = e^{-1} \mod \phi(n)$. Sendo assim, esta afirmação só é possível ser verdadeira caso d e e sejam relativamente primos a $\phi(n)$. De maneira similar, podemos considerar também que $mdc(\phi(n), d) = 1$.

Em resumo, o algoritmo segue os seguintes passos:

Selecione $p.q$	p e q são primos , $p \neq q$
Calcule $n = pXq$	
Calcule $\phi(n) = (p-1)(q-1)$	
Selecione o inteiro e	$mdc(\phi(n), d) = 1; 1 < e < \phi(n)$
Calcule d	$d = e^{-1} \mod \phi(n)$
Chave Pública	$PU = \{e, n\}$
Chave Privada	$PK = \{d, n\}$

Tabela 2.1: *Geração de Chaves*

2.2.2 Assinatura Digital

A ideia de usuários transportarem seus respectivos *QR-Codes* como identificadores pessoais por si só não é suficiente em termos de segurança, ainda que estes sejam analogamente vistos

Texto Claro	$M < n$
Texto Cifrado	$C = M^e($ mod n

Tabela 2.2: *Criptografia*

Texto Cifrado	C
Texto Claro	$M = C^d($ mod $n)$

Tabela 2.3: *Decriptografia*

como chaves pessoais, que estão sempre sob posse dos donos e ao chegar em lugares propícios a seus usos, como exemplo a entrada da casa do usuário, use a chave para liberar o cadeado.

A fraude de conteúdos digitais para um atacante experiente é possível de ser realizado, ainda que criptografado, sendo suficiente através uma criptoanálise bem estruturada e estudada por parte do atacante. Para aumentar a complexidade do sistema e garantir a autenticidade de um documento legal digital é disposto atualmente a presença da *Assinatura Digital*, similar a uma assinatura manual utilizada como autorização de documentos em papel.

Tal método permite assinar documentos digitais, garantindo autenticidade. Uma assinatura digital é um mecanismo de autenticação, que anexa ao código criado um número *hash*[3], que é único, na mensagem assinada e criptografando-a com a chave do criador ($PK_{emissor}$). Com isto, o receptor pode verificar a identidade do emissor e então garantir que a mensagem, mesmo criptografada, é realmente o emissor esperado.

Isto promove ao emissor o não repúdio, ou seja, afirma que o conteúdo não foi alterado, ou ainda, o receptor não pode gerar uma mensagem e atribuí-la a outra pessoa.

Um exemplo é uma transação financeira. Um cliente X manda uma ordem para o banco efetuar uma transação, como uma transferência de fundos. O banco precisa verificar se o cliente que está autorizando é de fato o cliente X, neste caso utiliza-se autenticidade. O banco deve prover um método de como provar que o cliente que está autorizando é de fato cliente X, atribuindo ao processo o ato de não-repúdio. Garantido isto, o banco não pode inventar a transação, sendo assim, se o banco atribuir ao cliente X uma transação, X não tem como provar que não foi ele realizou a transação.

Assinatura Digital e RSA

No exemplo anterior, podemos ter a soma da assinatura digital com a criptografia RSA garantindo autenticidade e confidencialidade, com o uso da chave privada do emissor e a chave pública do receptor. O emissor cifra a mensagem de permissão (P) com a chave privada ($PK_{cliente}$), garantindo autenticidade, e depois cifra com a chave pública do receptor (PU_{banco}), garantindo confidencialidade.

[3]O número hash é único e fácil computar a sua saída de tamanho fixo. Duas entradas diferentes não produzem a mesma saída *hash*, assim torna mais seguro o conteúdo do documento.

Quando o cliente cifra, é realizado o seguinte algoritmo:

$$C = E(PU_{banco}, E(PK_{cliente}, P))$$

Quando o banco decifra, o algoritmo realizado é:

$$P = D(PU_{cliente}, D(PK_{banco}, C))$$

Em resumo, o algoritmo para cifrar e decifrar, juntamente com assinatura digital, é observado na Figura 2.4.

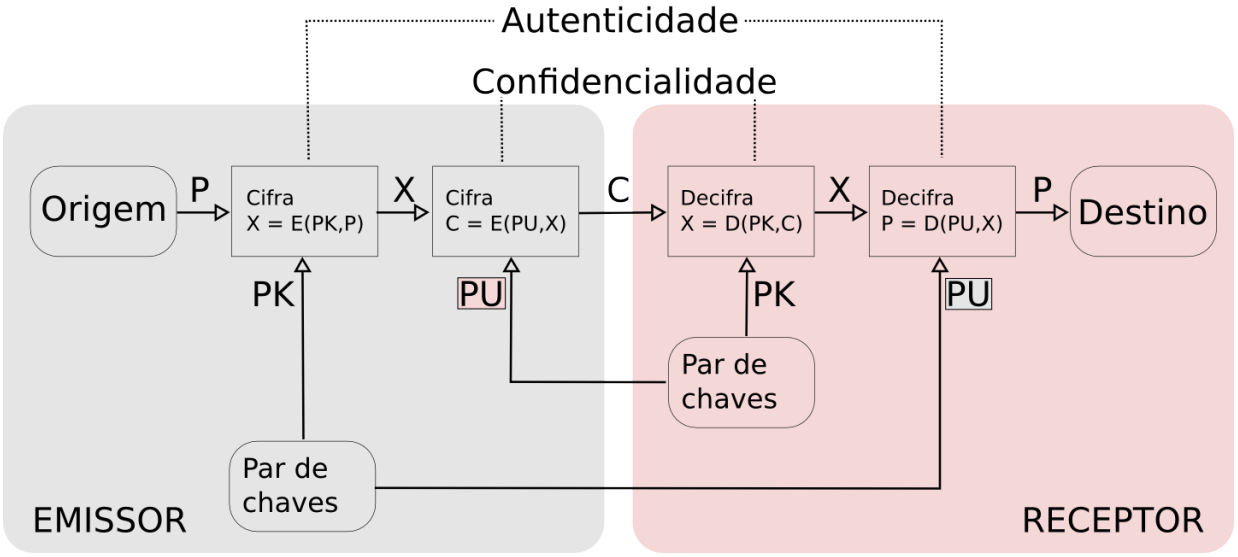


Figura 2.4: *Cifragem e decifragem com assinatura digital*

2.3 Trabalhos Relacionados

Trabalhos relacionados a *QR-Codes* e ao criptossistema RSA vêm sendo desenvolvidos nos últimos anos.

A instituição Banco do Brasil [15], propõe uma solução que através do computador, o usuário acesse a conta bancária. Em seguida, selecionando a transação desejada a partir de uma senha de 6 dígitos recebe uma imagem *QR-Code* com dados criptografados, em que deve realizar a captura da imagem.O usuário tem todo acesso ao sistema proposto pelo banco através de aplicativos de celular, apresentando a imagem do *QR-Code* ao leitor do aplicativo.

Buscando facilidade através de serviços online, Conde [16] propôs um novo modelo para e-Tickets baseando-se em *QR-Codes* com conteúdo criptografado pelo usuário. Esta proposição é compatível com sistemas de controle de acesso com infraestruturas mínimas, tem vários níveis de segurança e se baseia em bloco cifra com senhas de usuários.

Áreas voltadas a robótica, um trabalho sobre a criação de um robô *mobile* baseado em identificar *QR-Codes*, realizado por Li [17] mostra ser possível através do uso do método de

vetores, expressar as características de cores de uma imagem e usar o método mínimo de quadrados para determinar linhas de bordo do padrão *QR-Code*, podendo capturar claramente e se comportar bem em distâncias de 0,5 metro a 2 metros. Ainda com dispositivos *mobiles*, Fang [18] fornece um mecanismo de autenticação *offline* utilizando *QR-Codes*. Com base na tecnologia de criptografia visual, é possível verificar o acesso a *QR-Codes* e controlar a permissão dos dados protegidos.

Cho [19], no artigo *Study on Method of New Digital Watermark Generation Using QR-Code*, propõe um método para gerar *QR-Codes* que possam levar muitas informações e usá-las como marcas d'água digitais, principalmente em incluir várias informações como páginas web diretamente conectadas por links. E, por fim, em *Reversible data hiding with histogram-based difference expansion for QR code applications*, Huang [20] sugere um novo algoritmo na ocultação de dados reversíveis, com a aplicação associada a *QR-Codes*. Assim, com finalidade de reter o valor da imagem original, mantendo a capacidade para o acesso imediato para páginas web, seria a principal preocupação deste trabalho.

No caso de estudos e trabalhos relacionados ao criptossistema RSA, este tem sido aplicado nas mais diversas áreas de conhecimento. Seja no campo acadêmico, seja em âmbito empresarial. Fato é que este criptossistema ainda tem sido alvo de pesquisas, em especial na área das ciências exatas e tem sido bastante difundido.

No artigo de Masuda [21], observa-se que ao se estudar a Teoria de Corpos Finitos relacionando-a com a criptografia, observa-se naturalmente o conceito de números primos e divisibilidadade, existindo diversas maneiras de seguir esse caminho e os corpos finitos aliados ao criptossistema RSA é um deles.

No comercio voltado a serviços online e internet, temos como usuário deste criptossistema a Autoridade Certificadora Raiz da ICP-Brasil (AC-Raiz), competente em emitir, expedir, distribuir, revogar e gerenciar os certificados das autoridades certificadoras de nível imediatamente subsequente ao seu, utilizando RSA em sua infraestrutura [22]. Ainda na área de computação, voltado ao desenvolvimento de aplicações em linguagem de programação Java, há a existência de pacotes de segurança oferecendo várias camadas de controle de segurança, devido a sua forte associação à Internet. Utilizado para gerar criptografia, autenticação, assinatura digital e pares de chaves (pública e privada).

