UNIVERSIDADE ESTADUAL DO NORTE DO PARANÁ
CAMPUS LUIZ MENEGHEL - CENTRO DE CIÊNCIAS TECNOLÓGICAS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

FELIPE CHIAROTTI GIRON

ANÁLISE DA IMPLEMENTAÇÃO DO SISTEMA CRIPTOGRÁFICO RSA PARA A PLATAFORMA ARDUINO

BANDEIRANTES–PR

2018

FELIPE CHIAROTTI GIRON

ANÁLISE DA IMPLEMENTAÇÃO DO SISTEMA CRIPTOGRÁFICO RSA PARA A PLATAFORMA ARDUINO

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual do Norte do Paraná para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Me. Wellington Aparecido Della Mura

BANDEIRANTES–PR

2018

FELIPE CHIAROTTI GIRON

ANÁLISE DA IMPLEMENTAÇÃO DO SISTEMA CRIPTOGRÁFICO RSA PARA A PLATAFORMA ARDUINO

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual do Norte do Paraná para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

Prof. Me. Wellington Aparecido Della Mura
Universidade Estadual do Norte do Paraná
Orientador

Prof. Me. Carlos Eduardo Ribeiro
Universidade Estadual do Norte do Paraná

Prof. Ronaldo Cesar Mengato Júnior
Universidade Estadual do Norte do Paraná

Bandeirantes–PR, 21 de novembro de 2018

Dedico este trabalho em razão de Cintia Chiarotti, responsável por todo o apoio sempre que necessitei, e por ser a pessoa que me tornou capaz de alcançar qualquer objetivo.

AGRADECIMENTOS

Os principais agradecimentos vão ao meu orientador Prof. Me. Wellington Aparecido Della Mura, responsável por todo o apoio e direcionamento para que este trabalho fosse concluído. Aos professores que me proporcionaram a capacidade de adquirir aprendizado para seguir em frente na vida. Agradeço à UENP por proporcionar as experiências de vida que jamais serão esquecidas para que fosse necessário a conclusão de mais um objetivo em minha história. Por final, os agradecimentos aos meus colegas de classe restantes, José Maria Clementino Junior´, William Sdayle, André Luiz Lima e Rafael Ribeiro, esses os quais permaneceram na luta junto à mim para o alcance do primeiro grande objetivo em nossas vidas.

"O sucesso nasce do querer,
da determinação e persistência em se chegar a um objetivo.
Mesmo não atingindo o alvo,
quem busca e vence obstáculos, no mínimo fará coisas admiráveis.
(José de Alencar)

CHIAROTTI, F. G.. **Análise da implementação do sistema criptográfico RSA para a plataforma Arduino**. 78 p. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual do Norte do Paraná, Bandeirantes–PR, 2018.

RESUMO

O presente trabalho tem como objetivo analisar a implementação do sistema criptográfico RSA para a plataforma Arduino utilizando um microcontrolador AVR de 8 bits. O trabalho é proposto através de um estudo sobre qual o motivo dos problemas de segurança em IoT, quais são suas principais limitações, e quais as consequências geradas com isso. Ao verificar as razões e fazer um levantamento dessas informações e cruzá-las com o conhecimento de segurança da informação focado em criptografia e proteção de dados, obtêm-se maneiras de se implementar o sistema criptográfico de forma que se encaixe nas limitações presentes na plataforma Arduino. Com o resultado de uma implementação funcional do RSA em Arduino, adquire-se dados de tempo relacionados com o tamanho das mensagens a serem utilizadas e também o dos pares de chave, chegando à conclusão real das dificuldades encaradas por dispositivos de baixa capacidade.

Palavras-chave: Segurança. Criptografia. Internet das Coisas.

CHIAROTTI, F. G.. **Analysis of the portability of RSA cryptographic system for Atmel AVR Arduino**. 78 p. Final Project (Bachelor of Science in Computer Science) – State University Northern of Parana , Bandeirantes–PR, 2018.

ABSTRACT

The purpose of this work is to analyze the portability of the RSA cryptographic system for Arduino platform using an 8-bit AVR microcontroller. The work is proposed through an intense bibliographical review and study about the cause of the security problems in the Internet of Things, this includes which are the main problems, the result of this, and what are the consequences of this. By checking the reasons and making a survey of this information and crossing it with the security knowledge of the traditional information and its models and implementing the cryptographic system to fit the limitations present in the Arduino platform. With the result of a functional implementation of RSA in Arduino, time data are obtained relating to the size of the messages to be used and also the size of the key pairs, arriving at the real conclusion of the difficulties faced by low capacity devices.

Keywords: Security. Encryption. Internet of Things.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

IoT	*Internet of Things*
DoS	*Deny of Service*
DDoS	*Distributed Deny of Service*
CPU	*Central Processing Unit*
RAM	*Random Access Memory*

SUMÁRIO

1 INTRODUÇÃO

Muito se fala sobre a Internet das Coisas, ou do inglês *Internet of Things*. Toma-se esse conceito como novo e muito recente, mas isso só se deve ao fato da popularização da conexão com a Internet. Sendo uma tecnologia muito inovadora, à ponto de ser uma nova revolução industrial [2]. Tendo uma grande importância por transformar a Internet como é hoje, em um futuro totalmente integrado [3]

A Internet das Coisas não possui uma definição exata, sendo frequentemente discutida pelo fato de não se ter um consenso do termo [4]. Mas o fato é que a Internet das Coisas é um acumulado de objetos e conectividade [5] o qual possibilita dispositivos conectarem-se à internet através de seus sensores [6]. Ao analisar a palavra Internet das Coisas, nota-se duas palavras-chave, sendo elas "Internet" e "Coisas". O primeiro impõe uma visão de IoT como orientada a redes, já o segundo tem um foco em objetos genéricos à serem integrados. Com a relação entre os dois termos, pode-se interpretar como uma "rede universal de objetos interconectados" [7].

Esse paradigma tecnológico estende a interação entre humanos e aplicações a um novo nível, devido à conectividade que oferece às "coisas"[4]. Um objeto ("coisa") em IoT é definido como qualquer máquina, dispositivo, aplicação, computador e objetos físicos ou virtuais que possuem uma comunicação com a internet e podem consumir, passar ou ter acesso à informação digital [4]. As "coisas" tornam-se objetos inteligentes devido à sua conectividade dada pela internet, os quais são capazes de fazerem a comunicação e operação entre si ou até humanos.

Devido à combinação de diversas "coisas", os quais são objetos (normalmente do mundo real) e com a Internet, é possível gerar novas soluções a partir de objetos simples. Dessa maneira, a Internet das Coisas, consegue agregar valor e funções a um simples objeto. Ao combinar um objeto físico do mundo real, com a tecnologia, normalmente presente na forma de *Hardware* e *Software* obtém-se o resultado como um serviço baseado em "coisas" [8].

Apesar da Internet das Coisas possuir tantas vantagens e agregar valor à praticamente qualquer objeto, ser uma possível tecnologia tão revolucionária a torna um paradigma com muitos desafios e dificuldades. No mundo de hoje essa tecnologia ainda apresenta diversos problemas e desafios, muitos deles relacionados à segurança e privacidade [7].

A Internet das Coisas ainda engatinha no quesito segurança, o qual aqui é referido à provisão básica de serviços de segurança, incluindo confidencialidade, autenticação, integridade, autorização, impossibilidade de negação e disponibilidade [9]. Devido ao intenso

crescimento de dispositivos IoT portando uma segurança pobre, torna-os mais visados e vulneráveis a ataques cibernéticos [10]. Segundo Garcia-Morchon et al.[9], os principais problemas com segurança em IoT se devem à limitação de recursos existentes nessa tecnologia. Muitos dos dispositivos são utilizados à base de baterias, possuem memória limitada, baixo poder de processamento e baixa capacidade de armazenamento. Devido à fatores como estes, há a dificuldade de se aplicar mecanismos de segurança tais como criptografia. Também verifica-se muitas dificuldades quando o assunto é segurança devido à fatores que são muito recorrentes em IoT como interoperabilidade, independência, diversidade e mobilidade, os quais são fatores que trazem dificuldade para a padronização e aplicação dos esquemas padrões de segurança. A impossibilidade de se usar um mecanismo padrão de segurança em um dispositivo IoT se dá devido à todas essas limitações, pois os padrões de segurança tradicionais não foram pensados para operar com poucos recursos [11]. Em seu artigo, [9] diz que todas essas questões relacionadas à limitação, fazem com que dispositivos IoT precisem ser tratados de forma diferente em relação à segurança já que não é possível a utilização de mecanismos de segurança tradicionais neste tipo de tecnologia.

1.1 Problemática

Devido ao fato de que protocolos, padrões e mecanismos de segurança tradicionais já utilizados por computadores e outros dispositivos como criptografias não foram pensados nem construídos para rodar com baixo poder de processamento, memória ou armazenamento, torna-se inviável a utilização dos mecanismos tradicionais em dispositivos IoT. Essa inviabilidade se dá às limitações e baixas capacidades dos simples dispositivos utilizados com essa tecnologia. Muitas vezes, tais dispositivos são incapazes de executar defesas básicas necessárias como um algoritmo de criptografia, pois demanda um poder computacional e de memória maior do que o disponível. Sua memória limitada faz com que não haja a central de gerenciamento de memória, impossibilitando muitas técnicas de endereçamento, entre outras dificuldades como a fragmentação, obstáculos à serem enfrentados ao fazer a utilização desta tecnologia. [11].

A partir de tais considerações que não torna possível a aplicação dos já conhecidos e utilizados padrões, modelos e mecanismos padrões de segurança, em especial as criptografias. Visa-se desenvolver e efetuar a implementação do algoritmo do sistema criptográfico Rivest-Shamir-Adleman (RSA) já utilizado em outros dispositivos com alto poder de processamento e memória e trazê-lo para a Internet das Coisas, visando contribuir com a qualidade de segurança nesta tecnologia. Posteriormente fazendo a análise da implementação feita para Arduino e efetuando a verificação de sua execução. Desta maneira, elimina-se a necessidade de se utilizar ou adaptar modelos tradicionais.

1.2 Objetivos

O presente projeto visa desenvolver uma versão compatível do modelo criptográfico RSA para Arduino utilizando-se de seus algoritmos já existentes. Com isso, têm-se o objetivo de melhorar a segurança de um dispositivo Arduino que faça parte da Internet das Coisas através da utilização de um sistema criptográfico de alto nível e eficácia.

A partir de algoritmos e técnicas já existentes que em alguns casos não podem ser aplicadas em sua totalidade em IoT, será desenvolvida uma versão do sistema criptográfico RSA próprio que se adapte e seja capaz de ser utilizado em microcontroladores AVR Arduino.

Com o algoritmo em funcionamento visa-se efetuar uma análise de desempenho e viabilidade do mesmo nesta tecnologia e quais são suas limitações e dificuldades devido aos obstáculos da plataforma escolhida.

Afim de alcançar o objetivo principal, esse trabalho tem como objetivos específicos as seguintes questões:

- Definir o sistema criptográfico à ser implementado.
- Estudar o sistema criptográfico escolhido.
- Estudar as principais restrições IoT que impedem o uso desses mecanismos de segurança na tecnologia
- Definir a plataforma a qual o algoritmo será desenvolvido e analisado.
- Desenvolver o algoritmo criptográfico através da adaptação de acordo com as limitações e restrições da plataforma selecionada.
- Analisara eficiência e viabilidade do algoritmo.

1.3 Justificativa

A Internet das Coisas está se tornando uma infraestrutura chave para o desenvolvimento de novas possibilidades e o amadurecimento da internet. Contudo, devido ao intenso crescimento de dispositivos IoT possuindo uma segurança pobre, torna-os cada vez mais vulneráveis a ataques cibernéticos [10]. Em razão destes dispositivos não serem aptos a executar um mecanismo de segurança completo já existente em outras plataformas, tais como computadores e smartphones, esses dispositivos se tornam vulneráveis por muito tempo devido à fatores como por exemplo senhas padrões e erros não corrigidos [12]. Em vista disso, acredita-se que esse trabalho o qual possui o objetivo de implementar um modelo criptográfico de segurança para Arduino a partir de seu algoritmo base,

porém incapaz de se utilizar nessa plataforma contribua para o aumento da segurança nessa tecnologia e também um caminho a ser seguido tanto para iniciantes, quanto para os novos serviços digitais que estão por vir.

1.4 Organização do Trabalho

A organização do trabalho obedece a estrutura a seguir. No Capítulo 2 é apresentado uma visão referencial sobre o que é a Internet das Coisas e suas principais características. Em seguida também é apresentado suas dificuldades ao se tratar de segurança. Por último é apresentado as consequências desses problemas resultantes de limitações nessas plataformas e dispositivos da Internet das Coisas. O Capítulo 3 apresenta o método proposto para a solução de problemas de segurança em Iot fazendo a implementação do sistema criptográfico RSA de maneira que se encaixe em suas limitações e também a análise do algoritmo. O Capítulo 4 possui todos os passos da implementação, assim como as dificuldades, os obstáculos encontrados e como eles foram resolvidos para que se atingisse o objetivo do presente projeto. O Capítulo 5 apresenta os resultados obtidos pelos testes realizados do algoritmo implementado na plataforma Arduino, assim como relações e comparações dos resultados. O Capítulo 6 descreve as considerações do trabalho, em relação ao método proposto e aos problemas pendentes, bem como as dificuldades que foram enfrentadas com a implementação e trabalhos futuros a partir deste projeto.

2 FUNDAMENTAÇÃO

2.1 Internet das Coisas

O termo Internet das Coisas, ou Internet of Things (IoT) vem se popularizando cada vez mais, devido à sua recente ascensão identifica-se o termo como algo novo, porém a ideia de conectar objetos entre si é discutida desde 1991 quando a conexão de internet começou a se popularizar. O termo Internet of Things foi oficialmente proposto em 1999 por Kevin Ashton [2].

A Internet das Coisas tem sido interpretado por alguns autores como uma tecnologia de tal forma tão inovadora que o seu impacto na sociedade é equivalente ao de uma Revolução Industrial [6]. Na literatura, certos autores até identificam como a quarta revolução industrial. Diante disso este paradigma tecnológico está sendo uma das principais responsáveis por transformar a internet ainda em seu período de infância em uma internet do futuro a qual é totalmente integrada [3].

A definição exata e precisa de IoT é frequentemente discutida até os dias atuais devido ao fato de que não existe um consenso geral do significado. O termo é tão genérico que pode variar de computação ubíqua, computação pervasiva ou até mesmo cidades inteligentes [4]. O que existe é apenas um acordo geral de que IoT é um acumulado de objetos e conectividade [5] o qual possibilita dispositivos conectarem-se à internet através de seus sensores [6].