Uma variação do RSA proposto por Sun [23] em seu artigo, sugerido pelo autor como *Dual RSA* fornece uma análise de segurança. Esta variante da família RSA, poderia ser usado em situações que for requerido dois casos da RSA, com a vantagem de reduzir os requisitos de armazenamento para as teclas.

Entre estas, ainda podemos considerar outros trabalhos comerciais e não-comerciais também relacionados a aplicação do criptossistema RSA, como:

- **Protocolo SSL e OpenSSL:** camada do protocolo de rede responsável no gerenciamento de um canal seguro entre cliente e servidor, sob licença GPL;
- **Skype:** usa AES, também conhecido por Rijndael, que é usado sempre pelo governo

dos EUA para proteger informações confidenciais, e o Skype usa a criptografia máxima de 256 bits. As chaves públicas de usuários são certificadas pelo servidor do Skype no momento da conexão usando certificados RSA de 1536 ou 2048 bits [24];

- **Cartões inteligentes:** implementado em cartões inteligentes, como por exemplo, no cartão inteligente Gem Safe Xpress;
- **PGP**: o RSA é padrão para o software de segurança para correio eletrônico, podendo realizar criptografia e descriptografia, como ainda uso de assinaturas digitais e gestão de chaves públicas;
- **Módulos Paralelos de Operações em criptografia RSA por CPU/GPU híbrido Computação:** proposto por Lin [25], tem como objetivo um método para resolver o gargalo da execução de algoritmos de criptografia RSA em CPUs. O algoritmo de criptografia RSA envolve operações utilizando grandes números, então tem-se como busca o aprimoramento do desempenho de *hardwares* dos computadores, não afetando no comprimento da chave RSA, que é o aumento de segurança substancial.

Capítulo 3

Metodologia

Considerando os objetivos apresentados na Introdução e os conceitos abordados no Capítulo 2 (*Fundamentação Teórica*) a metodologia deste trabalho foi:

1. **Projeto do sistema**;
2. **Implementação do sistema**;
3. **Testes e documentação**.

3.1 Projeto do sistema

Nesse passo o sistema foi proposto e projetado, a partir de fluxogramas sobre como os componentes estarão interligados, as tecnologias necessárias e como elas se relacionam. Foi realizado um estudo e integração de bibliotecas e tecnologias destinadas na criação e formatação de *QR-Code*, além de implementação de uma tecnologia que fosse capaz de realizar a leitura de um *QR-Code* com alta precisão e alta velocidade, garantindo o melhor desempenho possível do sistema.

3.1.1 Sistema Proposto

A solução para o serviço de controle de acesso proposto neste trabalho é baseado no uso de *QR-Codes* que possuem conteúdo criptografado. Estes *QR-Codes* servirão como objetos de autenticação, funcionando como chave de acesso.

A Figura 3.1 apresenta o processo de geração de um *QR-Code*, que ocorre devido uma solicitação realizada pelo usuário ao administrador do sistema. O administrador do sistema, a partir de informações específicas do usuário, fornecerá um *QR-Code* criptografado. O *QR-Code* pode ser fornecido de forma impressa, ou por captura da imagem através de câmera de *smartphones* ou até mesmo recebimento da imagem via e-mail.

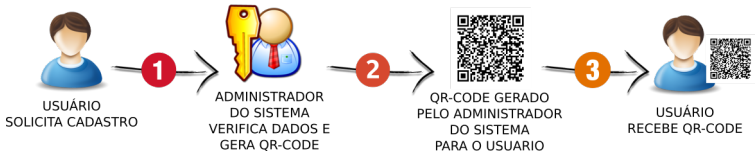


Figura 3.1: *Geração de QR-Code para usuários*

O processo para fornecer um *QR-Code* possui 3 etapas. A etapa 1, é o cadastro do usuário no sistema a partir uma senha temporária - **OTP**(*One-Time Password*) -, utilizando-a para gerar um *QR-Code*. Este passo é adotado para evitar fraudes, impossibilitando que o administrador do sistema possa gerar outros *QR-Codes* no nome do usuário sem autorização. Caso necessário um novo *QR-Code*, será fornecido ao usuário uma nova senha de curto prazo, via SMS ou e-mail. A Figura 3.2 representa esta primeira etapa.

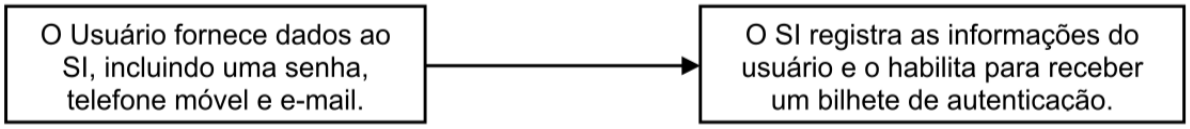


Figura 3.2: *Cadastro do usuário no sistema*

É importante destacar que esta etapa é crítica, devido existência de senhas. Isto acarreta que não foi solucionado por completo a dependência de métodos a partir do uso de senhas que, como citado na *Introdução* deste trabalho, pode ser um problema para pessoas idosas.

A etapa 2 é a geração do *QR-Code* específico para o usuário e a etapa 3 é a geração do *QR-Code* para autenticação.

Quando o usuário necessitar de um novo *QR-Code*, será fornecido uma senha que apenas o utente conhecerá. Se a senha for válida no momento da solicitação de um *QR-Code*, é gerado um novo *QR-Code* e entregue ao usuário. Se senha não for válida, o sistema irá gravar tentativa em *log* e informará o usuário via SMS ou e-mail. Este processo é apresentado na Figura 3.3.

É importante destacar que, a etapa de utilização de senha por parte do usuário não foi implementado, sendo abordado então no Capítulo 5 em *Propostas de Trabalhos Futuros* deste projeto.

Com posse de seu *QR-Code*, o usuário poderá utilizá-lo para poder ter acesso a determinados ambientes. O processo de autenticação ocorre basicamente sob forma apresentada na Figura 3.4.

Para a autenticação, o usuário apresentará seu *QR-Code* uma câmera leitora - chamaremos de autenticador - que irá fazer a leitura da imagem e enviará ao servidor do sistema. A imagem é processada e, verificando-se as informações contidas, são retornadas as ordens de permissões (liberado ou negado a entrada do usuário) para acesso aquela área.

Caso validado, o usuário terá acesso ao local. Caso de negado a permissão de entrada, o usuário não poderá acessar ao ambiente. Este processo é apresentado na Figura 3.5.

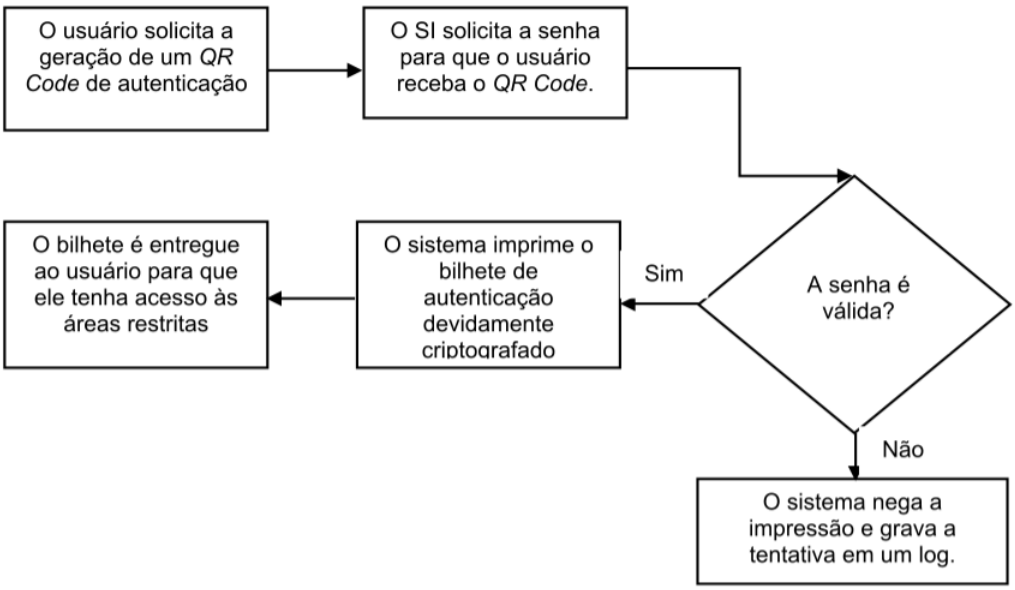


Figura 3.3: *Geração do QR-Code de autenticação*

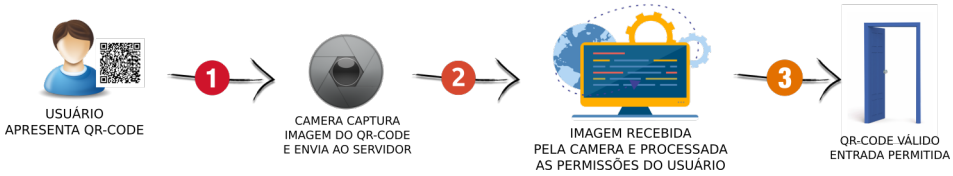


Figura 3.4: *Processo de autenticação de usuário*

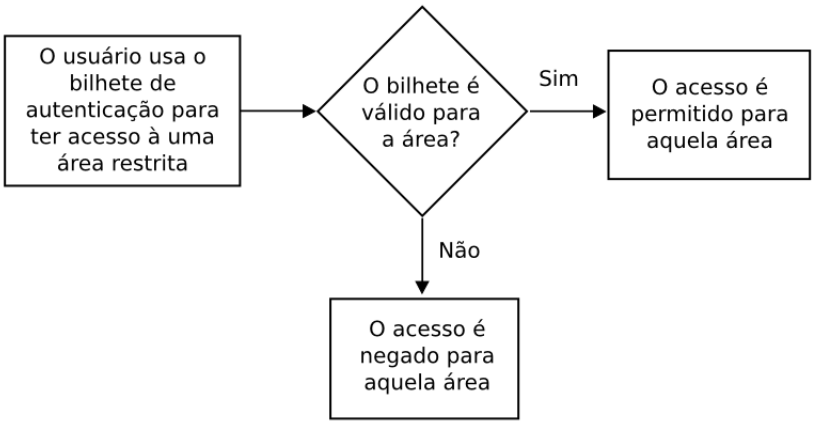


Figura 3.5: *Autenticação do usuário*

3.1.2 Arquitetura do Solução Code Control

A arquitetura do sistema é de acordo com a Figura 3.6, a comunicação entre os elementos é da seguinte forma:

1. A câmera autenticadora se comunica com o servidor enviando os dados da imagem capturada, através de comunicação *ethernet*, sempre que um *QR-Code* é apresentado para teste de autenticação. Esta captura é realizada pelo servidor é automático, devido a biblioteca de leitura de *QR-Code* que identifica o código esperado;

2. O servidor, irá processar os dados e, sendo válida a entrada, será enviado um sinal ao Arduíno informando em qual respectiva saída deve ser acionado um pulso eletromagnético;

3. O Arduíno irá executar seu processo e interpretar em qual saída deve ser enviado o sinal de pulso elétrico. Interpretado, é enviado sinal ao relé conectado a trava;

4. A tranca, recebido o sinal no relé através do Arduíno, acionará a trava eletromagnética.

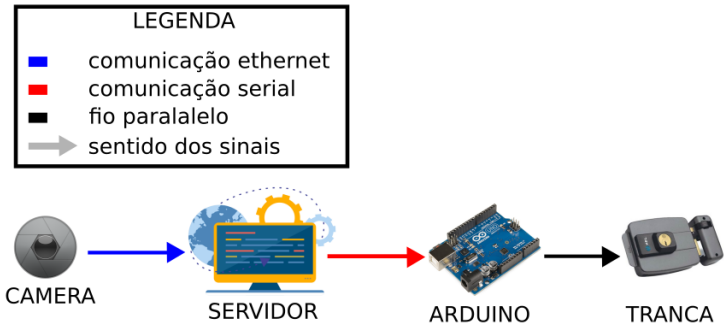


Figura 3.6: *Arquitetura do Code Control*

3.2 Usuários Submetidos ao Sistema

O sistema funciona baseado em torno de um elemento específico: o usuário. O *Code Control* possui políticas padrões para o administrador do sistema e para os demais usuários, seja um empregado ou um empregador, visitante ou outro qualquer perfil de usuário.

A Figura 3.7 apresenta o modelo Entidade-Relacionamento que pode adotado para criação do usuário Administrador no banco de dados.

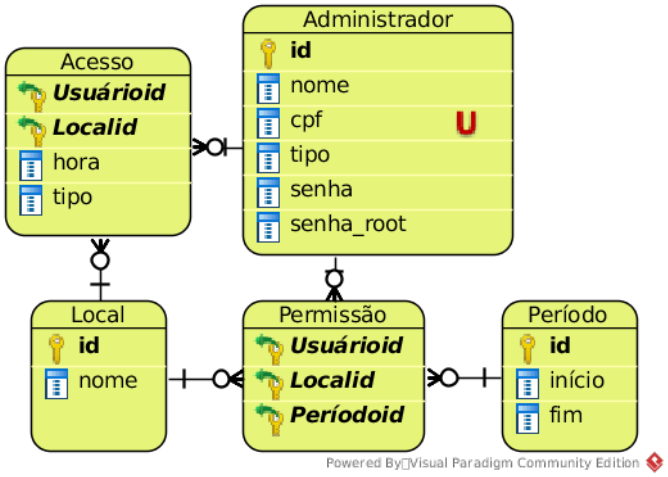


Figura 3.7: *Exemplo de Diagrama de Entidade e Relacionamento para o Administrador do Sistema*

O Administrador possuiria as seguintes características: um número de *id*, identificando o elemento como único no sistema; o nome pessoal do usuário e seu CPF; a classificação de seu tipo referente ao nível de acesso as áreas; senha pessoal, para realizar a função de adicionar e remover usuários no *Code Control*, e ; uma senha *root*, que será usada para instalar pacotes, atualizar e executar a maior parte da manutenção do sistema. Ao se autenticar como *root*, o Administrador terá total controle sobre o sistema.

Uma vez que o *Code Control* pode ser aplicado em diversas áreas como uma biblioteca, um banco ou até mesmo em residências. Sendo assim, o usuário que não é o Administrador será considerado uma entidade genérica[1], no qual este poderia adotar seguintes características: um número de *id*, identificando o elemento como único no sistema; o nome pessoal do usuário e seu CPF; a classificação de seu tipo referente ao nível de acesso as áreas; senha pessoal; estado recente de presença - se o usuário está dentro de uma sala ou ausente do prédio; um e-mail e número telefônico para contato.

A Figura 3.8 apresenta o modelo Entidade-Relacionamento que pode ser considerado para criação do Usuário Genérico no banco de dados.

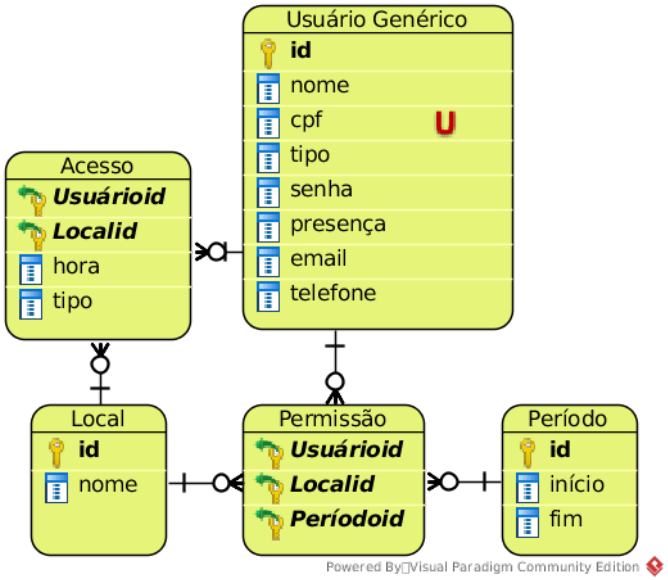


Figura 3.8: *Exemplo de Diagrama de Entidade e Relacionamento para um Usuário Genérico*

3.2.1 Controle dos Usuários

O controle de usuários, sendo a função específica do sistema, possui caráter de alto risco, uma vez que o sistema depende do seu bom funcionamento, apresentado na Figura 3.9.

A tarefa remete inclusão de um subsistema que gere arquivos *Log*. A geração de *Logs* serve para que o administrador verifique os registros dos eventos considerados relevantes, contribuindo acerca do comportamento esperado, ou inesperado, do sistema.

[1] Justamente pela diversidade de aplicação do *Code Control* em diversos ambientes

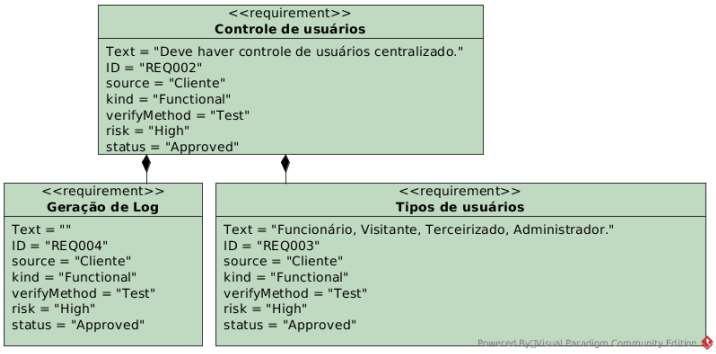


Figura 3.9: *Diagrama de Requerimento: Controle de Usuários*

Alta velocidade de autenticação do sistema deve ser proveniente da detecção de *QR-Codes*, além da análise dos dados para controle de acesso. A segurança necessária na autenticação deve manter o sistema intacto, prevenindo de ataques indesejados, garantindo integridade ao sistema, confidencialidade, o controle de acesso, irretratabilidade e total disponibilidade do sistema. Na Figura 3.10 observamos tais características.

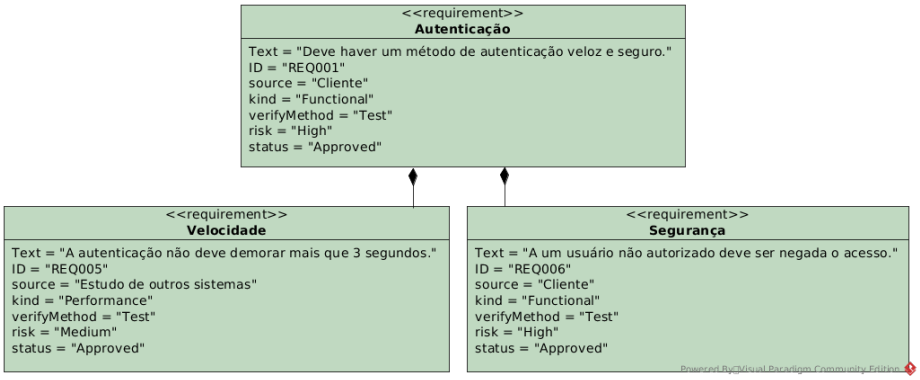


Figura 3.10: *Diagrama de Requerimento: Características da Autenticação*

3.2.2 Padrão de Leitura de Múltiplas Câmeras Autenticadoras

Para a escalabilidade referente ao aumento de quantidades de pontos de autenticação consideramos o que Gamma [26] afirma sobre o padrão de projeto conhecido como *Observer*. Este padrão é utilizado quando uma modificação em um objeto requer mudança em outros e não se sabe quantos objetos precisam ser mudados ou acrescentados. O diagrama apresentado na Figura 3.11 apresenta as classes *Autenticador* e *Sistema*, que implementam as funções de *interfaces* conhecidas pelo padrão *Observer* como objeto Observador e objeto Observado, respectivamente.