Ao navegar pela literatura, um leitor interessado no tema deve experienciar uma dificuldade real para entender o que IoT realmente significa, como quais ideias estão por trás deste conceito. Esta confusão se dá devido à composição sintática de dois termos: “Internet” e “Coisas”. O primeiro impõe uma visão de IoT como orientada a redes, já o segundo tem um foco em objetos genéricos à serem integrados. Quando os dois termos são colocados juntos, seu significado seria “Uma rede universal de objetos interconectados unicamente endereçados, baseados em protocolos de comunicação padrões” [7].

Devido à isso, Elkhodr, Shahrestani e Cheung[4] afirma que a internet das coisas estende a interação entre humanos e aplicações à uma nova dimensão devido a comunicação pelos objetos os quais podem ser humanos ou aplicações.

Os campos de aplicação para tecnologias são tão numerosos quanto diversos, este fato se dá pelo crescimento de soluções usando tecnologias IoT todos os dias [8]. As áreas de aplicação mais prominentes incluem indústrias inteligentes, casas inteligentes, e quando citado os projetos de cidades inteligentes, pode-se falar sobre sistemas de monitoramento em tempo real de parques e iluminação inteligente de ruas sendo exploradas [7].

Existe a proposição de que a tecnologia digital resultou em um novo tipo de arquitetura de produto: A arquitetura em camadas modulares. Podemos considerar como um híbrido de arquiteturas modulares de um produto físico e arquitetura em camadas da tecnologia digital [13].

- **Arquitetura modular de produtos físicos** provê um esquema onde o produto é decomposto em componentes acoplados, os quais recebem funcionalidades e são interconectados através de interfaces específicas [14].
- **Arquitetura em camadas** é incorporada em produtos físicos, gerando funcionalidades através das capacidades baseadas em software [15].

2.1.1 Objetos na Internet das Coisas

A Internet das coisas estende a interação entre humanos e aplicações a um novo nível, pelo fato de trazer conectividade às "coisas" [4] (Objetos), as quais tem um dos papéis mais importantes quando se trata de IoT.

Um objeto em IoT é definido como qualquer máquina, dispositivo, aplicação, computador e objetos físicos ou virtuais que possuem uma comunicação com a internet e podem consumir, passar ou ter acesso à informação digital [4].

As "coisas" tornam-se objetos inteligentes devido à sua conectividade dada pela internet, os quais são capazes de fazerem a comunicação e operação entre si ou até humanos. Como por exemplo em casas, uma aplicação seria uma rede autônoma com sensores físicos, combinados com comunicações *wireless* e tecnologia de micro-sistemas, os quais juntos podem fazer a medição constante de parâmetros ambientes como temperatura, umidade, iluminação, etc [16]

2.1.2 Agregação de valor por meio da Internet das Coisas

Nos útlimos anos, a ascensão da ciência e da tecnologia resultou em diversas evoluções muito importantes como o aumento no gerenciamento de energia, a expansão da comunicação de internet, a evolução das memórias e os avanços em microprocessadores. Em consequência dessa grande evolução ano a ano, tornou-se possível a digitalização de funções e capacidades essenciais em produtos da era-industrial [13]. Com isso a variedade de oportunidades não conhecidas anteriormente geram muito valor à Internet das Coisas.

A Figura 1 contém a lógica do valor de criação que a Internet das Coisas pode agregar. Ao observar a imagem, é possível ver como as soluções IoT combinam objetos físicos com a Tecnologia da Informação (TI) em forma de *Hardware* e *Software*. Essa combinação resulta em um serviço baseado em "coisas", os quais podem ser acessados não apenas localmente, mas de maneira global [8].

Coisa	+	TI (Hardware / Software)	=	Funções baseadas em coisa (Física / Local)	+	Serviço baseado em TI (Digital / Global)
	Lâmpada	IOT	→	Luz		Segurança
	Caixa	IOT	→	Capacidade de Armazenamento		Reposição Automática
	Trator	IOT	→	Rebocar Veículos		Predição de manutenção

Figura 1 – Internet das Coisas: Lógica de Produtos e Serviços

Fonte: Baseado em Wortmann e Flüchter[8]

Como visto na Figura 1, a Internet das Coisas utiliza-se de Coisas (objetos) do mundo real em conjunto com a Tecnologia da Informação para agregar valor. Na demonstração da imagem, o primeiro exemplo é uma lâmpada, a qual tem como função primária e básica fornecer iluminação para um local específico. Se a lâmpada for conectada de qualquer maneira com a Internet das Coisas, a mesma pode adicionalmente detectar a presença humana e servir como um sistema de segurança de baixo custo, o qual em um evento como uma invasão de determinado local, ativaria um modo de cintilamento e enviaria um alerta ao *smartphone* do proprietário.

De maneira similar, a funcionalidade básica de uma caixa de estoque é oferecer capacidade de armazenamento, com a aplicação de IoT, seria possível mensurar seu próprio peso, em razão disso, ao monitorar seu peso, seria capaz de fazer a detecção do estoque, oferecendo um serviço de reposição automática.

No caso do trator, sua função básica é oferecer equipamento para necessidades da fazenda. Desta maneira, conectando o trator à Internet das Coisas, ele poderia ter outras funções como ter facilidade na predição de manutenção e até mesmo a otimização dos serviços na fazenda [17].

Na Figura 1 fica claro como o paradigma tecnológico de Internet das Coisas tem a capacidade de agregar valor em objetos físicos do mundo real, tornando algo com uma simples função, em um aglomerado de funções adicionais.

Segundo Gubbi Jayavardhana e Buyya[18], ao fazer a análise da tecnologia IoT de uma perspectiva de alto nível, a Internet das Coisas depende de 3 elementos principais para que haja a real agregação de valor, sendo eles:

- ***Hardware***: Sensores, elementos atuantes e comunicação incorporado ao *Hardware*.

- ***Middleware***: Armazenagem por demanda e ferramentas computacionais para análise dos dados.
- **Apresentação**: Ferramentas com uma maneira simples para visualização e entendimento que possam ser acessados de diferentes plataformas.

Em sua publicação, Fleisch, Weinberger e Wortmann[17] diz que um objeto do mundo real agrega valor após passar por 5 etapas (camadas), para que então possa se tornar uma aplicação da Internet das Coisas. Um claro exemplo disto está na Figura 2 onde podemos observar as camadas de agregação de valor em uma aplicação IoT, a qual a partir de um único e simples objeto físico do mundo real tem seu valor agregado através da Internet das Coisas.

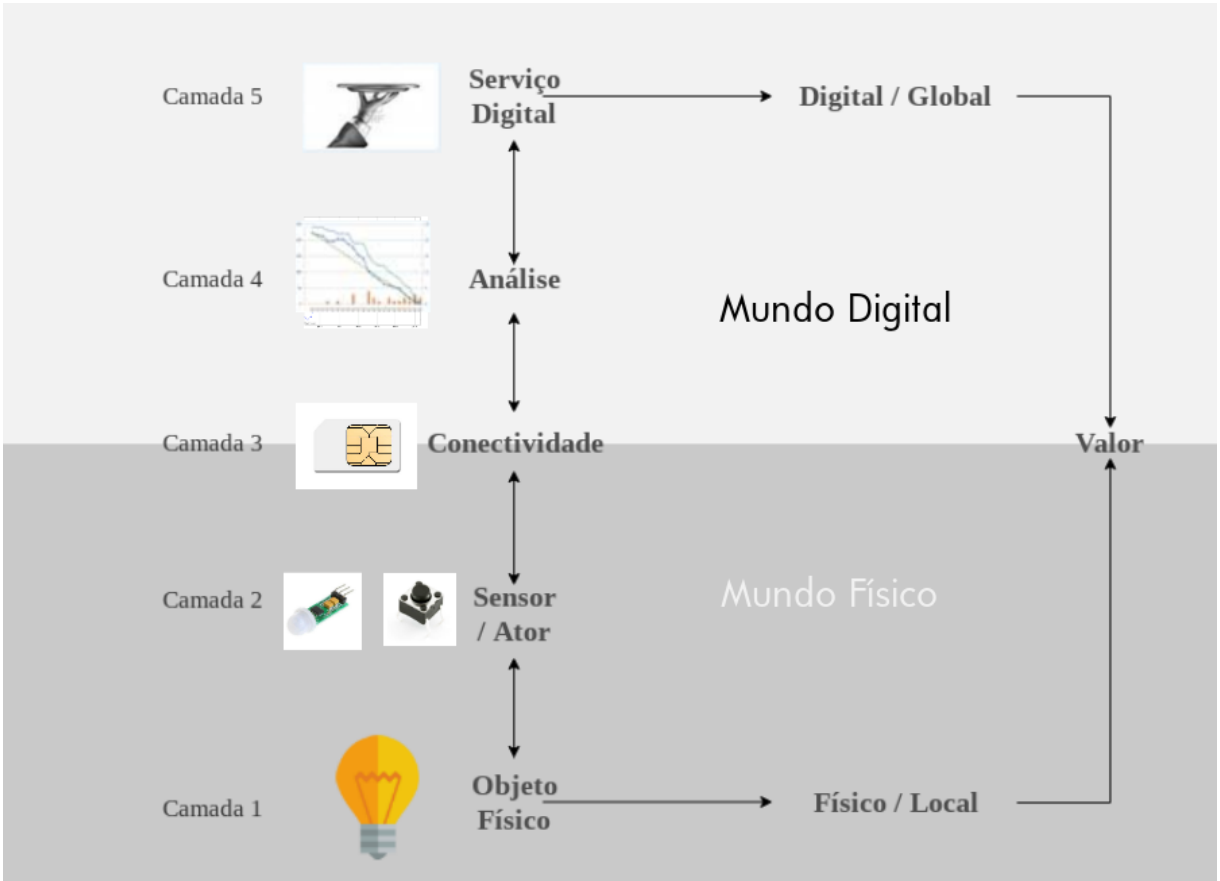

Figura 2 – Camadas de agregação de valor em aplicações IoT

Fonte: Baseado em Fleisch, Weinberger e Wortmann[17]

- **Camada 1 (Objeto Físico)**: O elemento físico, sendo usado uma lâmpada como exemplo em [Fig. 2] forma a primeira camada do modelo de agregação de valor. Este oferece o primeiro benefício de contato direto ao usuário.
- **Camada 2 (Sensor / Ator)**: Na camada 2 o objeto físico é equipado com um mini computador o qual possui elementos atuantes e sensores. O sensor mede os dados

locais, enquanto os elementos atuantes executam serviços locais, assim gerando benefícios locais. No exemplo da Lâmpada de LED observado na Figura 1, o sensor de presença mantém a verificação continuamente se existem pessoas presentes no local. O elemento atuante aciona a luz automaticamente quando a presença humana é detectada e desativa novamente quando não há ninguém, gerando benefícios locais.

- **Camada 3 (Conectividade)**: A camada 3 oferece a possibilidade da camada 2 conectar seus sensores e elementos atuantes à internet, tornando-os globalmente acessíveis. Neste exemplo, a lâmpada pode ser endereçada por um módulo de conexão com a internet e transmitir seus status para pessoas autorizadas ao redor do mundo.

- **Camada 4 (Análise)**: A conectividade por sí só, não agrega nenhum valor em especial. Na camada 4, os dados dos sensores são coletados, armazenados, checados e classificados. Devido à isto, os dados podem ser integrados com outros serviços *Web*. Seguindo o exemplo oferecido, na camada 4 toda vez que há a ativação e desativação da lâmpada é coletada junto com o horário, os quais em seguida são armazenados.

- **Camada 5 (Serviço Digital)**: A camada final usa as saídas providas pela camada anterior e as estrutura em um serviço digital. Este serviço pode ser empacotado em uma forma usável, como um serviço Web ou aplicação móvel. Em razão disso, torna o acesso global. Nesta camada em conjunto com o exemplo, quando o sensor de presença detecta uma pessoa no local, envia direto ao aplicativo no *smartphone*, gerando um alarme ao proprietário, à polícia ou até mesmo ao vizinho.

Embora as 5 camadas sejam diferentes umas das outras, elas não podem ser criadas de formas independentes. Para que se alcance a camada 5, é necessário executar todo o caminho entre a camada 1 e 5 [17].

2.1.3 Desafios da tecnologia

Apesar da Internet das Coisas possuir tantas vantagens, agregar tanto valor à praticamente qualquer objeto e ser uma possível tecnologia tão revolucionária, ainda é uma tecnologia com muitos desafios e dificuldades. No mundo de hoje essa tecnologia ainda apresenta diversos problemas, muitos deles relacionados à segurança e privacidade [7].

Os principais problemas e dificuldades enfrentadas por IoT podem ser listados como padronização, mobilidade, protocolos de transporte, autenticação, integridade de dados e privacidade como poderá ser visto nas seções seguintes.

2.2 Segurança da Internet das Coisas

O termo segurança pode significar uma ampla gama de diferentes conceitos. Acima de tudo, se refere à provisão básica de serviços de segurança, incluindo confidencialidade, autenticação, integridade, autorização, impossibilidade de negação e disponibilidade. [9].

A Internet das Coisas está se tornando uma infraestrutura chave para o desenvolvimento de novas possibilidades. Porém, devido ao intenso crescimento de dispositivos IoT portando uma segurança pobre, torna-os mais vulneráveis a ataques cibernéticos [10].

Um dos principais problemas e preocupações com a Internet das Coisas é a segurança devido ao fato de que essa questão em dispositivos IoT se tornam diferentes do tradicional [9], pois estes dispositivos não são aptos a executar um mecanismo de segurança completo devido ao consumo limitado de recursos. Como resultado, esses dispositivos se tornam vulneráveis por muito tempo devido à fatores como por exemplo senhas padrões e erros não corrigidos [12]. A impossibilidade de executar os mecanismos padrões tornam a segurança da Internet das Coisas um problema, entre elas, as limitações mais comuns são [9]:

- Dispositivos IoT tem uma baixa capacidade de processamento e a maior parte funcionam através de baterias.
- Dispositivos IoT normalmente possuem a memória altamente limitada quando comparados à celulares e computadores. Os esquemas de segurança convencionais não são modelados para dispositivos com limitação de memória
- Dispositivos IoT frequentemente usam uma baixa frequência de comunicação de dados por rádio. Os esquemas tradicionais de segurança não podem ser aplicados para sistemas baseados em IoT devido à baixa largura de banda de comunicação.
- A instalação de *patches* de segurança em dispositivos IoT são impraticáveis devido ao fator de que os leves sistemas operacionais devem ter módulos para integrar novos códigos ou bibliotecas.
- Dispositivos IoT móveis podem entrar em uma rede sem qualquer tipo de configuração específica
- Devido à heterogeneidade dos componentes torna-se difícil encontrar uma solução de segurança que acomode toda a variedade de diversos dispositivos.

Segundo Zhou, Zhang e Liu[11], os principais problemas de segurança e privacidade em IoT não se deve somente à toda restrição existente. A limitação em IoT é apenas uma das principais causas no quesito segurança. Zhou, Zhang e Liu[11] acredita que os

problemas quando se trata de privacidade e segurança se dá por 8 características ilustradas na Figura 3.

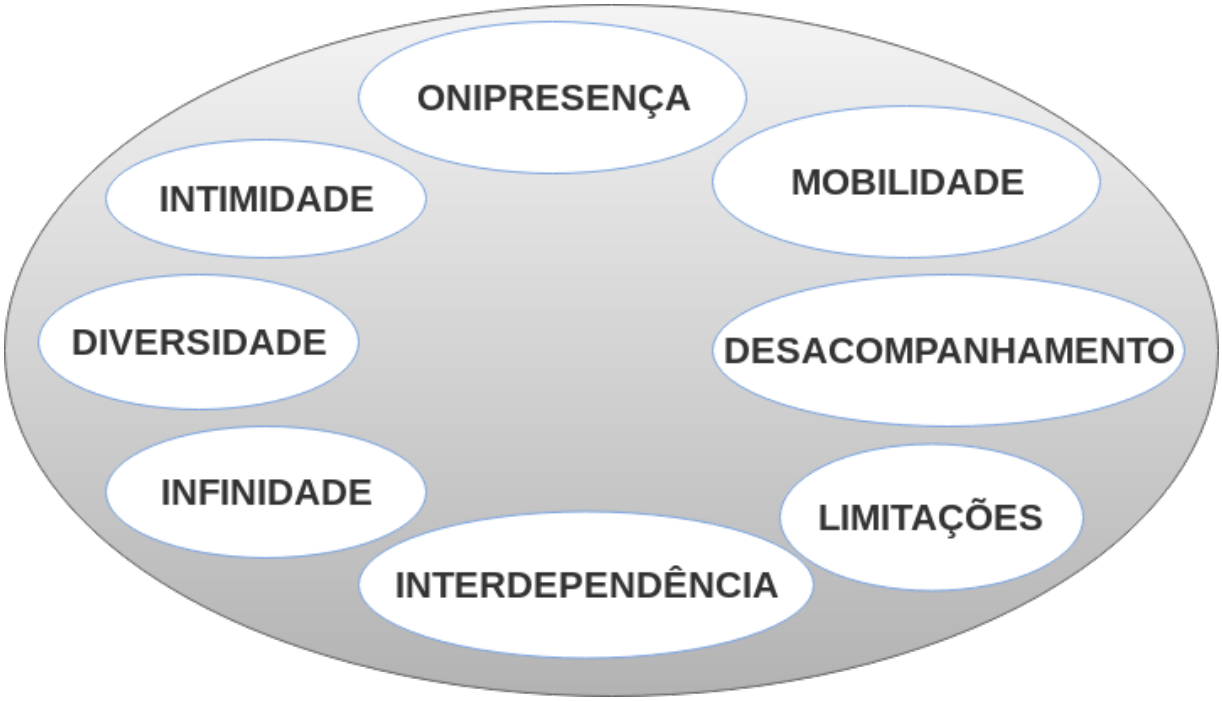

Figura 3 – Características responsáveis por problemas de segurança.

Fonte: Baseado em Zhou, Zhang e Liu[11]

- **Interdependência:** Conforme a evolução de dispositivos IoT, as interações entre dispositivos se tornaram mais complexas e a necessidade humana para o controle se tornou desnecessária. Esses dispositivos não precisam mais se comunicar explicitamente com algum humano para efetuar uma tarefa, muitos deles são controlados por outros dispositivos os quais são manuseados usando regras inteligentes, muitas vezes acessados pela nuvem através da Internet como a plataforma IFTTT [19]. Um exemplo de interdependência pode ser facilmente citado como um termômetro que ao detectar que a temperatura limite foi excedida, verifica se a tomada inteligente do ar condicionado está no modo "off" e então abre as janelas. Essa relação entre objetos é chamada interdependência. O problema com esta característica se dá pelo fato de que no exemplo, um invasor não precisaria de acesso ao termômetro ou ao ar-condicionado, nem as janelas. Bastaria ele controlar a tomada inteligente a qual está ligada à rede de Internet, apenas com isso, poderia simplesmente desligar o ar-condicionado, fazendo com que a temperatura se elevasse e as janelas abrissem automaticamente, gerando uma brecha para invasão física. Essa característica torna-se uma barreira para a segurança em dispositivos IoT pelo fato de que há muita dificuldade em proteger os objetos como um todo de forma independente, resultado de uma dependência entre eles.

- **Diversidade:** Para que se tenha a capacidade de funcionar em diferentes cenários, dispositivos IoT são feitos para operar em tarefas específicas e interagir com muitos ambientes físicos. Um exemplo disto é um sensor de umidade simples e pequeno, o qual com uma pequena memória dá conta de sua função enquanto uma máquina de uma indústria possivelmente tem mais performance do que um smartphone. Devido à isso, há uma grande dificuldade em se estabelecer protocolos padrões e sem falhas que englobem toda essa diversidade.

- **Limitações:** Devido à limitação física e de preço, muitos dispositivos IoT em indústrias são criados para operarem com um sistema leve. Esses dispositivos também possuem menos poder computacional e armazenamento que computadores tradicionais ou *smartphones*. A maior parte desses dispositivos são incapazes de executar defesas necessárias para a rede e sistemas. Por exemplo, dispositivo IoT simples, não possui a Unidade de Gerenciamento de Memória, dessa forma, várias técnicas de endereçamento de memória não podem ser aplicadas. Algorítmos de criptografia também enfrentam problemas, pois uma autenticação utilizando chave pública demanda muito recurso computacional. Em razão disso, os dispositivos se tornam alvos fáceis de ataques através de vulnerabilidades de memória para comprometer todo o resto.

- **Infinidade:** Em razão da rápida proliferação de dispositivos IoT, a quantidade de dados gerados, transmitidos e usados por esses dispositivos estão aumentando de maneira astronômica. Em 2016 houve um ataque de tráfego o qual era composto por mais de 1 milhão de dispositivos IoT. Dessa maneira, muitos dispositivos inseguros podem ser usados para orquestrar-se um ataque DDoS em larga escala [20].

- **Desacompanhamento:** Diversos módulos e dispositivos tais como sensores e medidores são feitos para operarem por um longo período de tempo sem acesso físico. Com o crescimento das redes wireless, estes simples sensores e módulos estão evoluindo para dispositivos IoT. Esse tipo de dispositivo normalmente possui uma dificuldade de se conectar fisicamente uma interface externa para a verificação de seu estado. Em razão disso, os ataques remotos são difíceis de detectar.

- **Intimidade:** Nos últimos anos, cada vez mais tem surgido dispositivos usando medidores, sensores, dispositivos que se pode vestir como por exemplo relógios digitais. Esses dispositivos não apenas coletam nossos dados biológicos como batimento cardíaco e pressão sanguínea, como também monitoram e gravam informações íntimas e atividades diárias como a temperatura de sua casa ou os locais que você esteve. Alguns pesquisadores [21] comprovaram que atacantes podem concluir se há alguém em uma casa apenas analisando os dados dos sensores de fumaça e dióxido de carbono.

- **Mobilidade:** Muitos dispositivos IoT, tais como vesíveis e carros inteligentes normalmente trocam de uma rede para outra de forma constante e se comunicam com novos dispositivos. Devido à isso, atacantes tendem a injetar códigos maliciosos em dispositivos IoT móveis, pois eles fazem a troca de redes com muita frequência, espalhando essa ameaça rapidamente. Como consequência, quando um dispositivo IoT móvel "salta" de uma rede à outro, ele carrega todos os dados para essa nova rede.

- **Onipresença:** Os dispositivos IoT tem cada vez mais se encaixado e entrado na vida das pessoas. Eles se tornarão uma parte indispensável do dia a dia das pessoas, estando por todos os lados no futuro. Em razão disso, Zhou, Zhang e Liu[11] acredita que no futuro, o humano se tornará o principal problema de segurança da Internet das Coisas. Isso se dá em razão devido à falta de preocupação com segurança dos criadores de dispositivos IoT, que produzem dispositivos inseguros, possuindo muitas vulnerabilidades e falhas de projeto. De acordo com as estatísticas de Govtech[22], até o ano de 2020 todos terão de 6 a 8 dispositivos IoT. Conforme esses dispositivos vão se tornando mais frequentes e próximos da humanidade, eles se tornarão mais inteligentes, eles serão aptos a completar muitas tarefas sem nenhuma intervenção humana. O resultado disso é a dificuldade de saber se um dispositivo tão próximo de você e usado muito na sua vida está comprometido até que o ataque cause sérias consequências.

2.2.1 Requisitos de segurança em sistemas IoT

Sobre segurança em sistemas e dispositivos IoT, Hossain, Hasan e Skjellum[10] afirma que tratando-se de tecnologias envolvendo a Internet das Coisas, existem diversas propriedades que devem ser consideradas ao elaborar uma solução para esse universo de conectividade. Essas propriedades fundamentais a se considerar ao desenvolver um sistema de segurança IoT podem ser categorizadas como: Requisitos de segurança da informação, Requisitos de segurança de nível de acesso e Requisitos de segurança funcionais. Esses requisitos são listados e explicados a seguir de acordo com a definição de Hossain, Hasan e Skjellum[10].

Requisitos de Segurança da informação

- **Integridade:** A integridade de um dispositivo IoT pode ser facilmente comprometida modificando os dados armazenados ou em transferência. A integridade permite que você saiba que todos os dados recebidos não possuíram alteração.

- **Proteção da Informação:** A privacidade e confidencialidade de dados tanto online quanto armazenados devem ser protegidos. A proteção se dá pela limitação de acesso de informação e divulgação para terceiros.

- **Anonimato:** Essa propriedade esconde a fonte dos dados e ajuda com confidencialidade e privacidade. Como exemplo pode ser usado um carro inteligente, que não deve compartilhar dados próprios enquanto passa informações sobre a rodovia.

- **Impossibilidade de negar:** A impossibilidade de negar é a certeza que algo não pode ser desfeito. Por exemplo, um dispositivo IoT não pode voltar atrás e negar uma mensagem que já foi enviada.

- **Frescor:** Dispositivos IoT em geral, normalmente lidam com dados em tempo real, devido a isso, a necessidade de um frescor nas mensagens é uma propriedade muito importante para sistemas IoT pois garante que o dado é recente.

Requisitos de nível de acesso

- **Autenticação:** Essa propriedade permite que cada nó conheça sua identidade na Internet das coisas. Por exemplo, um receptor que recebeu alguns dados, consegue ter certeza que os dados vieram da fonte correta e confiável. A autenticação também garante que usuários ganhem acessos especiais à dispositivos Iot e às redes.

- **Controle de acesso:** Ao levar em conta essa propriedade, há a garantia de proteger redes IoT, dispositivos, serviços e recursos. Têm-se a possibilidade de um usuário autenticado ser incapaz de acessar certos serviços ou recursos.

Requisitos funcionais de segurança

- **Interoperabilidade:** O uso de soluções de segurança em IoT não devem impactar na operação do amontoado de objetos e comunicação de um sistema IoT.

- **Escalabilidade:** Um enorme número de dispositivos IoT estão conectados através das redes de dados, e a cada dia mais e mais dispositivos se conectam. Portanto um sistema de segurança proposto deve prover escalabilidade suficiente.

- **Eficiência de Memória:** Devido ao fato de possuírem armazenamento e memória limitada, deve-se levar em consideração essa propriedade. Os algoritmos de segurança devem ser otimizados e consumir o mínimo de espaço em RAM (*Random Access Memory*) durante a execução e não consumir muito armazenamento para armazenar artefatos de criptografia.

- **Computação Mínima:** Ao modelar um sistema de segurança em IoT, deve-se pensar no menor uso de computação, comunicação e processamento. Ao desenvolver esse sistema de segurança, deve-se trocar a menor quantidade de mensagens possível e não consumir muito da CPU.

- **Resiliência:** Sistemas de segurança em IoT, devem evitar pontos únicos de falha, pois uma única falha em um único objeto ou componente, pode comprometer todo o sistema.

2.2.2 Ataques na Internet das Coisas

A integração de dispositivos inteligentes ("coisas") alcança muitos domínios em IoT como cuidado com a saúde, casas inteligentes, cidades inteligentes, transporte inteligente. De qualquer maneira, a interconexão entre as "coisas", resulta em problemas de segurança difíceis de prever [7].

Devido à falta de padrões de segurança em IoT, a vulnerabilidade em dispositivos mal configurados se torna um grande problema [23]. Existem diversos tipos de ataques em IoT, tais como *DoS (Denial of Service)*, *Spoofing*, *Sybil* e *Node Capture*. Abaixo será analisado uma comparação entre os diferentes tipos de ataques, seus alvos, as fraquezas e explicação do ataque.

- **Falsificação (*Spoofing*):** É uma técnica maliciosa utilizada por hackers para depreciar sistemas, indivíduos e organizações [24]. Os ataques são criados ao gerar uma mensagem falsa de erro criando uma rota para aplicação de muitas outras técnicas [25]. No começo, o falsificador não transmite um sinal, apenas ouve ao transmissor apropriado. Quando o transmissor legítimo para de transmitir, o falsificador passa a mandar um sinal não confiável [26]. Um exemplo de falsificação é quando o atacante polui toda a rede enviando rotas falsas de informação, como por exemplo no Facebook, um usuário quer curtir algo, e acessa um link que o redireciona para autenticação, porém é uma página falsa, ao inserir seus dados, terá suas informações pessoais, usuário e senha roubados. [23].

- **Sybil:** Com a evolução da Internet das Coisas, os dispositivos ficaram expostos à ataques conhecidos como *Sybil*, o qual faz com que um único nó na rede, possua várias identidades. Isso significa que o atacante pode estar em mais de um lugar ao mesmo tempo, resultando na degradação da integridade dos dados e recursos. Esse tipo de ataque é usado para roubar informação através da propagação de *malware* em um *website*.

- **Negação de Serviço (DoS):** Um ataque de Negação de Serviço pode fazer com que a rede tenha sua capacidade de tráfego reduzida. Existem 2 tipos de ataques de negação, sendo eles: Ataque de negação de serviço distribuído e Ataque de negação de serviço comum [27] [28]. Em um ataque comum, é necessária uma ferramente que envie pacotes à uma rede à ponto do sistema parar, ou até ser reiniciado. Enquanto o distribuído pode fazer um único ataque utilizando muitos dispositivos.