Se adicionar-mos uma câmera, esta será vista como um novo Autenticador, solucionando portanto o nosso problema existente de adição de novos pontos autenticadores.

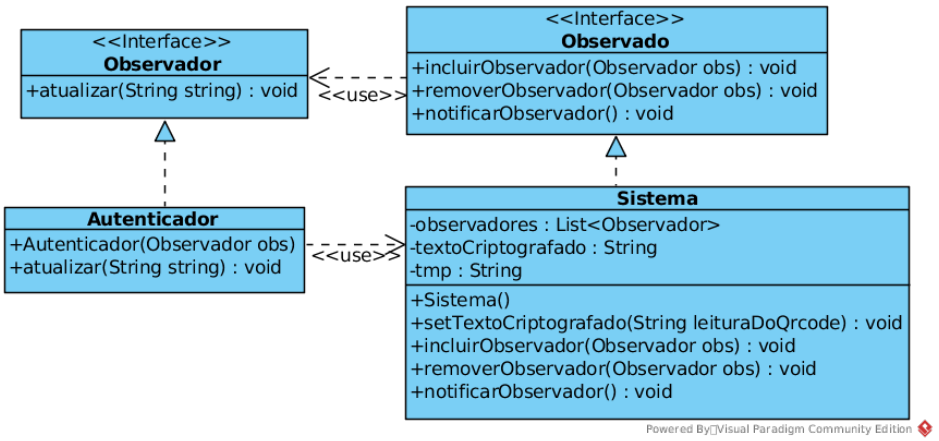


Figura 3.11: *Diagrama de Classes: Padrão Observer*

3.2.3 Bibliotecas para Criação e Leitura do QR-Code (Extensões)

Gerar e ler um *QR-Code* são ações importantes do sistema, portanto, agregamos itens que pudessem realizar tais tarefas com sucesso, obedecendo os requisitos técnicos de formatação do *QR-Code*, atendendo a alta velocidade na geração e leitura dos códigos, seguir normas e especificações internacionais prescritas e verificação teste de integridade dos códigos. Além disto, as ferramentas foram agregadas também para evitar um esforço desnecessário, apropriando-se de tecnologias públicas que correspondesse as necessidades do projeto. Sendo assim, foi realizado uma pesquisa de bibliotecas externas para criação e leitura de um *QR-Code*.

Há um grupo que destina-se no desenvolvimento de artefatos com a finalidade da criar *QR-Codes*, este grupo é conhecido por *ZXing*. Tal grupo dispõe de um sistema de controle de versão(SVN) público e gerenciam todo o seu código fonte através de uma página *Github*[2]. O *ZXing* é código aberto e multiformato, destinado para geração de diversos tipos de códigos de barra, desde **1D** até **2D**. Suporta diversas linguagens de programação (Java, Python, C++, entre outras).

Os formatos suportados pela biblioteca ZXing estão dispostos na Tabela 3.1

1D product	**1D industrial**	**2D**
UPC-A	Code 39	QR Code
UPC-E	Code 93	Data Matrix
EAN-8	Code 128	Aztec (beta)
EAN-13	Codabar	PDF 417 (beta)
	ITF	
	RSS-14	
	RSS-Expanded	

Tabela 3.1: *Formatos de códigos suportados pela biblioteca ZXing*

[2]https://github.com/zxing

Outra biblioteca utilizada no trabalho é *Sarxos*[3]. Esta biblioteca destina-se a leitura do *QR-Codes*, permitindo ao usuário utilizar de *webcams* externas, conectando-se diretamente via comandos Java. Projetada para abstrair os recursos da câmera comumente utilizados, basta o usuário fazer poucas alterações para ter captura de imagens. Esta biblioteca serve ainda de apoio a vários *frameworks* destinados a captura de imagens. Sua principal ênfase é o uso em plataformas *Raspberry PI* e recebe atualizações constantes.

3.3 Implementação do sistema

Após levantamento de requisitos e identificadas as tecnologias que seriam utilizadas implementamos as ações que o sistema poderia realizar. A codificação do software foi realizada em código Java, considerando Diagrama de Classes do sistema apresentado na Figura 3.12.

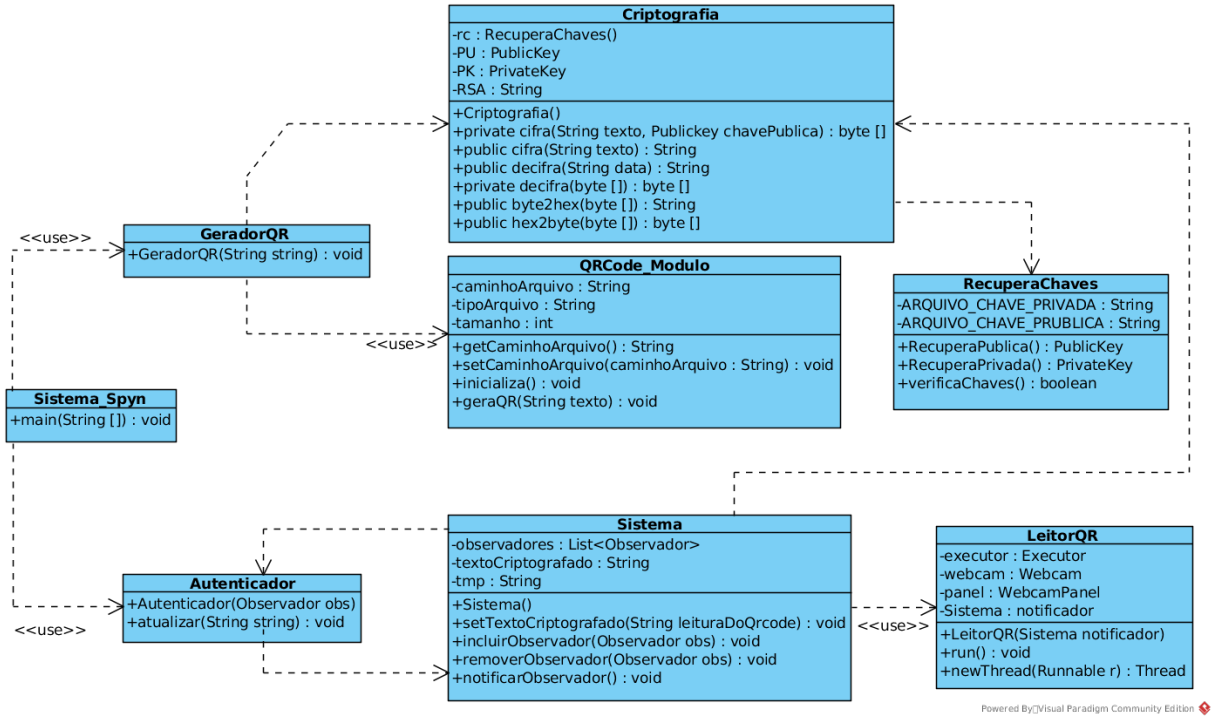


Figura 3.12: *Diagrama de Classes: Visão Geral*

Foi necessário ainda a criação de central responsável pela liberação da tranca eletromagnética de portas, utilizando um *Arduíno Uno (Duemilanove)* com uso de códigos de programação desenvolvido em linguagem C - e *Shield Ethernet* W5100 -, e confecção de protótipo de tranca magnética da porta, com ativação a partir de correntes elétricas enviadas por relé confeccionado, para acionamento e interrupção da trava.

[3]https://github.com/sarxos/webcam-capture

3.3.1 Central de Liberação da Tranca Eletromagnética Utilizando o *Arduíno*

A central de liberação da tranca é um componente formado pela junção de dois objetos que serão descridos a seguir: o Arduíno, que é a peça de comunicação entre o sistema e a trava, e a trava eletromagnética, acionada a partir de impulsos recorrentes do relé produzido de maneira caseira.

Plataforma Arduíno

Para comunicação entre a central do sistema e os autenticadores, foi utilizado uma placa Arduíno conectada a um dispositivo com interface de rede internet conhecida por *W5100* (o *Shield Ethernet*). Na Figura 3.13 observamos o *Arduíno* e o *Shield Ethernet*.

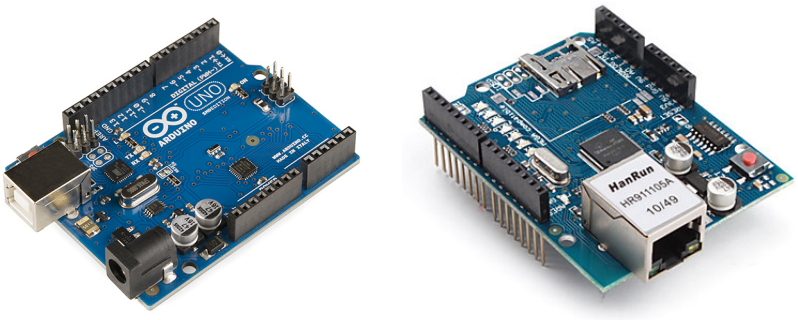


Figura 3.13: *Plataforma Arduíno e Shield Ethernet para Arduíno*

A plataforma *Arduíno* tem sido amplamente utilizado em projetos que envolvam implementações de hardware e software em diversas áreas e aplicações eletrônicas. A plataforma consiste em um kit hardware considerado eletrônico livre, ou seja, projetado e oferecido da mesma maneira que um software de código livre. É composto por uma placa de circuito impresso, com presença de um microcontrolador *ATmega* do fabricante *ATMEL*.

Para maiores funcionalidades do kit *Arduíno*, utiliza-se placas de expansão, conhecidas como *Shields*, de fácil implementação e encaixe.

O *Arduíno* se comunica com o *W5100* utilizando os pinos digitais 4, 10, 11, 12 e 13. Sendo o pino 10 utilizado para selecionar o *W5100*. Esses pinos não podem ser utilizados para outras conexões. O *Shield* apresenta um conector RJ45 fêmea, que pode ligar através de um cabo UTP (*Unshielded Twisted Pair*) em um conector RJ45 macho à rede de computadores.

Para as configurações de rede, são usadas as bibliotecas padrões do *Arduíno SPI.h* e *Ethernet.h*. São atribuídos os parâmetros necessários para que o serviço funcione de acordo com o esperado.

Tranca Magnética para Portas

A Figura 3.14 mostra como funciona a tranca. Um dispositivo leitor de informações externas, quando recebe uma informação apropriada, envia um pulso ao *Circuito de Controle*. Tal circuito recebe o pulso e aciona o relé (Circuito de Acionamento de Trava Elétrica(*Driver*)), liberando apropriadamente a trava, permitindo a entrada do usuário ao ambiente.

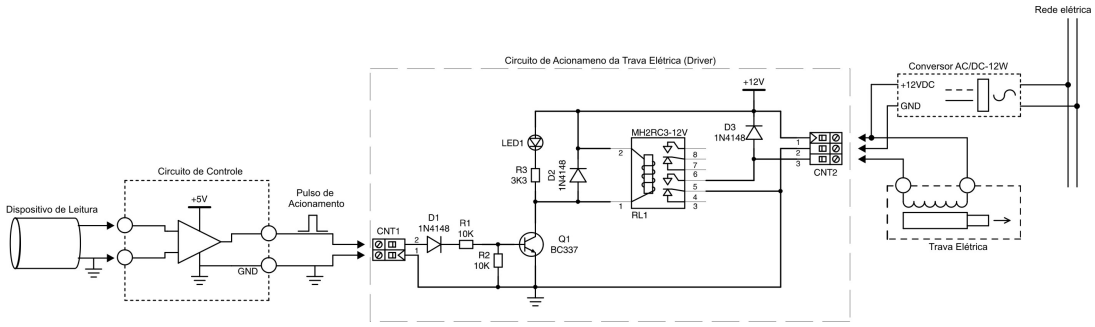


Figura 3.14: *Protótipo de Tranca Magnética para Portas*

A trava eletromagnética, originalmente está em estado de bloqueio da porta, e pode vir a liberar a passagem para o usuário a partir de pulso elétricos, como resposta de leitura de um *QR-Code* válido.

Alimentada continuamente por uma tensão de *12 VDC*(Volts Corrente Alternado), a fechadura ou trava eletromagnética, é composta por um eletroímã e uma placa metálica conhecida como *atraque*. Quando o atraque se aproxima da área de atração do campo magnético, posicionada sob as barras metálicas da fechadura, esta se prende magneticamente, de forma que para conseguir retirá-lo é necessária aplicar uma determinada força.

A força que deve ser aplicada pode variar a partir da especificação da trava, sendo a variação de força para o atraque de 50 kgf(quilograma força) até 1500 kgf.

3.4 Testes e Documentação

Após construído o sistema, submetemos a cerca de 35 pessoas que forneceram apenas seus nomes e criptografamos. Após criptografia, foi gerado *QR-Code* e entregamos a seus respectivos donos.

Após entregue o *QR-Code*, foi realizado a leitura em um sistema computacional externo, como *smartphones*, e foi constatado na leitura que o conteúdo lido é totalmente incompreensível, em formato hexadecimal. Os usuários também se submeteram para tentativa de liberação da tranca e em todos os casos houve sucesso.

Finalmente, removemos os nomes de determinados usuários de nosso banco de dados e ao submeter os *QR-Codes* dos usuários que tiveram seus nomes excluídos, não houve liberação da tranca eletromagnética.

Concluídas as etapas anteriores, este trabalho pôde ser desenvolvido.

3.5 Discussão

Para obter sucesso no projeto fez-se necessário o levantamento de requerimentos. Esta etapa foi realizada baseado-se em um conjunto de ideias pré-formadas discutíveis, havendo alguns pontos que, na medida que foi sendo produzido o trabalho, foi sendo aprimorado ou excluído. Para cada requerimento levantado, foi estabelecida uma série de situações que possam suprir eventuais imprevistos, afim de verificação se tal requisito pode ser tangível. A grande vantagem desta abordagem é a possibilidade de criar o conceito do sistema proposto ser estável e padronizá-lo para futuras ações e modificações.

A necessidade de conhecimento sobre padrões de projetos proporcionaram qualidades ao sistema como a modularização, que facilita o escopo de estudo a terceiros, uma vez que, possuem flexibilidade e podem ser aplicados nas diversas etapas do desenvolvimento de um software, que sob um contexto, executam combinações recorrente de elementos, utilizam documentos mapeados e podem gerar automaticamente serviços e arquivos, buscando otimização do sistema.

Capítulo 4

Resultados

4.1 Composição do Sistema

Considerando a formatação apresentada na Seção 4.2, o sistema é composto por 5 (cinco) módulos, disposto nos Apêndices A, B, C, D e E, classificados como:

A Criptografia: destinado a criptografia e descriptografia;

B Notificação: classes que implementam o padrão *Observer* ao sistema;

C QR-Code: formatação de um QR-Code, a partir da entrada de dados;

D Persistência: persistência das chaves PU e PK;

E Gerador e Leitor: direcionado à chamada da geração de QR-Code e acionamento da leitura para autenticação, respectivamente.

No Apêndice F temos o módulo que contém a codificação para o dispositivo *Arduíno*. O *Arduíno* receberá informações do sistema e acionará a trava eletromagnética.

4.1.1 Criptografia

A criptografia é baseada no algoritmo RSA. A classe *Criptografia.class* (Apêndice A - módulo A.1) usa esse algoritmo para prover os serviços de criptografia e descriptografia.

A principal composição desta classe faz menção aos métodos **cifrar** e **decifrar**. Como os nomes dos métodos propõe, são destinados a atividade de criptografar um texto e/ou descriptografar os códigos existentes no *QR-Code*. Esta classe não é responsável pela geração de *QR-Code*, mas concentra-se exclusivamente a criptografia.

Trabalha apenas com dados no formato *String*, porém o seu retorno aos demais métodos é formato *byte*, consequente há a necessidade de métodos posteriores trabalharem desta forma, simplificando os processos e determinando singularidade para o que cada método deve realizar.

4.1.2 Notificação

Como visto no Capítulo 3, na Seção *Padrão de Leitura de Múltiplas Câmeras Autenticadoras*, o padrão possui pelo menos duas *interfaces*. Uma das *interfaces* é conhecida como **Observador**, daí o nome atribuído ao padrão *Observer*. As classes que implementarem essa *interface*, que por sua vez conterá um método de atualização, devem sempre ser notificadas quando qualquer atualização do estado dos objetos observados ocorrerem. Já a outra *interface* adotada é a denotada *Observada*, onde qualquer classe que adotar este tipo será vista como um objeto observado continuamente. Este tipo de classe geralmente possui como atributo uma lista dos observadores associados, para que, quando alterado os estados, possam ser notificados.

No Apêndice B - módulo B.1 e B.2, verifica-se em código Java as *interfaces* propostas: Observador e Observado. A *interface* **Observador**, é necessário pelo menos a existência do método *atualizar()*. Usado para conter a regra de atualização específica de cada observador, definindo um contrato de implementação obrigatório para com os objetos observadores, que após as notificações de mudança no objeto observado, seja realizado algum tipo de processo.

O segundo caso, a *interface* **Observado** contém três métodos que serão implementados nos objetos que adotarem este padrão. O primeiro método é *incluirObservador(Observador obs)*, este realiza a adição de um observador através de um parâmetro à lista de observadores. O segundo método é *removerObservador(Observador obs)*, que realiza ação contrária a anterior, ou seja, remove um observador passado por parâmetro da lista de observadores do objeto observado, e por fim, o último método existente da interface é o *notificarObservadores()*, que serve para notificar todos os observadores caso haja alguma mudança no objeto observado.

Chamaremos as classes que implementam tais interfaces de:

- **Autenticador**: quando este implementa o padrão do **Observador** e;
- **Sistema**: que este implementará o padrão do **Observado**.

A classe *Autenticador.class*, possui seu construtor que incrementa observadores. Além deste construtor, a classe também possui a implementação do método *atualizar(String textoCriptografado)*, que é a mensagem que será enviada sempre ao observador sobre a mudança de estado ou atividade realizada pelo objeto observado.

Em *Sistema.class*, temos os métodos que a classe observada assume. O método *notificarObservadores()* realiza alterações e notifica aos observadores, sendo assim, o processo de autenticidade encontra-se neste método.

4.1.3 QR-Code

Voltada especificamente para projeção do *QR-Code*, de acordo com a informação recebida, a classe *QR-Code.class*, contida no Apêndice C - módulo C.1, foi implementada a partir de

classes e bibliotecas disponibilizadas pelo projeto *ZXing*. Este classe garante a criação de um *QR-Code* que atende a todas as especificações necessárias, e pode ser lido em qualquer plataforma ou aplicativo leitor de *QR-Code*.

4.1.4 Persistência

Duas classes compõe este bloco, disponíveis no Apêndice D - módulo D.1 e D.2. *Armazena.class* e *Recupera.java*, são classes responsáveis pela geração de um par de chaves totalmente novas e utilização de chaves que já estejam disponíveis ao sistema, respectivamente.

Utilizando a criptografia RSA, estas chaves são armazenadas localmente, porém a classe *Armazena.java* e seus métodos não serão disponíveis livremente ao usuário. Posteriormente, a solução para administração das chaves será via *hardware*, como gravação da chave privada em um dispositivo de armazenamento externo.