Os ataques listados e explicados anteriormente, consistem em ataques direto à rede ou pacotes trafegados, porém, todos os dispositivos IoT consistem em sensores que monitoram a variação dos parâmetros. Informação que pode ser facilmente modificada por atacantes. Em sua publicação, [23] defende que também existem ataques com intenção de prejudicar a informação e os dados de um dispositivo, desta forma, os ataques baseados em dano de informação podem ser divididos em 5 categorias.

- **Interrupção**: A principal intenção em um ataque de interrupção é a disponibilidade do sistema. Dessa maneira, gerando uma exaustão de recursos, resultando em até mesmo no desligamento do dispositivo.
- **Escuta (*Eavesdropping*)**: Um adversário bloqueia o receptor do dispositivo pegando os pacotes transmitidos quando escuta o canal de comunicação. Dispositivos RFID tem tendências em serem atacados através de escutas. Esse tipo de ataque permite que um usuário malicioso roube dados confidenciais, senhas ou qualquer tipo de informação privada [29].
- **Alteração**: Esse tipo de ataque resulta nas informações do dispositivo sendo alteradas ou modificadas por atacantes, resultando numa ameaça dos requisitos de segurança de integridade.
- **Repetição de mensagem**: Permite a interceptação, reenvia a mensagem original e a modifica para comprometer os dispositivos IoT alvos. Os atacantes mantêm a atual sessão para enviá-la mais tarde. Desse modo, a mensagem reenviada causará confusão no dispositivo IoT, resultando em um perigoso problema para o sistema caso não seja capaz de se recuperar.
- ***Man-in-the-middle***: Atacantes secretamente invadem e alteram a comunicação entre dois dispositivos os quais acreditam estarem se comunicando diretamente. Segundo Bhushan, Sahoo e Rai[30], nos dias de hoje, o tipo de ataque *man-in-the-middle* é o mais bem-sucedido o qual tem o objetivo de ganhar o controle sobre os dados transferidos em uma rede. Onde é utilizado ma escuta, porém os dados de rede não são apenas interceptados, eles também pode ser re-roteados [29].

2.3 Arduino

Arduino é uma plataforma *open-source* a qual provê uma forte base para o mundo do software e hardware. O projeto Arduino começou em 2005, desenvolvido pela pesquisa de alguns estudantes do *Interaction Design Institute*. A ideia inicial de desenvolvimento do Arduino era prover uma plataforma para pessoas que gostam de fazer as próprias coisas como hobby [31]. Placas Arduino são placas de desenvolvimento para criar uma interface entre diversos módulos e periféricos, sensores e comunicação sem fio, desta maneira,

provê uma plataforma IoT baseada em produtos e projetos. Possui um microcontrolador ATMEL o qual utiliza uma arquitetura AVR de 8 bits, acompanhada de três memórias, normalmente nos modelos mais básicos como o Arduino UNO, as memórias são 32KB de memória Flash, 2KB de SRAM e 1KB de EEPROM, acompanhadas de uma velocidade de *clock* de 16MHz.

Figura 4 – Placa Arduino UNO R3

Fonte: Retirado de: Arduino[32].

Essa plataforma pode ter muitos benefícios e características próprias que tornam sua importância muito grande, características essas, que tornam a plataforma Arduino única, diferenciando-se de seus concorrentes e até mesmo placas de outros segmentos. [33][34]:

- **Independência de Plataforma:** As placas Arduino possuem compatibilidade entre plataformas, tais como Windows, Linux e até mesmo o sistema operacional MAC OS. Devido à este fator, possui uma independência de plataforma muito maior que as outras placas dessa área.
- **Efetividade de Custo:** A tecnologia Arduino é barata se comparada à outros microcontroladores. A placa básica, a qual é boa para todos os tipos de projetos básicos para estudantes e entusiastas, custa menos de 50 dólares, sendo a razão de uma aceitação tão grande pelo mundo.
- **Fácil de programar:** A tecnologia Arduino oferece uma maneira simples e fácil para programar através de um ambiente integrado (Arduino IDE), o qual possui fácil compreensão, flexibilidade, robustez para todos os tipos de usuários, exemplos de código e diversas outras facilidades.

- ***Open Source Hardware:*** Pertence à categoria de *hardwares open source*, desta maneira, a tecnologia Arduino pode ser utilizada por vários fabricantes de maneira que possam fazer suas próprias placas personalizadas.

- ***Open Source Software:*** Arduino funciona com base em C e C++, desta maneira, várias bibliotecas podem ser levadas para a arquitetura AVR e serem diretamente usadas em soluções e programas Arduino.

2.4 Criptografia

Existem diversos sistemas e aplicações para proteger a transmissão de dados e aumentar a segurança, a criptografia é uma delas, podendo ser definida como a conversão de um dado em um código cifrado que pode ser decifrado e também pode ser seguramente enviado através de uma rede pública ou privada [35]. A criptografia funciona em dois passos para alcançar a segurança: codificação e decodificação. Uma chave secreta é usada para encriptar e decriptar o texto secreto. Antes de enviar o dado ao destino, é gerado com o uso da chave uma cifra indecifrável baseado no texto original, e apenas a pessoa com a chave pode decifrá-la. Desta maneira qualquer pessoa não autorizada é incapaz de saber a informação original [36].

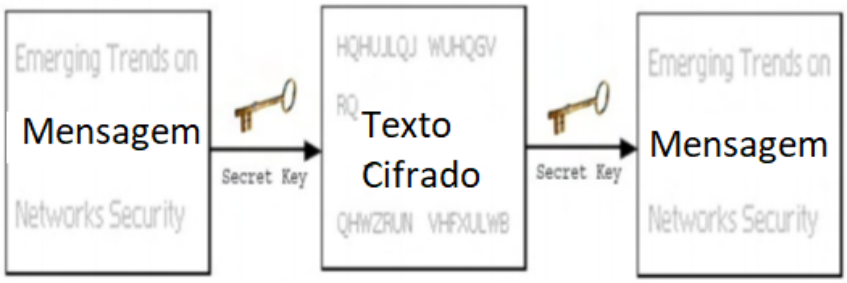

Figura 5 – Etapas base de uma criptografia.

Fonte: Baseada em: Joshi e Joshi[35].

Como observado na Figura 5, a criptografia funciona com base em dois passos simples: Codificação e Decodificação. A mensagem original passa por uma chave secreta, resultando em uma mensagem codificada. Quando esta mesma chave é utilizada novamente, a mensagem codificada volta a ser a informação original. O nível de segurança da mensagem e da informação depende do algorítmo de codificação e decodificação e também da força e tamanho da chave secreta utilizada [37].

Os algorítmos criptográficos podem ter vários tipos de categorias, porém os mais utilizados são: Criptografia de Chave Simétrica e Criptografia de Chave Assimétrica. Na criptografia de chave simétrica é utilizada uma única chave secreta compartilhada para codificar ou decodificar a informação, como observado na Figura 6 Mandal et al.[36]

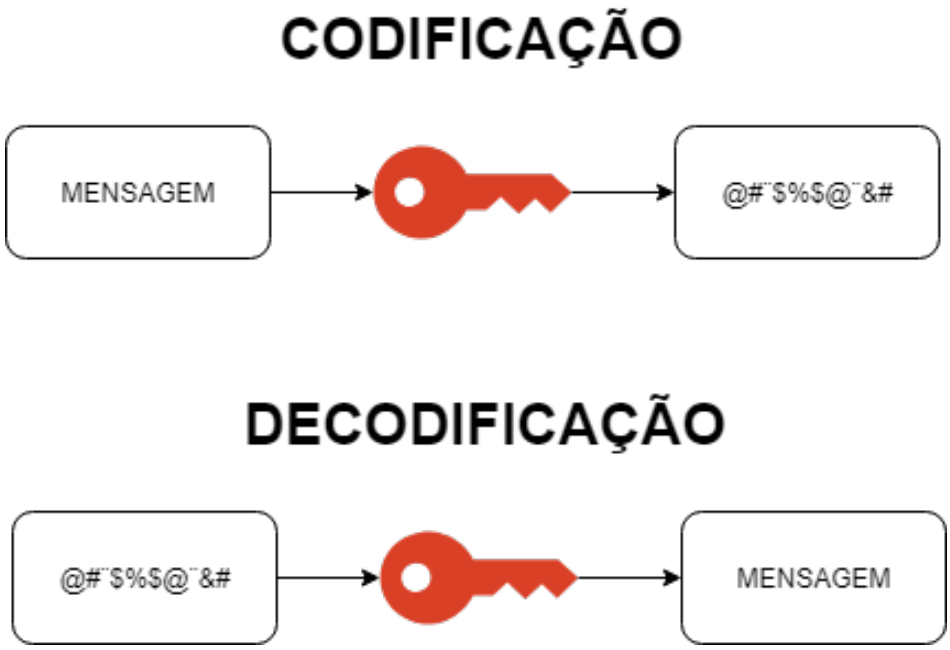

Figura 6 – Etapas de codificação e decodificação em Criptografia de chave simétrica.

Fonte: Baseada em: Mandal et al.[36].

Na Criptografia de Chave Assimétrica, existe uma chave pública (conhecida por todos) usada para a codificação, e uma chave privada (conhecida apenas pelo receptor), a qual é usada para decodificação [35].

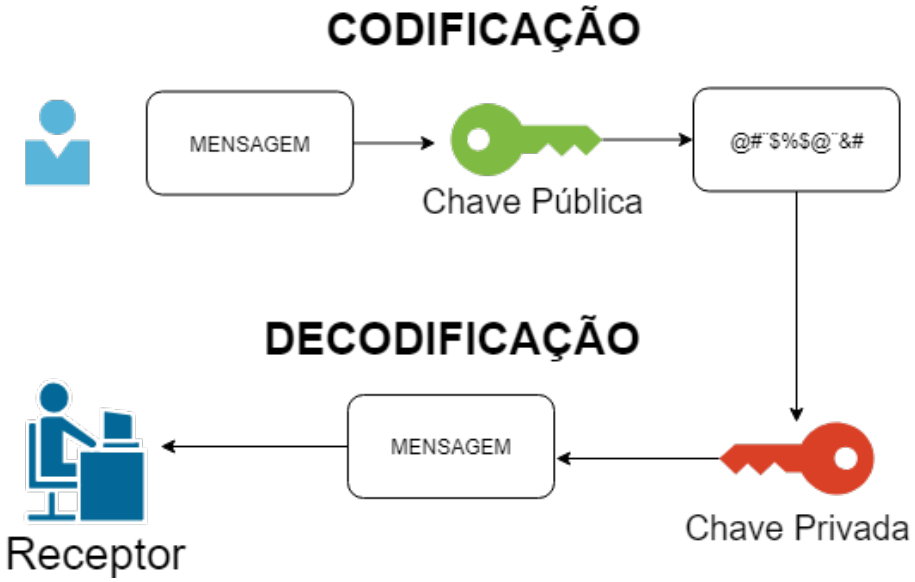

Figura 7 – Etapas de codificação e decodificação em Criptografia de chave assimétrica.

A Figura 7 representa um algorítmo de criptografia de chave assimétrica. Onde a chave pública é compartilhada entre os dois pontos, sendo essa, a chave responsável pela codificação, enquanto apenas o receptor pode conhecer a chave privada, chave essa

responsável pela decodificação feita pelo chave pública.

2.4.1 RSA

O sistema criptográfico Rivest-Shamir-Adleman (RSA) é um dos sistemas de criptografia de chave pública mais amplamente usado mundialmente [38]. Este fato ocorre devido ao fato de que é o algoritmo de chave pública recomendado pelos padrões ISO para sistemas criptográficos [39]. A principal diferença e operação do RSA é computar a exponenciação modular. Sendo baseado em aritmética utilizando grandes números em módulo, muitas vezes se tornando lento em ambientes restritos. O algoritmo RSA sempre utiliza e computa números grandes, isso faz com que a geração do par de chaves leve um tempo maior. A geração do algoritmo funciona basicamente em três etapas: Algoritmo de geração de números primos, teste de primalidade e geração da chave privada. Especialmente quando o RSA decodifica o texto cifrado e gera as assinaturas, nessas etapas, são necessário mais poder de processamento, capacidade computacional e tempo [40]. O algoritmo RSA funciona com base em passos para a geração do par de chaves, para a codificação e também a decodificação. Passos esses, responsáveis por computar os valores necessários para a geração das chaves, assim como executar os métodos de codificação. O RSA funciona com base em um par de chaves pública e privada, tendo tais valores sido resultantes de cálculos para encontrar o módulo n, o expoente público e e o expoente privado d, para que, desta maneira, possa efetuar os algoritmos de codificação e decodificação.

2.4.2 Curvas Elípticas

O princípio básico do algoritmo de Curvas Elípticas é transferir o problema discreto de logaritmo que existe há centenas de anos para as curvas elípticas para que sua implementação encontre o resultado. Com isso, duas principais dificuldades matemáticas são encontradas ao envolver criptografia de chave pública, sendo eles: Decomposição fatorial e Logaritmo discreto [41].

O algoritmo criptográfico de curvas elípticas comparado ao algoritmo do RSA, faz com que a curva elíptica necessite de uma chave de menor tamanho. Porém, as curvas possuem um processamento melhor e mais rápido e necessitam de menos memória, mesmo assim, isto é aplicado apenas na pequena área de implementação de hardware, a qual necessita de uma rápida computação em tempo real. Desta maneira, o RSA possui um nível de segurança e confiabilidade mais elevado e com mais aplicações [42].

O sistema criptográfico de curvas elípticas, transfere o algoritmo original de codificação para as curvas elípticas. Não apenas realiza o protocolo de troca de chaves e as operações de codificação e decodificação através da chave pública, porém também faz a assinatura digital. Sua base é transformar os algoritmos padrões no algoritmo de curva elíptica para sua implementação. Por exemplo, no RSA, é utilizado adição modular co-

mum, após isto ser transferida para curvas elípticas, se tornam ponto de adição e ponto de multiplicação, também chamados por adição elíptica e multiplicação elíptica [41].

2.4.3 AES

O Advanced encryption standard (AES) é uma criptografia simétrica e possui três versões da cifra de Rijndael, sendo elas AES-128, AES-192 e AES-256. É usada para aplicações de camada para prover soluções. A operação de codificação consiste em uma matriz 4x4 a qual possui blocos de 128 à 256 bits. Desta forma, é uma cifra de blocos, ou seja, opera em blocos de tamanho fixo (128 bits, ou 16 bytes) [43].

Em outras palavras, é um algoritmo cuja função direta (cifragem) recebe como entradas um bloco de 128 bits (a mensagem) e uma chave do tamanho escolhido, e devolve uma saída também de 128 bits (a cifra). A função inversa (decifragem) recebe como entrada um bloco de 128 bits (a cifra) e devolve como saída um bloco de 128 bits. Se a chave for a chave correta, essa saída será idêntica à mensagem original [43].

Em sua publicação, através do estudo e comparação entre os algoritmos RSA e AES, Hasib e Haque[44] afirma que o AES é mais rápido e possui uma melhor segurança se comparado com outros algoritmos de chave simétrica. Porém, é de grande dificuldade escrever um algoritmo AES de alta performance para propósitos de evitar ataques. Ele também afirma que o RSA é considerado mais seguro com chaves grandes. Com os resultados de sua pesquisa, chega-se à conclusão que o RSA possui mais aplicações e funcionalidades e o AES é muito mais rápido.

3 PROJETO

No presente capítulo, serão detalhados os passos tomados afim de se desenvolver o sistema criptogáfrico RSA para a plataforma Arduino.

3.1 Visão Geral

Apontados os problemas relacionados à segurança na Internet das Coisas, sendo eles de muitos tipos, mas com o principal desafio relacionado à aplicar os mecanismos e modelos de segurança já existentes, o presente projeto visa implementar o sistema criptográfico de segurança RSA específico para Arduino baseado nos modelos e padrões de segurança já conhecidos.

O projeto possui passos necessários bem definidos para que se alcance o objetivo e porte o sistema para a internet das coisas, desta maneira sendo um trabalho de natureza original. Sendo feito através de uma pesquisa exploratória. Utilizando de uma abordagem qualitativa, visa obter um resultado pelo meio de um estudo sobre IoT, suas restrições e modelos tradicionais de segurança. Devido ao fato de ser uma pesquisa bastante teórica, envolvendo múltiplas partes e fatores não só resultantes e consequentes, é utilizado o procedimento de pesquisa bibliográfica buscando os modelos já existentes visando a apresentação de algo presumivelmente melhor que os já disponíveis e que se encaixa na realidade da IoT.

Em razão do alto nível de conhecimento teórico e materiais, utiliza-se de passos bem definidos, estruturados para que se obtenha o melhor resultado possível através do estudo de múltiplos modelos e métodos de segurança em relação às restrições IoT e maneiras de como quebrar essa barreira. Os passos utilizados podem ser acompanhados na ilustração das etapas representadas na Figura 8.

A figura 8 representa uma visão geral do projeto, a qual está estruturada em etapas necessárias para que se obtenha os resultados. Cada etapa tem grande importância no projeto, sendo necessário que se cumpra todas as fases para que se obtenha uma pesquisa eficaz. As etapas ilustradas na Figura 8 são explicadas detalhadamente a seguir.

3.2 Etapas

As etapas do projeto foram divididas em forma de passos responsáveis por cada parte do projeto. Sendo essas partes complementadas por seu nível à direita e continuada em um nível inferior.

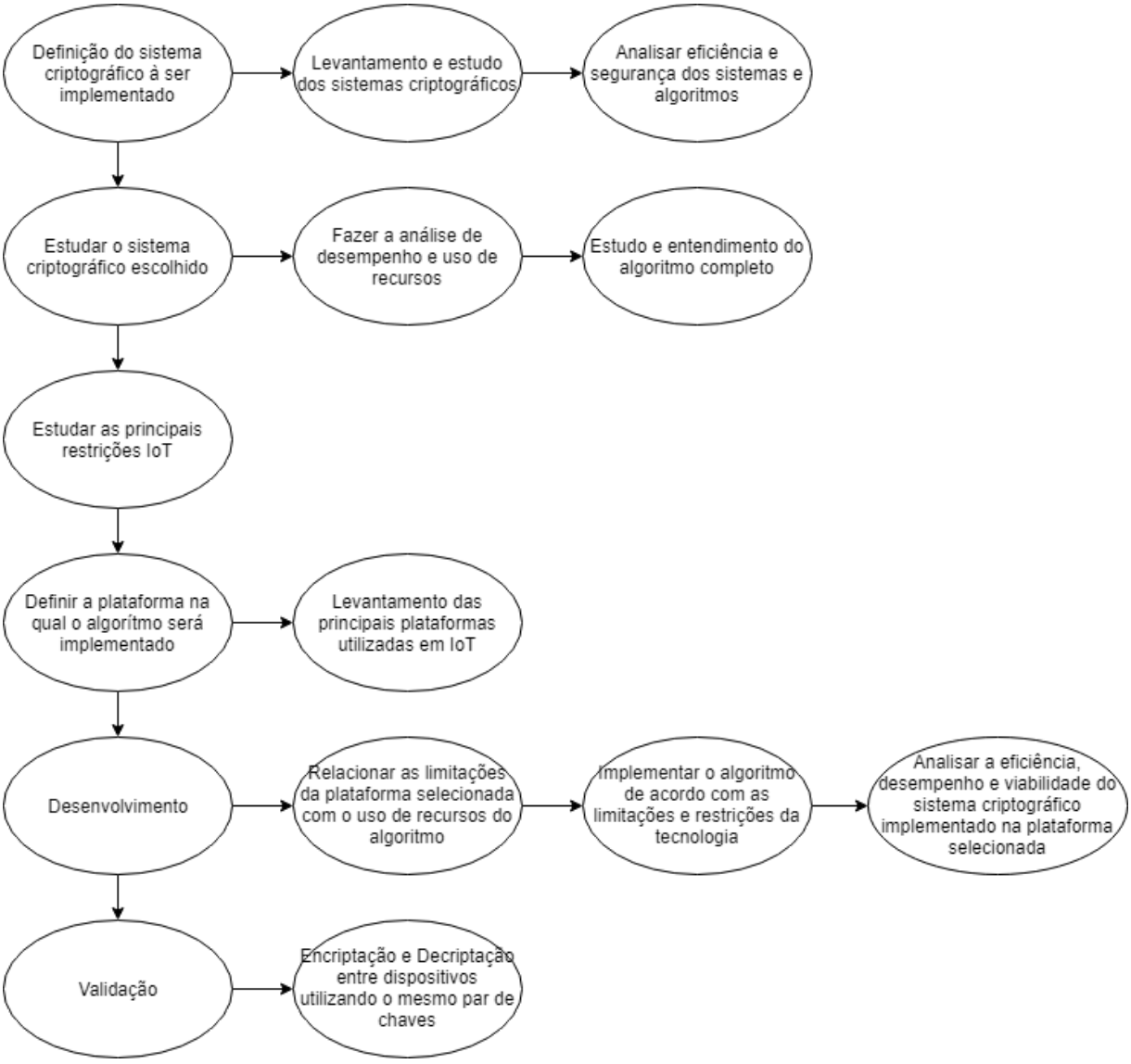

Figura 8 – Visão Geral do Projeto.

A primeira parte do projeto, composta de etapas de levantamento, análise e estudo, foi feita através do material obtido sobre segurança da informação, criptografia e redes de computadores.

A primeira parte da pesquisa, a qual foi definida como ”Definição do sistema criptográfico à ser implementado” foi focada em elencar os principais sistemas criptográficos utilizados para a segurança nos dias atuais. Nessa etapa, houve uma pesquisa teórica em artigos, livros e publicações sobre quais são os principais sistemas criptográficos, seu desempenho e qual seu nível de segurança, critérios estes, que foram utilizados para a definição do sistema à ser implementado, portado e analisado para a Internet das Coisas. O sistema escolhido foi o RSA, com base no artigo publicado por Khan et al.[45], o qual utiliza de diversos sistemas criptográficos na plataforma RaspBerry para fazer a análise de dados.

Ao se definir qual sistema criptográfico seria utilizado como base nessa pesquisa, sendo ele o RSA. A próxima etapa a ser definida foi ”Estudar o sistema criptográfico escolhido”, onde houve um grande levantamento de dados e publicações para que se fizesse um estudo detalhado sobre o algoritmo e o sistema em si. Neste passo, usou-se como base critérios tais como tempo de execução, uso de recursos pelo algoritmo, assim como essa etapa foi responsável por ter um entendimento mais profundo sobre o funcionamento e implementação do sistema.

Na próxima etapa, definida como ”Estudar as principais restrições IoT”, foram elencados e listados as principais dificuldades e obstáculos quando se fala em IoT, obstáculos e restrições os quais tornam a segurança um dos maiores problemas quando se fala desta tecnologia, resultando em problemas de segurança os quais fazem com que se tenha um nível pobre deste requisito na Internet das Coisas. Após a realização deste passo, foi montado de forma estruturada quais são os principais problemas de segurança, sendo organizados em relação à seu nível de risco e danos resultantes. Quando trata-se de segurança, a criptografia e sistemas criptográficos são um dos maiores problemas, devido à seu alto consumo de recursos computacionais, tais como memória e processamento. A etapa de elencar as principais restrições em IoT foi de muita importância, isso se dá pelo fato de que as limitações de IoT foram elencadas, catalogadas e descritas de maneira que se possa saber o porque dessa limitação. Sendo uma etapa que necessitou de um maior estudo e análise das informações para entender quais são as barreiras para a segurança essencial ser aplicada nessa área.

Com o levantamento sobre os sistemas criptográficos, a definição de qual seria utilizado, chega-se à etapa de ”Definir a plataforma na qual o algoritmo será implementado”, etapa essa responsável por fazer um levantamento e estudo das plataformas utilizadas na Internet das Coisas, em resultado da pesquisa, chegou-se à duas principais plataformas, sendo elas: Arduino e Raspberry. As duas são ótimas plataformas de prototipagem e também de desenvolvimento, por ser de baixo custo e capaz de ser flexível devido à capacidade de recompilação do código sendo executado nas mesmas. Para se definir qual seria a utilizada no projeto em questão, foram utilizados alguns critérios como acessibilidade, facilidade, tamanho e possibilidade de geração de novas soluções. Levando em consideração esses critérios, a plataforma definida para utilização foi a Arduino, pois tem questões como baixo custo, possui uma alta capacidade de gerar novas soluções e também um tamanho mínimo.

Após a conclusão da primeira parte do projeto, sendo ela todo levantamento e estudo sobre sistemas criptográficos para definição de qual utilizar e também sua ligação com os problemas e restrições em IoT, assim como qual a plataforma seria utilizada, obtém-se toda a base para que o projeto seja concluído, sabendo que se implementará o sistema criptográfico RSA na plataforma de prototipagem Arduino. A segunda parte

do projeto dá-se pela implementação, análise e validação do algorítmo criptográfico na plataforma Arduino.

O desenvolvimento é a etapa responsável para se chegar ao resultado do projeto, o qual é aplicar a criptografia RSA em IoT. Para que isso seja possível, essa etapa é complementada com alguns passos menores. Os passos sendo o de relacionar relacionar as limitações da plataforma selecionada com o uso de recursos do algoritmo, implementar o algoritmo de acordo com as limitações e restrições da tecnologia, e, por fim analisar a eficiência, o desempenho e a viabilidade do sistema criptográfico implementado na plataforma selecionada. Com isso chega-se à conclusões sobre o uso deste tipo de criptografia na Internet das Coisas e em Arduino, assim como também se descobre quais são as limitações e problemas enfrentados para a utilização de mecanismos de segurança nesta plataforma e também tecnologia.

No início da etapa de desenvolvimento, foram utilizados a partir de todos os passos anteriores os dados e resultados obtidos até agora na fase de pesquisa e definição. Com os dados analisados, e o estudo mais profundo do algoritmo RSA, chegou-se à uma relação entre a implementação do algoritmo e as limitações da plataforma, com isso, gerou-se conclusões de possíveis problemas que seriam encontrados e como resolvê-los para que se consiga fazer o algoritmo funcionar no Arduino, de forma que respeite e se encaixe em suas limitações. Ao completar essa etapa, chegou-se à etapa mais importante do projeto, sendo a etapa crucial para que se obtenha bons resultados, sendo também o objetivo da pesquisa, previamente definida como ”Implementar o algoritmo de acordo com as limitações e restrições da tecnologia”, durante essa etapa, o algoritmo responsável pelo funcionamento do sistema criptográfico RSA foi implementado na plataforma Arduino utilizando-se a linguagem de programação C, em conjunto com algumas necessidades do uso de um nível mais baixo da linguagem utilizando diretamente tipos primitivos de variável do microcontrolador e da linguagem. Durante esta fase, foram encontrados muitas dificuldades, problemas e também limitações devido à arquitetura AVR de 8 bits, sendo a base do Arduino, a qual dificulta muitas vezes a manipulação e armazenamento de dados grandes, tais como os pares de chaves pública e privada do RSA, e também devido à memória e processamento do dispositivo, assim como limitações de bits no tamanho de variáveis. Foram necessárias muitas adaptações, modificações e técnicas para conseguir a implementação total do algoritmo sem problemas ou falhas para que conseguisse fazer a portabilidade do RSA para a plataforma de baixa memória e uso limitado de processamento. A última parte da etapa de desenvolvimento dá-se por analisar a eficiência, desempenho e viabilidade do sistema criptográfico implementado na plataforma selecionada. Esta etapa foi responsável por fazer testes consecutivos para coleta de dados e análise posterior sobre a execução do sistema em placas Arduino afim de se saber qual seu desempenho, se sua aplicabilidade é viável e as razões do uso do mesmo.

Para finalizar o projeto, após a etapa de desenvolvimento, foi realizada a etapa de validação. Essa etapa foi responsável por validar os resultados obtidos da implementação do algoritmo RSA adaptado para a plataforma Arduino e arquitetura AVR. Durante esta etapa, foram executado diversas vezes de maneira exaustiva a encriptação de muitos dados em um dispositivo e decodificados em uma outra placa arduino a qual possui o mesmo par de chaves para troca de mensagens seguras.

3.3 Tecnologia

O mundo da Internet das Coisas é tão amplo e diversificado que pode trazer algumas dificuldades à esse trabalho. A tecnologia IoT depende muito do tipo de dispositivo utilizado, entre eles, estão o Raspberry e o Arduino.

O Raspberry é um dispositivo com arquitetura ARM utilizando muitas vezes como base o sistema operacional baseado em Linux. Esse dispositivo tem sido muito utilizado em casos como prototipação, eletrônica, computador pessoal portátil ou até mesmo controlador de sistemas inteligentes. Sendo uma ótima tecnologia e ferramenta para a obtenção dos resultados pretendidos ao propor um modelo de segurança para este tipo de dispositivo.

O Arduino é outra ferramenta muito útil para se aplicar os resultados deste projeto devido à seu grande uso e facilidade para se desenvolver sistemas inteligentes de maneira simples e barata, sendo escolhida como a plataforma utilizada no projeto devido à essas características. Sendo utilizada neste projeto uma placa Arduino UNO com um microcontrolador ATMEL o qual possui uma arquitetura AVR de 8 bits.