Os métodos localizados em *Recupera.java* são destinados a recuperação das chaves privadas e públicas, para poder descriptografar e criptografar, respectivamente, o conteúdo que estará contidos nos *QR-Codes*

4.1.5 Gerador e Leitor

Estas classes são responsáveis pela geração e leitura de *QR-Codes*.

Em Apêndice E - módulo E.1 e E.2, estão localizados as classes *GeraQR.java* e *LerQR.java*. A primeira classe citada, comunica-se diretamente com a classe *QR-Code.java*, contida no Apêndice C - módulo C.1, para o fornecimento dos *QR-Codes* de autenticação. A segunda classe, mantém iteração continuamente com o módulo B.3 e B.4, que são os objetos Observados e Observadores do sistema.

4.2 Arduíno

A plataforma utilizada para criação do sistema foi o *Linux Debian*. A instalação do *Arduíno IDE* na plataforma *Linux* é considerada simples. Existe um pré-requisito necessário antes de instalar e executar o *Arduino IDE*. Esse pré-requisito envolve ter um ambiente de execução *Java* já instalado.

O download do *Arduíno IDE* é a partir do site do *Arduíno*. A instalação do *Arduíno IDE* é bastante intuitiva com caixas de dialogos orientando o usuário.

Para codificação, utilizamos um cabo USB conectando o *Arduino Uno* ao computador. O segundo passo é selecionar a porta serial (que está debaixo da opção de menu **Board**). No Linux e Mac, será algo mais na direção de */dev/tty.xxxxx* ou algo similar.

Com o ambiente para codificação do Arduíno preparado, foi possível realizar a codificação. O código é apresentado no Apêndice F - módulo F.1, com comunicação do *Arduíno* com o sistema via comunicação *socket* e trava eletromagnética.

4.3 Protótipo do Sistema e Testes

A elaboração do sistema utilizou persistência de usuários a partir de tecnologias como **JPA** e formatação da interface com **JavaFX** para aplicação em ambiente *standalone*[1]. No sistema foi adotado apenas a inserção do nome do usuário como informação necessária a ser codificada, dispensando a inclusão de outras informações como abordadas no Capítulo 3, na Seção *Principais Características e Comportamentos do Sistema*, em *Usuários Submetidos ao Sistema*, em que foi explorado várias características que o usuário pode ter.

4.3.1 Interface do Sistema

Realizado a etapa de codificação, foi realizada a criação da interface do sistema utilizada pelo usuário. A Figura 4.1 mostra o layout gerado para o sistema proposto.

Figura 4.1: *Protótipo do Sistema*

A interface é composta por 3 (três) botões, com finalidades de cadastrar o usuário ao sistema (botão *Cadastrar*), remover o usuário do sistema (botão *Remover*) e acionar a câmera que realizará a autenticação (botão *Autenticador de QR-Code*).

Para realizar o cadastro de um usuário, basta a inserção do nome do usuário no campo de texto identificado por *Nome* e clicar no botão *Cadastrar*. A ação deste botão irá adicionar o nome do usuário ao banco de dados do sistema, e irá gerar um *QR-Code*, com a informação do usuário, criptografado. Como podemos observar na Figura 4.2.

Ao clicar no botão *Autenticador de QR-Code*, será acionado a câmera, como mostra a Figura 4.3. A partir deste momento, o usuário poderá apresentar o *QR-Code*. A imagem é capturada e exibida na janela do ponto 1 em destaque na Figura 4.3. No ponto 2, temos

[1] Funcionamento de forma autônoma.

Figura 4.2: *Adição de usuário*

todos os retornos de ação realizada pelo sistema, como também de possíveis erros advindos de uma ação inesperado do administrador, como apertar o botão *Cadastrar*, mas não ter inserido o nome desejado para tal ação.

Figura 4.3: *Protótipo do Sistema - Autenticador de QR-Code*

Por fim, na Figura 4.4 temos a ação de remoção do usuários do banco de dados de pessoas que podem acessar o local restrito.

Ao clicar neste botão, será exibido uma nova janela que exibe todos os usuários que tem permissão de acesso aquele local. Caso o administrador tenha desejo de retirar o usuário desta lista, basta inserir o seu código de identificação no espaço destinado para isto e clicar no botão *Remover*, caso contrário, basta fechar esta janela. Se o usuário for removido, será retornado ao administrador uma mensagem de remoção do usuário.

4.3.2 Tranca Eletromagnética

Como visto no Capítulo 3, na Seção *Tranca Magnética para Portas*, um dispositivo leitor de informações externas envia um pulso ao circuito. Isto aciona o relé liberando a trava de entrada, permitindo acesso ao usuário no ambiente.

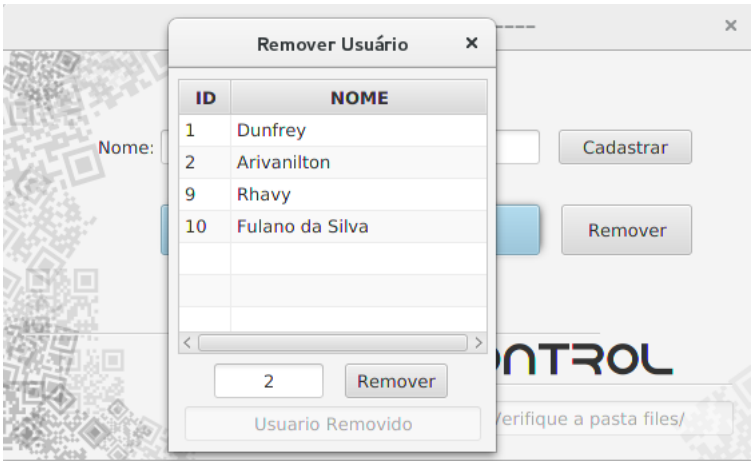


Figura 4.4: *Remoção de usuários do sistema*

Na Figura 4.5 temos a trava eletromagnética ligada ao relé, que recebe os pulso e aciona a tranca.

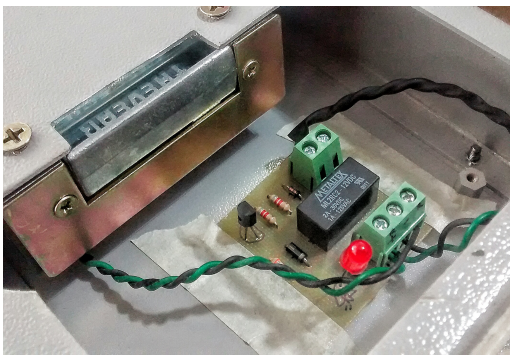


Figura 4.5: *Tranca eletromagnética e relé*

Para fabricação do circuito relé foram utilizados os componentes conforme Tabela 4.1.

Tabela 4.1: *Tabela de componentes para fabricação de relé*

COMPONENTES	QUANTIDADE
Resistor 10K	2
Resistor 3K3	1
Diodo 1N4148	3
LED	1
Semi-Condutor BC337	1
Relé MetalTex MH2RC3-12v	1

4.3.3 Custo Estimado da Solução Proposta

O custo estimado para o sistema proposto está disposto na Tabela 4.2, considerando os materiais utilizados e a produção do kit de autenticação de usuários - valor total de R$ 64,47. Alguns equipamentos como o *Arduino UNO* e *Shield Ethernet* foram adquiridos no comér-

cio eletrônico internacional no site *AliExpress*[2]. O relé foi confeccionado com componentes adquiridos no comércio local.

Ademais, deve-se considerar o valor de câmera IP, em que o valor não foi acrescentado por ter sido utilizado a webcam do próprio servidor, que no nosso caso foi o notebook em que rodou o sistema e realizou a autenticação.

Tabela 4.2: *Tabela de preços de produto por ponto de autenticação Code Control*

CUSTO DA SOLUÇÃO PROPOSTA	
PRODUTO	PREÇO UNITÁRIO
Arduíno UNO	R$ 31,45
Shield Ethernet	R$ 24,53
Relé	R$ 8,49

Realizando ainda uma comparação de preços com algumas soluções existentes, baseando-se em seus preços médios atuais do ano de 2015, em mercado brasileiro, é possível levantar os valores por pontos de autenticação mostrados na Tabela 4.3

Tabela 4.3: *Tabela de preços de produto por ponto de autenticação de outros sistemas*

Pontos de Autenticação	**Valor**
Relógio Eletrônico de Ponto Gertec Marque Ponto (Biométrico)	R$ 1.990,00
Controle De Acesso Biométrico Henry Argos 1900 Digitais	R$ 1.490,00
Fechadura digital Intelbras com acesso por senha ou proximidade FR200	R$ 799,00
Controle De Acesso Digital Biométrico e Cartão RFID	R$ 920,00
Controle De Acesso Cartão Proximidade	R$ 564,90

[2]www.aliexpress.com

Capítulo 5

Considerações Finais e Sugestões de Trabalhos Futuros

5.1 Considerações Finais

Este trabalho teve por objetivo fornecer um sistema de controle de acesso físico de baixo custo, que possa ser acessível a um pessoa, ou empresa, com pouco recurso monetário mas que precise do serviço que provenha sigilo físico. A ideia principal consiste em utilizar *QR-Codes*, pois são bastante populares nos dias atuais, junto com criptografia, garantindo portabilidade e segurança.

A criação do protótipo do sistema apresentados no Capítulo 4, baseado na teoria apresentada no Capítulo 2, mostram que ouve sucesso neste projeto e o sistema produzido é uma boa opção para ser aplicado em ambientes empresariais e em ambientes domésticos, devido simplicidade de uso e segurança que pode ser fornecida.

Além da segurança e baixo custo, caso o ambiente necessite de mais pontos autenticadores, basta a conexão da câmera autenticadora com a base de dados e adição de um ponto de liberação que tem-se o controle de acesso do ambiente.

Por fim, o projeto vislumbra a possibilidade de ser modularizado afim de acrescentar novas técnicas de autenticação, como biométrica, adaptando o projeto a outros cenários, seja controle de acesso físico ou a segurança de dados que são transmitidos em uma rede ethernet.

5.2 Sugestões de Trabalhos Futuros

Os resultados obtidos no trabalho indicam a eficácia do sistema de controle de acesso físico. Entretanto, foram identificadas algumas limitações dos resultados que vislumbram possibilidades para trabalhos futuros, destacando-se:

- Análise de segurança a partir de envio de chaves públicas junto com o QR-Code criptografado do usuário;

- Utilização de criptografia via *hardware*;
- Utilização de esteganografia, que visa esconder um *QR-Code* sob uma imagem aparentemente sem nenhum valor ou significado. Porém, para implementação desta técnica, é preciso substituir as câmeras por pontos de recepção de dados digitais, para ser possível a leitura dos *bits* e reconstrução da sobreposição de imagens digitalmente;
- Substituir a utilização dos *QR-Codes* por pontos NFC com criptografia, sendo necessário apenas aproximação do elemento, e não leitura digital através de uma câmera.

Apêndice A

Criptografia

A.1 Criptografia.class

```
1 public class Criptografia {
2
3     private RecuperaChaves rc = new RecuperaChaves();
4     private PublicKey PU;
5     private PrivateKey PK;
6
7     private final String RSA = "RSA";
8
9     public Criptografia(){
10        PU = rc.RecuperaPublica();
11        PK = rc.RecuperaPrivada();
12    }
13
14    private byte[] cifra(String text, PublicKey chavePublica) throws
          Exception {
15        Cipher cipher = Cipher.getInstance(RSA);
16        cipher.init(Cipher.ENCRYPT_MODE, chavePublica);
17        return cipher.doFinal(text.getBytes());
18    }
19
20    public final String cifra(String text) {
21        try {
22            return byte2hex(Criptografia.this.cifra(text, PU));
23        } catch (Exception e) {
24            e.printStackTrace();
25        }
26        return null;
27    }
28
29    public final String decifra (String data) {
30        try {
```

```
31              return new String(decifra(hex2byte(data.getBytes())));
32          } catch (Exception e) {
33              e.printStackTrace();
34          }
35          return null;
36      }
37
38      private byte[] decifra (byte[] src) throws Exception {
39          Cipher cipher = Cipher.getInstance(RSA);
40          cipher.init(Cipher.DECRYPT_MODE, PK);
41          return cipher.doFinal(src);
42      }
43
44      public String byte2hex(byte[] dado_byte) {
45          String dado_hexa = "";
46          String stmp = "";
47          for (int n = 0; n < dado_byte.length; n++) {
48              stmp = Integer.toHexString(dado_byte[n] & 0xFF);
49              if (stmp.length() == 1) {
50                  dado_hexa += ("0" + stmp);
51              } else {
52                  dado_hexa += stmp;
53              }
54          }
55          return dado_hexa.toUpperCase();
56      }
57
58      public byte[] hex2byte(byte[] dado_hexa) {
59          if ((dado_hexa.length % 2) != 0) {
60              throw new IllegalArgumentException("Ola.");
61          }
62
63          byte[] dado_byte = new byte[dado_hexa.length / 2];
64
65          for (int n = 0; n < dado_hexa.length; n += 2) {
66              String item = new String(dado_hexa, n, 2);
67              dado_byte[n / 2] = (byte) Integer.parseInt(item, 16);
68          }
69          return dado_byte;
70      }
71  }
```

Apêndice B

Padrão *Observer*

B.1 Observador.class

```
1 public interface Observador {
2
3   public void atualizar(String string);
4
5 }
```

B.2 Observado.class

```
1 public interface Observado {
2
3   public void incluirObservador(Observador obs);
4   public void removerObservador(Observador obs);
5   public void notificarObservadores();
6
7 }
```

B.3 Autenticador.class

```
1 public class Autenticador implements Observador {
2
3   public Autenticador(Observado obs){
4     obs.incluirObservador(this);
5   }
6
7   @Override
8   public void atualizar(String textoCriptografado) {
9     System.out.println("Usuario tem acesso: "+textoCriptografado);
10  }
```

```
11
12 }
```

B.4 Sistema.class

```
1 public class Sistema implements Observado {
2
3   private List<Observador> observadores;
4   private String textoCriptografado;
5
6   public Sistema() {
7     this.observadores = new ArrayList<>();
8   }
9
10  public void setTextoCriptografado(String leituraDoQrcode) {
11    this.textoCriptografado = leituraDoQrcode;
12  }
13
14  @Override
15  public void incluirObservador(Observador obs) {
16    this.observadores.add(obs);
17  }
18
19  @Override
20  public void removerObservador(Observador obs) {
21    int ind = this.observadores.indexOf(obs);
22    if (ind >= 0) {
23      this.observadores.remove(obs);
24    }
25  }
26
27  @Override
28  public void notificarObservadores() {
29    for (Observador observador : observadores) {
30
31    if(textoCriptografado.equals("Carlos")){
32      textoCriptografado = "Permitido";
33    } else {textoCriptografado = "Negado";}
34
35    observador.atualizar(this.textoCriptografado);
36    }
37  }
38 }
```

B.5 Desbloqueio.class

```
1 public class Desbloqueio {
2
3     public void executa() throws Exception {
4         Socket clientSocket;
5         //IP do Arduino e Porta de comunicacao adotada no sistema.
6         clientSocket = new Socket("192.168.0.2", 37776);
7         try (DataOutputStream outToServer =
8                 new DataOutputStream(clientSocket.getOutputStream())
9             ) {
10                 outToServer.writeBytes("P");
11                 clientSocket.close();
12             }
13     }
14 }
```

Apêndice C

QR-Code

C.1 QRCode.class

```
1 public class QRModulo {
2
3     private String caminhoArquivo = "files/ImagemQR.png";
4     private String tipoArquivo = "png";
5     private int tamanho = 300;
6
7     public String getCaminhoArquivo() {
8         return this.caminhoArquivo;
9     }
10
11    public void setCaminhoArquivo(String caminhoArquivo) {
12        this.caminhoArquivo = caminhoArquivo;
13    }
14
15    public void inicializa() {
16        System.out.println("Inicializado o programa de gerar QR-Codes");
17    }
18
19    public void geraQR(String texto) {
20
21        File myFile = new File(caminhoArquivo);
22        try {
23            Hashtable<EncodeHintType, ErrorCorrectionLevel> hintMap = new
                  Hashtable<>();
24            hintMap.put(EncodeHintType.ERROR_CORRECTION,
                  ErrorCorrectionLevel.L);
25            QRCodeWriter qrCodeWriter = new QRCodeWriter();
26
27            BitMatrix byteMatrix = qrCodeWriter.encode(texto,
                  BarcodeFormat.QR_CODE, tamanho, tamanho, hintMap);
28            int CrunchifyWidth = byteMatrix.getWidth();
```

```
29              BufferedImage image = new BufferedImage(CrunchifyWidth,
                    CrunchifyWidth,
30                      BufferedImage.TYPE_INT_RGB);
31              image.createGraphics();
32
33              Graphics2D graphics = (Graphics2D) image.getGraphics();
34              graphics.setColor(Color.WHITE);
35              graphics.fillRect(0, 0, CrunchifyWidth, CrunchifyWidth);
36              graphics.setColor(Color.BLACK);
37
38              for (int i = 0; i < CrunchifyWidth; i++) {
39                  for (int j = 0; j < CrunchifyWidth; j++) {
40                      if (byteMatrix.get(i, j)) {
41                          graphics.fillRect(i, j, 1, 1);
42                      }
43                  }
44              }
45              ImageIO.write(image, tipoArquivo, myFile);
46          } catch (WriterException e) {
47              e.printStackTrace();
48          } catch (IOException e) {
49              e.printStackTrace();
50          }
51      }
52
53  }
```

Apêndice D

Persistência

D.1 Armazena.class

```
1 public class Armazena {
2
3   private final String algGenChave = "RSA";
4   private final String ARQUIVO_CHAVE_PRIVADA = "files/private.key";
5   private final String ARQUIVO_CHAVE_PUBLICA = "files/public.key";
6
7   public Armazena(){
8     System.out.println("Armazenando chaves.");
9   }
10
11  public void geraChaves() {
12    try {
13      final KeyPairGenerator keyGen = KeyPairGenerator.getInstance(
            algGenChave);
14      keyGen.initialize(2048);
15      final KeyPair chave = keyGen.generateKeyPair();
16      File privateKeyFile = new File(ARQUIVO_CHAVE_PRIVADA);
17      File publicKeyFile = new File(ARQUIVO_CHAVE_PUBLICA);
18      if (privateKeyFile.getParentFile() != null) {
19        privateKeyFile.getParentFile().mkdirs();
20      }
21      privateKeyFile.createNewFile();
22      if (publicKeyFile.getParentFile() != null) {
23        publicKeyFile.getParentFile().mkdirs();
24      }
25      publicKeyFile.createNewFile();
26      try (
27        ObjectOutputStream publicKey_oos = new ObjectOutputStream(
28        new FileOutputStream(publicKeyFile))) {
29          publicKey_oos.writeObject(chave.getPublic());
30      }
```

```
31        try (
32          ObjectOutputStream privateKey_oos = new ObjectOutputStream(
33          new FileOutputStream(privateKeyFile))) {
34            privateKey_oos.writeObject(chave.getPrivate());
35        }
36      } catch (NoSuchAlgorithmException | IOException e) {  }
37    }
38 }
```