4 DESENVOLVIMENTO

A grande revolução das comunicações e especialmente conexões *wireless* e móvel criou uma grande necessidade de segurança [46]. Sabendo-se que na plataforma IoT há diversos problemas e dificuldades como visto na seção 2.1.3, verifica-se que a maior parte dos problemas ou dificuldades estão relacionados à segurança dessas plataformas.

Isto se dá por fatores como a falta de importância por parte dos desenvolvedores com a segurança em dispositivos IoT [10]. A maior precariedade em segurança para IoT se dá devido ao fato de que essa questão em dispositivos IoT se tornam diferentes do tradicional [9] principalmente devido às grandes limitações de recursos normalmente presentes neste tipo de tecnologia, desta maneira, tornando os dispositivos pobres em segurança e vulneráveis [12]. As limitações de recursos para compensar em características como tamanho, preço, facilidade e flexibilidade são o principal tipo de limitação em IoT como citado e explicado detalhadamente na seção 2.2.

Segundo Singh et al.[47], os algoritmos de criptografia convencionais não encaixam perfeitamente no cenário IoT em razão de inúmeras limitações de recursos e condições, tais como poder de processamento, bateria limitada e execução em tempo real. Em razão das vulnerabilidades em IoT por falta de mecanismos de segurança, sejam eles criptografia ou outros tipos, afetam diretamente os dispositivos e podem causar diversos problemas de segurança como por exemplo, vazamento de informações [48].

Mandal et al.[36] afirma que a segurança da informação é um aspecto importante do compartilhamento de informação através de redes, em razão da falta de mecanismos de segurança e vulnerabilidades devido às limitações dessas plataformas no universo da Internet das Coisas, o presente artigo visa fazer a análise do sistema criptográfico RSA adaptado e implementado para a plataforma Arduino e quais seus problemas e dificuldades em razão das limitações. Sendo muito importante a aplicação de uma criptografia em Arduino pois a criptografia é a técnica para escrever segredos [36]. Algoritmos criptográficos tornam dados e informações seguras de qualquer ataque interno ou externo, enquanto também provê integridade, confidencialidade, incapacidade de negação e autenticidade ao dado. Hossain, Hasan e Skjellum[10] afirma que, são características obrigatórias quando se trata de segurança da informação, características essas detalhadas na seção 2.2.1.

O algoritmo criptográfico RSA foi implementado e portado para a plataforma Arduino, considerando suas restrições e limitações. O sistema criptográfico RSA é o algoritmo de criptografia de chave publica mais usado mundialmente. Ele pode ser usado para codificar uma mensagem sem a necessidade da troca de uma chave secreta separadamente [49]. A criptografia RSA funciona com base em um par de chaves, sendo a chave pública,

a qual é compartilhada por todos, de maneira que todos tem conhecimento, chave essa responsável pela codificação da mensagem, e uma chave secreta, conhecida por chave privada, a qual apenas uma das partes tem conhecimento, assim como observado na Figura 7 e também explicado em detalhes na seção 2.4. A chave pública e a chave privada do RSA, juntas, formam um par de chaves [50].

No presente projeto utilizou-se a plataforma de prototipagem Arduino devido às suas grandes características no mundo da Internet das Coisas. A interface integrada utilizada para a implementação do algoritmo foi a Arduino IDE, utilizando a linguagem C como base para programação e desenvolvimento. Devido à isto, o algoritmo implementado a ser explicado à seguir possui trechos do código em linguagem C em conjunto com o uso da biblioteca BigNumber. O algoritmo RSA funciona em 3 passos:

- **Algoritmo de geração de chaves:** durante a etapa de geração de chaves, é necessário encontrar e computar 2 números primos distintos p e q, o módulo (n), o valor totiente de n, conhecido como $\varphi(n)$, o expoente público e e o privado (d).
- **Algoritmo de codificação:** Este algoritmo é responsável por utilizar a chave pública para transformar o texto plano original m em uma cifra codificada c.
- **Algoritmo de decodificação:** Esta função do algoritmo RSA utiliza a chave privada para transformar o texto cifrado c em uma mensagem de texto plano m.

4.1 Geração de chaves

O algoritmo de geração de chaves do sistema criptográfico RSA se baseia em 5 passos. De acordo com o código implementado a seguir, cada etapa será devidamente detalhada e explicada, assim como o código desenvolvido 4.1. Esses passos são responsáveis pelos cálculos dos valores de chaves pública e privada, sendo eles, obter os valores primos para que se compute o módulo n, se obtenha o Phi(n) $(\varphi(n))$ para encontrar o expoente público e o privado, valores responsáveis por compor o par de chaves.

Algoritmo 4.1 – Algoritmo de geração do par de chaves.

```
void generate_key_pair(uint64_t* public_key,
        uint64_t* private_key, int k) {
        uint32_t p;
        uint32_t q;
        uint64_t n;
        do{
                p = prime_number(k / 2);
```

```
            q = prime_number(k - (k / 2));
        }while( ((n = (uint64_t) p * (uint64_t) q) < 95 ) ||
        (p == q));

        uint64_t phi_n = (uint64_t) totient(p) *
        (uint64_t) totient(q);
        uint64_t e = generate_e(phi_n);
        uint64_t d = generate_d(e, phi_n);

        public_key[0] = n;
        public_key[1] = e;
        private_key[0] = n;
        private_key[1] = d;
}
```

4.1.1 Geração dos primos P e Q

A primeira etapa do algoritmo de geração de chaves, se baseia na geração e teste de primos, sendo esta etapa definida por: Escolha dois números primos distintos, sendo eles p e q de tamanho aproximado.

Algoritmo 4.2 – Geração de primos p e q.

```
uint32_t p = prime_number(k / 2);
uint32_t q = prime_number(k - (k / 2));
```

Algoritmo 4.3 – Função *prime_number*.

```
        uint64_t prime_number(int modulus) {
        uint64_t value= 1;
        for(int i = 0; i< modulus / 4; i++){
            value *= random(1, ipow(2, 4));
        }
        while (!isprime(value)) {
            value += 1;
        }
        return value;
}
```

Algoritmo 4.4 – Função *isprime*.

```
bool isprime(uint64_t number) {
        for (uint64_t i = 2; i <= (uint64_t) sqrt(number); i++) {
                if (number % i == 0) {
                        return false;
                }
        }
        return true;
}
```

A função **prime_number** [4.3] é responsável por gerar um número primo de até k bits, funcionando da seguinte maneira: Um número aleatório é gerado através do método **random** (nativo do arduino), esse número gerado está entre 1 e 2^k. Sendo assim, o valor gerado está dentro do intervalo $1 <$ **value** $< 2^k$. Após a geração do valor, há um laço de repetição utilizando a função **isprime** [4.4] é responsável por fazer a verificação se o número passado como argumento é primo, caso não seja, ele incrementa o valor em 1 e testa novamente. Esta solução foi utilizada devido ao problema resultante de que quando se utiliza de um valor de k bits elevado, demanda muito processamento e tempo para encontrar um número primo muito grande. Desta maneira, a solução utilizada foi gerar um número grande qualquer de k bits, e ir testando os próximos valores para encontrar o primo mais próximo dele. O algoritmo de teste de primos 4.4 é simples, ele faz a verificação se existe resto da divisão do valor por todos os valores até sua raiz quadrada, em caso do resto for igual a 0, ele retorna falso, indicando que o valor não é um primo, caso contrário, ele continua a verificação até o fim do laço de repetição, retornando true se o valor for um número primo.

4.1.2 Computar o módulo n

Pelo algoritmo, a segunda etapas é obter n, este valor é chamado de módulo, tal que n é o produto $n = pq$. Desta maneira, o algoritmo completo de 4.2, se dá pela seguinte implementação:

Algoritmo 4.5 – Encontrar o produto $n = pq$.

```
uint32_t p;
uint32_t q;
uint64_t n;
        do{
                p = prime_number(k / 2);
                q = prime_number(k - (k / 2));
```

```
        }while( ((n = (uint64_t) p * (uint64_t) q) < 95 ) ||
        (p == q));
```

O valor de n não pode exceder k bits, tal que k é a quantidade de bits da chave, por exemplo (8, 16 ... 1024, 2048). Em razão da multiplicação de dois valores, tanto p quanto q, devem ter $k/2$ bits, para que o produto $n = pq$ não exceda k bits. Pelo algoritmo, têm-se que p e q precisam ser dois primos distintos, desta maneira, o laço gera novos valores para p e q sempre que ambos forem iguais. Pela construção do RSA, dá-se que o valor a ser cifrado (m) precisa estar no intervalo entre 1 e n, sendo assim $1 < m < n$. Neste projeto utiliza-se cada caractere individual da mensagem com seu valor ASCII. Desta forma, utiliza-se apenas os valores printáveis, para isso, leva-se em consideração que os valores printáveis vão de 32 à 127, têm-se 95 valores, em razão disso, faz-se a normalização da tabela ASCII durante e codificação e decodificação, tornando os valores de 1 à 95. Devido ao fato de o valor de m (1 à 95 ASCII) precisar estar entre 1 e n, faz-se a verificação para que enquanto o valor de n for menor que 95, gere-se novos p e q.

4.1.3 Calcular o totiente de Euler de n $\varphi(n)$

O valor de φ se dá pelo totiente de n, isso significa que $\varphi(n) = (p-1)(q-1)$. Desta maneira, a função para que se retorne o valor totiente de um número, é representada em 4.7, a qual verifica se o número é primo através da função 4.4, e em caso positivo, retorna o próprio número decrementado de 1. Desta maneira, em 4.5, o valor de φ é obtido por $\varphi(n) = totiente(p) * totiente(q)$.

Algoritmo 4.6 – Parte do código relacionado à $\varphi(n)$ apresentado em 4.1..

```
uint64_t phi_n = (uint64_t) totient(p) * (uint64_t) totient(q);
```

Algoritmo 4.7 – Função Totiente.

```
uint64_t totient(uint64_t n) {
        if (isprime(n)) {
                return n - 1;
        }
}
```

4.1.4 Encontrar um expoente público e

Após ter os valores primos de p e q, encontrar o valor de n e também o valor de φ, o próximo passo para a geração de chaves é encontrar o expoente público e, valor esse, o

qual junto de n, é conhecido por ser a chave pública. A chave pública do RSA, é conhecida por ser a combinação de (n, e), ou seja, o módulo do RSA (n) e o expoente público (e). Pelo fato desse valor ser compartilhado e de acesso público, existem duas opções para a definição do mesmo. O valor do expoente público e, pode ser dado estaticamente através de valores predefinidos, sendo eles 3, 5, 17, 257, 65537. Esses valores são recomendados para serem utilizados, pois já que o valor de e é publico e também enviado através da rede, não há problema algum em qualquer interceptador descobrir, devido à isso, esses valores de acordo com a teoria dos números, são mais fáceis computacionalmente de se utilizar em cálculos. A outra alternativa, e a forma utilizada neste projeto, é calcular o valor de e. As regras para se escolher um expoente público e, são dadas pela necessidade de que e seja menor que $\varphi(n)$ e um inteiro positivo, assim também como é obrigatório ele ser um coprimo de $\varphi(n)$, ou seja, um valor que o máximo divisor comum entre e e $\varphi(n)$ seja 1. Desta maneira, em notação matemática formal, sabe-se que o valor a ser encontrado, necessita ser $1 < e < \varphi(n)$, ser coprimo com $\varphi(n)$, o que significa $mdc(e, \varphi(n)) = 1$.

Algoritmo 4.8 – Função *generate_e*.

```
uint64_t generate_e(uint64_t phi_n) {
        for (uint64_t i = 2; i < phi_n; i++) {
                if (mdc(i, phi_n) == 1) {
                        return i;
                }
        }
}
```

Algoritmo 4.9 – Função *mdc*.

```
uint64_t mdc(uint64_t value_one, uint64_t value_two) {
        uint64_t rest = 1;
        uint64_t max_value = max(value_one, value_two);
        uint64_t min_value = min(value_one, value_two);

        while (min_value != 0) {
                rest = max_value % min_value;
                max_value = min_value;
                min_value = rest;
        }
        return max_value;
}
```

A função para encontrar o valor do expoente público 4.8 funciona através do uso do valor de $\varphi(n)$ como argumento para verificar qual valor entre 1 e $\varphi(n)$ obedece à afirmação $mdc(e, \varphi(n)) = 1$. Para isso, usa-se um laço de repetição de 2 à $\varphi(n)$ e é efetuado a verificação se o máximo divisor comum 4.9 entre i e $\varphi(n)$ resulta em 1, em caso de positivo, o valor de e é encontrado e retornado para o algoritmo de geração de chaves [4.1].

Algoritmo 4.10 – Parte do código relacionado à e apresentado em 4.1.

```
uint64_t e = generate_e(phi_n);
```

4.1.5 Computar o expoente privado d

O último passo de cálculos necessários para os valores das chaves no algoritmo RSA é o de encontrar o expoente privado d, valor o qual constitui a chave privada em complemento com n. Desta forma, a chave privada é formada por (n, d). O valor de d deve ser computado onde $1 < d < \varphi(n)$ e a congruência $ed \equiv 1(mod\phi(n))$. Assim, chega-se a conclusão que o expoente privado d, seja o inverso multiplicativo de e em $(mod\varphi(n))$. Para que se alcance este resultado, é utilizado o algoritmo de Euclides estendido e o conceito de inverso multiplicativo, proveniente da aritmética modular.

Algoritmo 4.11 – Parte do código relacionado à d apresentado em 4.1.

```
uint64_t d = generate_d(e, phi_n);
```

Algoritmo 4.12 – Função *generate_d*.

```
uint64_t generate_d(uint64_t e, uint64_t phi_n) {
        uint64_t d = 0;
        while (mod(d * e, phi_n) != 1) {
                d++;
        }
        return d;
}
```

Algoritmo 4.13 – Função *mod*.

```
uint64_t mod(uint64_t x, uint64_t y) {
        if (x < y) {
                return x;
```

```
        }
        return x % y;
}
```

Para fazer o cálculo do valor do expoente privado d, usa-se como base o expoente público d e também $\varphi(n)$. A função implementada para a geração do valor de d, utiliza o módulo inverso, através do algoritmo funcionando de maneira em que d começa valendo zero, e é incrementado sucessivamente por 1, até que $(d * e) mod \varphi(n)$ seja igual à 1, pois desta maneira, ele obedece à congruência $ed \equiv 1 (mod \phi(n))$.

4.1.6 Chave Pública e Privada

Ao final do algoritmo de geração do par de chaves obtém-se todos os valores necessários para montar as chaves pública e privada, também chamadas de chave de codificação e chave de decodificação.

Algoritmo 4.14 – Parte do código relacionado à montagem das chaves apresentado em 4.1.

```
public_key[0] = n;
public_key[1] = e;

private_key[0] = n;
private_key[1] = d;
```

Utilizando dos valores encontrados, monta-se a chave pública através do par (n, e), encontrados respectivamente como o módulo (n) e o expoente público (e). Em seguida, a chave privada é montada através do par (n, d), sendo eles os valores encontrados como o módulo, assim como o utilizado na chave pública, e o o expoente privado d. Neste momento, a chave pública já está apta à ser compartilhada com outros dispositivos, e a privada mantida secreta pelo recipiente, assim como os valores d, p, q e o valor φ devem ser mantidos secretos. Com as chaves geradas e montadas, cada uma das duas tem sua responsabilidade própria no sistema criptográfico RSA, sendo a chave pública responsável por fazer a codificação, e a privada a decodificação.

4.2 Codificação

O algoritmo de codificação do sistema criptográfico RSA, está presente tanto no recipiente quanto no receptor, ele transforma uma mensagem m, em uma cifra c.

Algoritmo 4.15 – Função *encrypt*.

```
void encrypt(char* message, uint64_t* public_key,
```

```
uint64_t* encrypted_message, int message_size) {
        uint64_t buffer;
        for (int i = 0; i < message_size / sizeof(uint64_t); i++) {
                if (i < strlen(message)) {
                        int char_value = (int)message[i];
                        buffer = ipow(char_value, public_key[1])
                        % public_key[0] ;
                        encrypted_message[i] = buffer;
                } else {
                        encrypted_message[i] = 0;
                }
        }
}
```

Algoritmo 4.16 – Função *ipow*.

```
uint64_t ipow(uint64_t a, uint64_t b) {
        uint64_t answer = 1;
        for (; b--;)
        answer *= a;
        return answer;
}
```

A função de codificação, utiliza como base uma mensagem m, e a chave pública (n, e), onde é feita a aplicação de uma potenciação modular, para que a mensagem m torne-se uma cifra c. Como explicado na seção 4.1.2, um vetor de char é desmontado, e cada caractere contido na mensagem, é transformado em uma mensagem m utilizando seu código na tabela ASCII como base, em seguida, como o algoritmo implementado utiliza apenas os caracteres printaveis num alcance de 32 à 127, devido à isso, foi feita a normalização dos valores retirando 95 do valor ASCII de cada mensagem m. Este valor ASCII, após a normalização, é transformado em um int, onde a mensagem necessita ser um inteiro positivo m em $1 < m < n$ para que seja efetuado o cálculo de codificação através da aplicação da potenciação modular onde $c = m^e \mod n$. A função *ipow* [4.16], responsável por fazer a potenciação, foi implementada e utilizada de forma independente devido à erros de conversão e precisão resultantes do tipo de retorno do método *pow* original. O algoritmo de codificação armazena a mensagem m codificada em um vetor de inteiros o qual será o valor resultante de cada caractere (mensagem) cifrado. Ao final da função o valor armazenado em *encrypted_message* está pronto para ser enviado ao destinatário ou ao recipiente de maneira segura e com seu valor secreto.

4.3 Decodificação

O algoritmo de decodificação está presente apenas no recipiente, o qual é o único conhecedor e portador da chave privada (n, d), e também dos valores gerados por ele. Em posse da chave privada, o recipiente é capaz de decodificar qualquer mensagem vindo de destinatários que utilizaram a sua chave pública para codificar a mensagem. Assim como na codificação, a decodificação também é feita através da potenciação modular $m = c^d \mod n$, tal que m é a mensagem resultante após a decodificação, c é a mensagem cifrada recebida, e os valores d e n formando a chave privada. Após a execução desse algoritmo, é possível obter o texto puro e original enviado. Ao contrário da codificação, a decodificação requer um maior nível de processamento, memória e também tempo, implicando nas restrições do Arduino, e também dos recursos disponíveis.

Algoritmo 4.17 – Função *decrypt*.

```
void decrypt(uint64_t* encrypted_message, uint64_t* private_key,
char* decrypted_message) {
        int i = 0;
        BigNumber result;
        do {
                result = ipowbig(encrypted_message[i],
                private_key[1]) % BigNumber(private_key[0]);
                decrypted_message[i] = result;
                result = 0;
                i++;
        } while (encrypted_message[i] > 0);
}
```

Algoritmo 4.18 – Função *ipowbig*.

```
BigNumber ipowbig(uint64_t base, uint64_t expoente) {
        BigNumber  answer = 1;
        BigNumber bigA = base;
        for (int i = 0; i < expoente; i++) {
                answer = answer * bigA;
        }
        return answer;
}
```

O algoritmo implementado necessitou do uso da biblioteca BigNumber [51] em razão de que diferente da codificação, que o valor da potência normalmente é baixo, devido ao valor do e ser um valor mínimo. Quando executado a decodificação, o valor resultante da potenciação modular $m = c^d \mod n$ se torna muito grande, ultrapassando até mesmo os 64 bits máximo à ser utilizados pelo Arduino e por C. Devido à isso, a biblioteca BigNumber faz operações matemáticas e armazena valores de até 256 bits. Para que se executasse a potenciação com valores tão grandes, foi criado a função *ipowbig* [4.18], a qual é responsável por executar a potenciação de grandes números usando o BigNumber como base. Na função de decodificação, é utilizado um laço para percorrer todos os valores do vetor de mensagem criptografada, em seguida é executado a potenciação modular a qual resulta em uma mensagem limpa de criptografia e esse valor m é armazenado em um vetor de char para que se utilize a mensagem original e legível.

4.4 Dificuldades e soluções na implementação

Durante a implementação das 3 etapas do sistema criptográfico RSA, foram encontrados muitos desafios e barreiras a vencer devido às limitações presentes na plataforma. A tecnologia Arduino possui uma arquitetura AVR baseada em 8 bits, sendo uma arquitetura muito simples e com um limite pequeno de bits, devido à este fator, torna-se complexo manipular dados acima de 8 bits, necessitando sempre de uma adaptação. Para o sistema RSA, números pequenos são insignificantes pelo fato de sua extrema segurança dar-se em razão dos longos e grandes números utilizados para as chaves.

4.4.1 Tipos de dados

Ao iniciar a implementação, já começaram os problemas e dificuldades, o primeiro deles foi em relação às variáveis e a capacidade de armazenamento de dados nelas. Não possuindo tipos de variáveis com uma grande capacidade de armazenamento, houve uma grande dificuldade, utilizou-se de inteiros, inteiros não sinalizados, longos, longos não sinalizados e por fim, foi utilizado um tipo chamado de *long long* devido à sua capacidade de 64 bits, indo do valor 2^{-32} até 2^{32}. Porém o algoritmo RSA utiliza apenas inteiros positivos, e havia a necessidade de que armazena-se valores de até 2^{64}, com isso, o *long long* apesar de ter capacidade de 64 bits, não resolvia os problemas. Com isso, após muita pesquisa e estudo sobre C e Arduino, foi utilizado o tipo de variável *uint64_t*, tipo este, capaz de armazenar 64 bits não sinalizados, ou seja, em um alcance de 2^1 até 2^{64}.

4.4.2 Velocidade de processamento

Houveram problemas os quais surgiram durante a fase de implementação do algoritmo de geração de chaves com o cálculo de números primos [4.2]. O cálculo de números primos de grande ordem são complexos e difíceis, executar esses cálculos para se gerar um

grande número primo se torna ainda mais trabalhoso e demorado em um dispositivo com baixos recursos como o Arduino, devido à isto, havia um problema, o algoritmo gera um número grande de ordem até 2^{64}, e em seguida, testa se ele é primo através do método padrão de verificação de primos [4.4], com isso, quando crescia o tamanho do número, tornava-se incalculável pelo processador do Arduino, e se tornava inviável em razão de tempo. Para solucionar a questão, utilizou-se de uma tática diferente, um número aleatório é gerado [4.3], assim como anteriormente, e ao invés de verificar se ele é primo, e caso não seja, gerar outro, assim sucessivamente, gera-se um único número de ordem grande, caso ele não seja primo, o algoritmo encontra o próximo primo após aquele número [4.3], isto reduz o tempo gasto, memória utilizada e poder de processamento.

4.4.3 Conversão de tipos

Métodos básicos e nativos da própria linguagem C e plataforma Arduino tiveram de ser reimplementados devido à incompatibilidades ou problemas resultantes da conversão. Os métodos pow [4.16] e mod [4.13] foram reimplementados manualmente. O método *pow* estava gerando resultados incorretos pois seu tipo de retorno, assim como seus argumentos são do tipo *double*, o qual possui uma capacidade de 64 bits sinalizados, desta maneira, possuindo um intervalo de $2-32$ à 2^{32} resultando em problemas de conversão e precisão quando utilizado um *uint64_t*, com isso, gerando resultados inesperados e incorretos. A função *mod* precisou ser reimplementada pelo mesmo motivo, ao utilizar o operador % (mod) obtinha-se o resultado entre dois inteiros, desta maneira, esta função também foi reimplementada para ser capaz de trabalhar com números de maior ordem.

4.4.4 Problemas de memória

Durante a implementação e utilização do algoritmo em Arduino, o que mais se depara é com problemas de memória, desde a pouca quantidade, resultante de seus apenas 2048 bytes no modelo UNO, e 8196 na versão MEGA. Esses problemas vão de falta de memória até fragmentação e estouro da memória mesmo com disponibilidade. A baixa quantidade de recursos, faz com que o dispositivo não tenha uma central de gerenciamento de memória para que se mantenha otimizada sua SRAM, desta maneira, ao ficar alocando e desalocando variáveis, principalmente de maneira dinâmica, a memória começa a ficar muito fragmentada, com pequenos espaços entre cada bloco, de maneira que se tornem impreenchíveis mais tarde, com isso, mesmo com memória disponível, muitas vezes houveram problemas com impossibilidade de alocação ou falta de memória.

Ao fazer a decodificação [4.17], é necessário muita memória devido ao fato de que a base e o expoente utilizados para o cálculo da chave privada utilizam valores muito altos, desta maneira, o resultado se transforma em um número extenso, gerando um estouro de memória, muitas vezes mesmo com memória disponível. Este acontecimento

se dá pela potenciação com números de alta ordem, para isso, com o limite de 64 bits do tipo *uint64_t*, necessitou-se de uma alternativa capaz de armazenar um valor acima deste tamanho, com isso, foi utilizada a biblioteca *BigNumber* [51], capaz de armazenar números de até 256 em uma arquitetura Arduino de 8 bits, assim como também executar operações sobre esses números. Porém o uso desta biblioteca deve ser bem controlado e de maneira cautelosa, pois seu código utiliza alocação dinâmica à cada modificação feita, gerando muita fragmentação. O problema com o cálculo de um número muito grande foi resolvido com a biblioteca *BigNumber*, porém, com a memória presente no Arduino, torna-se inviável a utilização de uma chave de 16 bits ou superior, o principal problema se dá pela falta de memória, estourando assim o limite disponível, também há problema com fragmentação, que quanto mais ele calcular, enquanto o Arduino não for reiniciado, continuará abrindo mais espaços muitas vezes impreenchíveis em pequenos blocos, tornando a memória ainda menos capacitada. Por fim, durante a decodificação mesmo com o *BigNumber*, muitas vezes a potenciação de um número por outro, leva á resultados acima de 256 bits, desta maneira, impossibilitando o cálculo e o armazenamento. A utilização desta biblioteca para manipular números grandes foi essencial para o projeto, por sua capacidade de armazenar esses altos valores para que se calcule a decodificação, porém, ela causa mais fragmentação de memória ao utilizar a decodificação acima de 8 bits. Em razão disso, a utilização de um valor acima de 16 bits para o tamanho da chave torna-se inviável executar o algoritmo de decodificação devido às restrições presentes. Porém com o limite padrão da plataforma de até 64 bits, o dispositivo é capaz de gerar as chaves de até 64 bits e também executar o método de codificação [4.15], mesmo sendo incapaz de processar a decodificação.

5 RESULTADOS E VALIDAÇÃO

Os resultados foram obtidos utilizando a execução dos 3 algoritmos implementados sendo eles o algoritmo de geração de chaves, de codificação e o de decodificação, fazendo inúmeras iterações consecutivas. Desta forma, utilizou-se uma ferramenta chamada "RSA Calculator" [52] para comprovar a consistência do algoritmo implementado com o original.

5.1 Testes e Resultados

Os testes foram realizados através da análise de dados obtidos como memória utilizada, tempo de processamento, tamanho de chave e tamanho da mensagem. Os testes para obtenção de dados e resultados importantes foi feito através da execução de 1000 iterações entre os métodos de gerar os pares de chave, codificar e decodificar utilizando diferentes valores, desta maneira analisando o tempo necessário para cada algoritmo em relação ao tamanho de suas variáveis. O teste foi realizado utilizando uma chave de tamanho de 8 bits, baseando no uso dos algoritmos 1000 vezes para a coleta de dados de cada iteração para o uso na análise.