D.2 Recupera.class

```
1 public class Recupera {
2
3   private final String ARQUIVO_CHAVE_PRIVADA = "files/private.key";
4   private final String ARQUIVO_CHAVE_PUBLICA = "files/public.key";
5
6   public PublicKey RecuperaPublica() {
7     if (verificaChaves()) {
8       try {
9         ObjectInputStream inputStream = null;
10
11        inputStream = new ObjectInputStream(new FileInputStream(
              ARQUIVO_CHAVE_PUBLICA));
12        final PublicKey publicKey = (PublicKey) inputStream.readObject();
13        return publicKey;
14      } catch (IOException | ClassNotFoundException e) {  }
15    } return null;
16  }
17
18  public PrivateKey RecuperaPrivada() {
19    if (verificaChaves()) {
20      try {
21        ObjectInputStream inputStream = null;
22
23        inputStream = new ObjectInputStream(new FileInputStream(
              ARQUIVO_CHAVE_PRIVADA));
24        final PrivateKey privateKey = (PrivateKey) inputStream.readObject
              ();
25        return privateKey;
26      } catch (IOException | ClassNotFoundException e) {  }
27    }   return null;
28  }
29
30  public boolean verificaChaves() {
31    File privateKey = new File(ARQUIVO_CHAVE_PRIVADA);
32    File publicKey = new File(ARQUIVO_CHAVE_PUBLICA);
```

```
33      return privateKey.exists() && publicKey.exists();
34    }
35 }
```

Apêndice E

Gerador e Leitor

E.1 GeraQR.class

```
1 public class GeraQR {
2
3   public void SpynGeraQR(String texto) {
4     Criptografia c = new Criptografia();
5     byte[] textoCriptografado = c.cifrar(texto.getBytes());
6
7     QRCode qrcode = new QRCode();
8     qrcode.inicializa();
9     qrcode.geraQR(new String(textoCriptografado));
10  }
11 }
```

E.2 LerQR.class

```
1 public class LerQR extends JFrame implements Runnable, ThreadFactory {
2
3   private Executor executor = Executors.newSingleThreadExecutor(this);
4   private Webcam webcam = null;
5   private WebcamPanel panel = null;
6   private Sistema notificador;
7
8   public LerQR(Sistema notificador) {
9     super();
10
11    this.notificador = notificador;
12    Dimension size = WebcamResolution.QVGA.getSize();
13    webcam = Webcam.getWebcams().get(0);
14    if(webcam != null){
15      webcam.setViewSize(size);
16    }
```

```
17      panel = new WebcamPanel(webcam);
18      panel.setPreferredSize(size);
19      add(panel);
20      pack();
21      executor.execute(this);
22    }
23
24    @Override
25    public void run() {
26      webcam.open();
27      do {
28        try {
29          Thread.sleep(100);
30        } catch (InterruptedException e) {
31          e.printStackTrace();
32        }
33        Result result = null;
34        BufferedImage image = null;
35        if (webcam.isOpen()) {
36          if ((image = webcam.getImage()) == null) {
37            continue;
38          }
39          LuminanceSource source = new BufferedImageLuminanceSource(image);
40          BinaryBitmap bitmap = new BinaryBitmap(new HybridBinarizer(source)
                ));
41          try {
42            result = new MultiFormatReader().decode(bitmap);
43          } catch (NotFoundException e) { }
44        }
45        if (result != null) {
46            notificador.setTextoCriptografado(result.getText());
47            notificador.notificarObservadores();
48        }
49      } while (true);
50    }
51
52    @Override
53    public Thread newThread(Runnable r) {
54      Thread t = new Thread(r, "Sistema Spyn");
55      t.setDaemon(true);
56      return t;
57    }
58
59  }
```

Apêndice F

Arduíno

F.1 AcionarTranca

```
1 #include <SPI.h>
2 #include <Ethernet.h>
3
4   byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
5   byte ip[] = { 192, 168, 0, 2 }; // ip arduino
6   EthernetServer server (37776); // servidor na porta 37776
7
8   const int led = 2;
9
10  int valorLido = 0;
11
12  byte pinoSlaveSelect = 10; // pino de controle do Ethernet Shield
13
14  void setup () {
15    Serial.begin(9600);
16    while (!Serial) {
17    ; // Se nao conectar serial
18    }
19
20    // Comeca comunicacao Ethernet
21    Ethernet.begin( mac, ip);
22    server.begin();
23
24    pinMode(pinoSlaveSelect, OUTPUT); // habilitando o uso do SPI
25
26    pinMode(led, OUTPUT); // Porta de saida do LED
27  }
28
29  void acendeLed(){
30    digitalWrite(led, HIGH);    // turn the LED on (HIGH is the voltage
        level)
```

```
31      delay(5000);                // wait for a second
32      digitalWrite(led, LOW);     // turn the LED off by making the voltage
            LOW
33    }
34
35    void loop() {
36      EthernetClient client = server.available();
37
38      if (client) { // se receber um caracter
39
40      valorLido = client.read();
41
42      if (valorLido == 'P'){ // Permitido
43        acendeLed();
44      }
45
46    }
47
48 }
```

Referências Bibliográficas

[1] B. MENKUS, "Physical security: Selecting an access control system," *Computers & Security*, pp. 201–205, 1989. 1

[2] P. L. HOVING, "To install an access control system," *Activities and Checklists. Computers & Security*, vol. 2, pp. 163–170, jun 1983. 1

[3] R. Hans, "Using a biometric system to control access and exit of vehicles at shopping malls in south africa," in *Engineering Technology and Technopreneuship (ICE2T), 2014 4th International Conference on*, pp. 148–151, Aug 2014. 1, 2

[4] D. DZUNG, "Security for industrial communication systems," *Procedings of the IEEE*, vol. 93, pp. 1152–1177, jun 2005. 1

[5] M. Lehtonen, F. Michahelles, and E. Fleisch, "Trust and security in rfid-based product authentication systems," *Systems Journal, IEEE*, vol. 1, pp. 129–144, Dec 2007. 1, 2

[6] M. A. KOWTKO, "Biometric authentication for older adults. systems," *Applications and Technology Conference (LISAT). Long Island: IEEE*, pp. 1–6, 2014. 1

[7] F. Civico and A. Peinado, "Low complexity smart card-based physical access control system over ip networks," in *Electrotechnical Conference, 2004. MELECON 2004. Proceedings of the 12th IEEE Mediterranean*, vol. 2, pp. 799–802 Vol.2, May 2004. 2

[8] C.-H. Huang and S.-C. Huang, "Rfid systems integrated otp security authentication design," in *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2013 Asia-Pacific*, pp. 1–8, Oct 2013. 2

[9] S. Chague, B. Droit, O. Boulanov, S. Yanushkevich, V. Shmerko, and A. Stoica, "Biometric-based decision support assistance in physical access control systems," in *Bio-inspired Learning and Intelligent Systems for Security, 2008. BLISS '08. ECSIS Symposium on*, pp. 11–16, Aug 2008. 2

[10] A. Bhanushali, B. Mange, H. Vyas, H. Bhanushali, and P. Bhogle, "Article: Comparison of graphical password authentication techniques," *International Journal of Computer Applications*, vol. 116, pp. 11–14, April 2015. Full text available. 2

[11] E. D. P. Fanfara and M. Dufala, "Usage of asymmetric encryption algorithms to enhance the security of sensitive data in secure communication," *10th IEEE Jubilee International Symposium on Applied Machine Intelligence and Informatics*, jan 2012. 5

[12] W. Stallings, *Criptografia e segurança de redes: princípios e práticas*. Pearson Prentice Hall, 2008. 5, 10

[13] W. Diffie and M. E. Hellman, "Multiuser cryptographic techniques," in *Proceedings of the June 7-10, 1976, National Computer Conference and Exposition*, AFIPS '76, (New York, NY, USA), pp. 109–112, ACM, 1976. 6

[14] D. Wave, "What is a qr code," apr 2015. 6, 7

[15] B. do Brasil, "Bb-code," jun 2015. 13

[16] D. Conde-Lagoa, E. Costa-Montenegro, F. Gonzalez-Castao, and F. Gil-Castiñeira, "Secure etickets based on qr-codes with user-encrypted content," in *Consumer Electronics (ICCE), 2010 Digest of Technical Papers International Conference on*, pp. 257–258, Jan 2010. 13

[17] W. Li, F. Duan, B. Chen, J. Yuan, J. Tan, and B. Xu, "Mobile robot action based on qr code identification," in *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*, pp. 860–865, Dec 2012. 13

[18] W.-P. Fang, "Offline qr code authorization based on visual cryptography," in *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2011 Seventh International Conference on*, pp. 89–92, Oct 2011. 14

[19] D.-J. Cho, "Study on method of new digital watermark generation using qr-code," in *Broadband and Wireless Computing, Communication and Applications (BWCCA), 2013 Eighth International Conference on*, pp. 585–588, Oct 2013. 14

[20] H.-C. Huang, F.-C. Chang, and W.-C. Fang, "Reversible data hiding with histogram-based difference expansion for qr code applications," *Consumer Electronics, IEEE Transactions on*, vol. 57, pp. 779–787, May 2011. 14

[21] A. M. Masuda, "Tópicos de corpos finitos com aplicações em criptografia e teoria de códigos," *26 Colóquio Brasileiro de Matemática*, jun 2007. 14

[22] ICP-BRASIL, "Infraestrutura de chaves públicas brasileira," apr 2015. 14

[23] H.-M. Sun, M.-E. Wu, W.-C. Ting, and M. Hinek, "Dual rsa and its security analysis," *Information Theory, IEEE Transactions on*, vol. 53, pp. 2922–2933, Aug 2007. 14

[24] Skype, "Chamadas de voz e vídeo via skype," apr 2015. 15

[25] C.-H. Lin, J.-C. Liu, C.-C. Li, and P.-W. Chu, "Parallel modulus operations in rsa encryption by cpu/gpu hybrid computation," in *Information Security (ASIA JCIS), 2014 Ninth Asia Joint Conference on*, pp. 71–75, Sept 2014. 15

[26] E. Gamma, R. Helm, R. Johnson, and J. M. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1 ed., 1994. 21