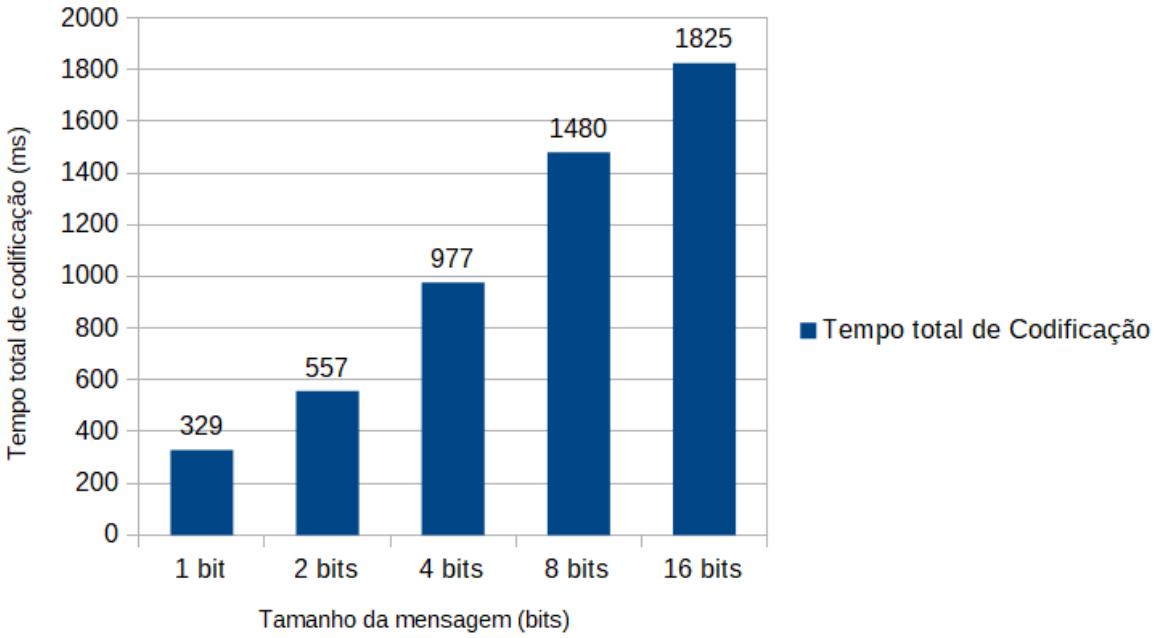

Figura 9 – Relação entre tamanho da mensagem e tempo para codificação

No gráfico é apresentada a comparação entre a quantidade de bits m de uma mensagem em relação ao tempo necessário para codificá-la. Para isto, foram executadas 1000 iterações utilizando o algoritmo de codificação. Ao final, observa-se que apesar do tempo

crescer aumentando a quantidade de bits em uma mensagem, este valor relativamente decai. Para codificar uma mensagem de apenas um bit ”a” por 1000 vezes, levou-se um tempo total de 329 milissegundos, para a codificação de uma mensagem de 2 bits, sendo eles ”ab”, o tempo necessário para as 1000 iterações foi de 557 milissegundos. Com uma mensagem de 8 bits sendo ela ”abcdefgh”, a execução de 1000 iterações custou 1480ms (1,48 segundos) e o mesmo teste com uma mensagem de 16 bits levou 1825ms para a codificação. Com esses resultados, nota-se que apesar do valor de tempo necessário para codificação crescer conforme o tamanho da mensagem, o algoritmo se torna mais rápido, pois se um único bit precisa de 329ms para ser codificado, caso executado o algoritmo independentemente em uma mensagem de 16 bits, seria necessário uma média de 5264ms, enquanto o algoritmo levou apenas 1825ms para a codificação total.

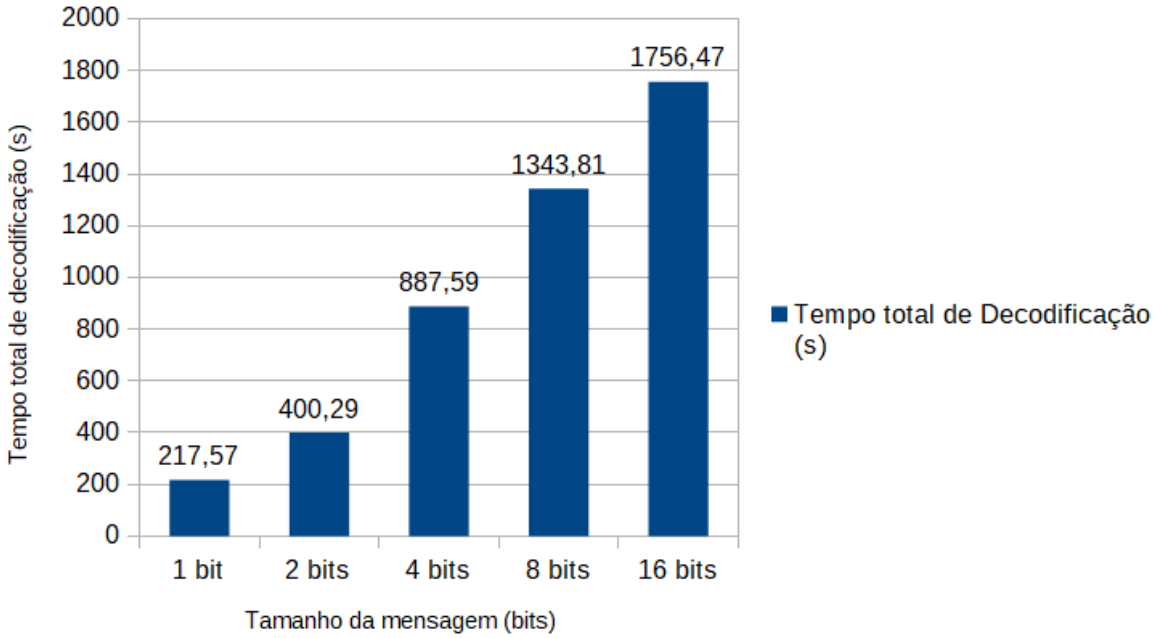

Figura 10 – Relação entre tamanho da mensagem e tempo para decodificação

O custo de tempo para a decodificação é muito maior do que a codificação como observado nos dados e nos gráficos, isso se dá pelo fato de que o expoente privado utilizado no cálculo para decodificação é de ordem muito maior que o expoente público utilizado na codificação. Enquanto na codificação era possível trabalhar na base de milissegundos por seu rápido desempenho para executar o algoritmo, na decodificação é usada a base de tempo em segundos para melhor entendimento e análise dos dados. O tempo necessário para a decodificação de uma mensagem de 16 bits ou caracteres nos testes executados em 1000 iterações se dá por um valor muito grande, sendo ele 1756 segundos, sendo o equivalente à aproximadamente 30 minutos de processamento.

Através de todos os testes e dados, houveram mais execuções para obtenção de médias de tempo para cada algoritmo, as execuções funcionaram com base de 1000 iteração

para cada algoritmo independente, sendo eles o codificação e decodificação, os dados obtidos, assim como a análise dos gráficos podem ser vistas à seguir:

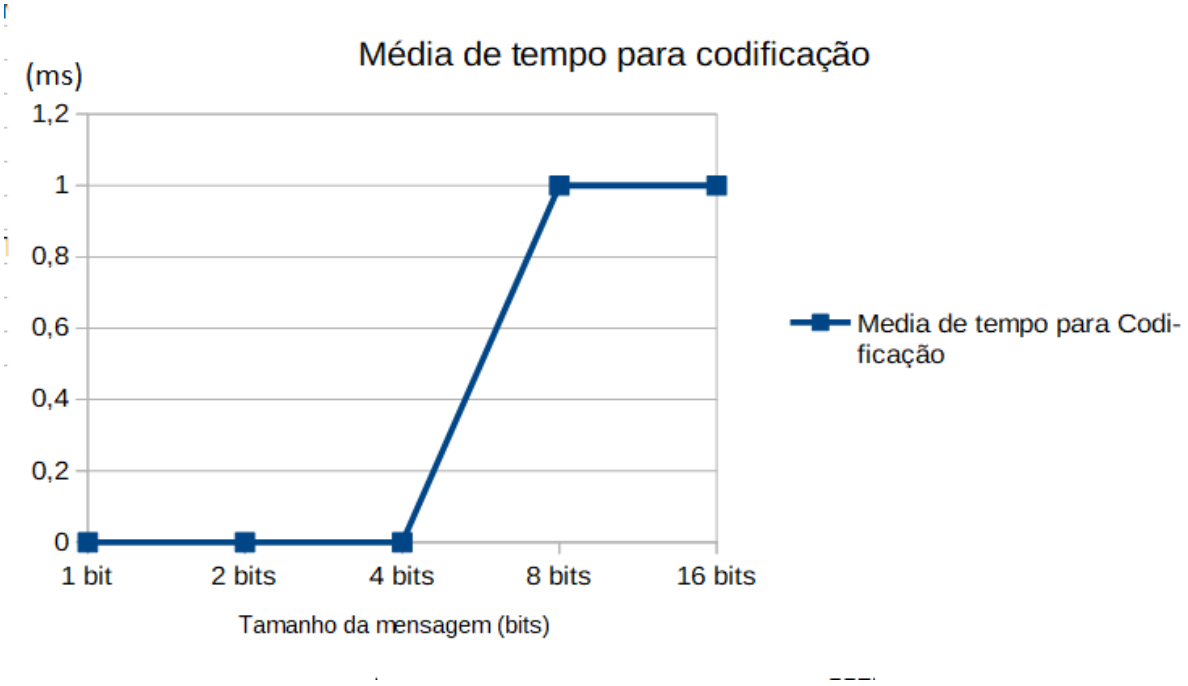

Figura 11 – Relação entre tamanho da mensagem e tempo médio para codificação

A média de tempo para a execução do algoritmo de codificação foi obtida através do tempo total para codificação no teste de 1000 iterações 9, desta maneira, verifica-se que o tempo necessário para codificar uma mensagem de até 4 bits é menor que 1ms, valor este tão pequeno que tende à zero. Com os testes de 8 e 16 bits, o tempo necessário se torna acima de 1ms.

Os valores de média de tempo para decodificação de mensagens de diferentes tamanhos também foram obtidos através dos dados coletados em 10. O algoritmo apresentou bons desempenhos em relação às limitações do dispositivo, tendo a vantagem de que quanto maior o tamanho da mensagem, menor é o tempo em relação ao caso de se executar os algoritmos mais vezes com mensagens menores. Este fato faz com que a decodificação de uma mensagem de 16 caracteres (bits) não tenha um grande tempo, esse custo de tempo para a decodificação gira em torno de 1,76 segundos.

A relação direta entre o tamanho em bits da chave e o tempo de execução do algoritmo para a geração do par de chaves se dá em relação do complexo cálculo de muitos dados na etapa do algoritmo de geração de chaves. Desta maneira, com os dados coletados, enquanto o algoritmo de geração do par de chaves leva apenas 5ms com um tamanho de chave de 8 bits, o mesmo demora mais de 23 segundos para gerar um par de chaves de 64 bits, estes valores são relativamente altos, mesmo em relação às limitações e os recursos baixos da plataforma Arduino.

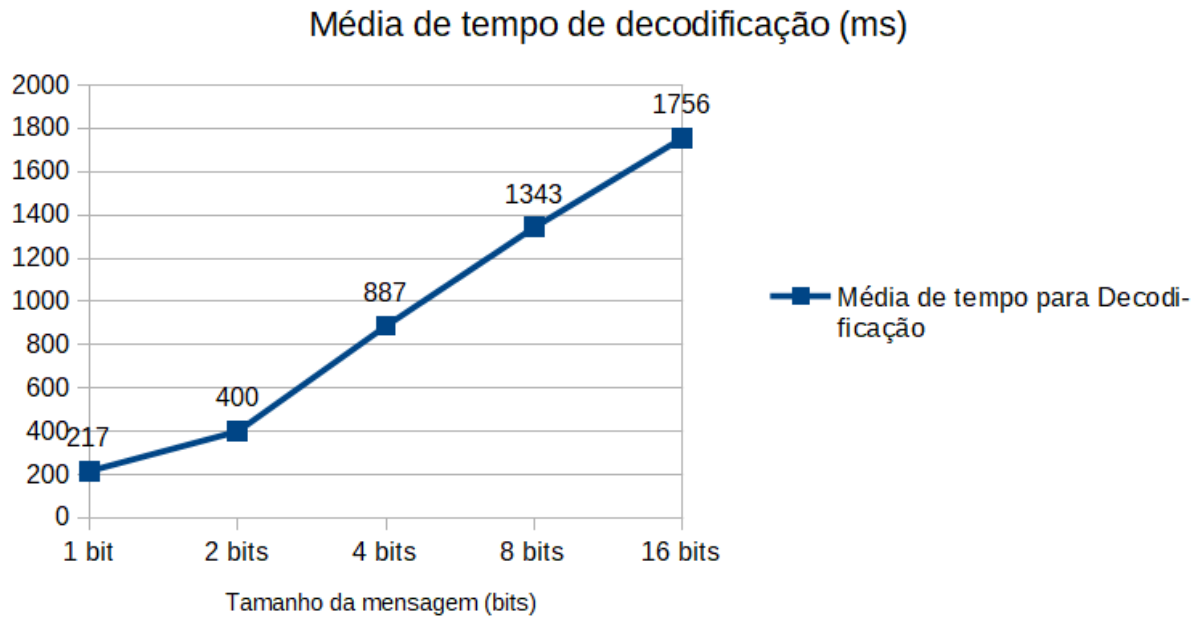

Figura 12 – Relação entre tamanho da mensagem e tempo médio para decodificação

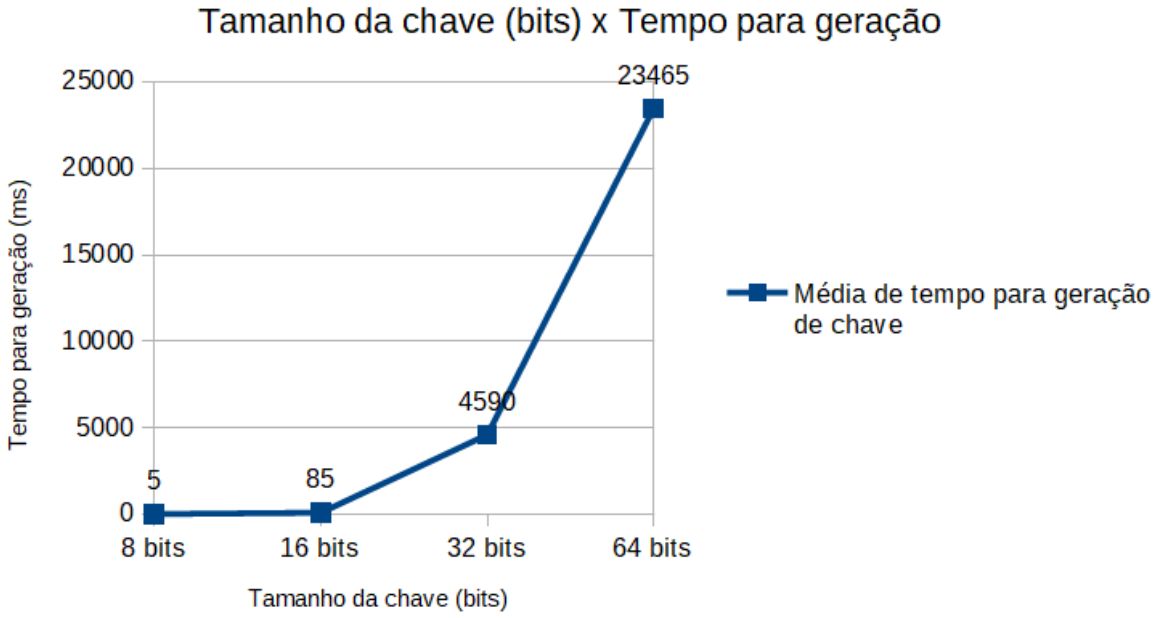

Figura 13 – Relação entre tamanho da chave e tempo para geração do par de chaves

Levando em consideração que a evolução exponencial ocorre na medida em que aumenta o tamanho da chave, e também da mensagem, aumentando assim o tempo necessário para as operações, em seu trabalho Andrade e Silva[1], apresenta os dados de sua pesquisa aplicando o algoritmo RSA em uma mensagem com 100 caracteres, utilizando módulos de 1024, 2048 e 4096 bits.

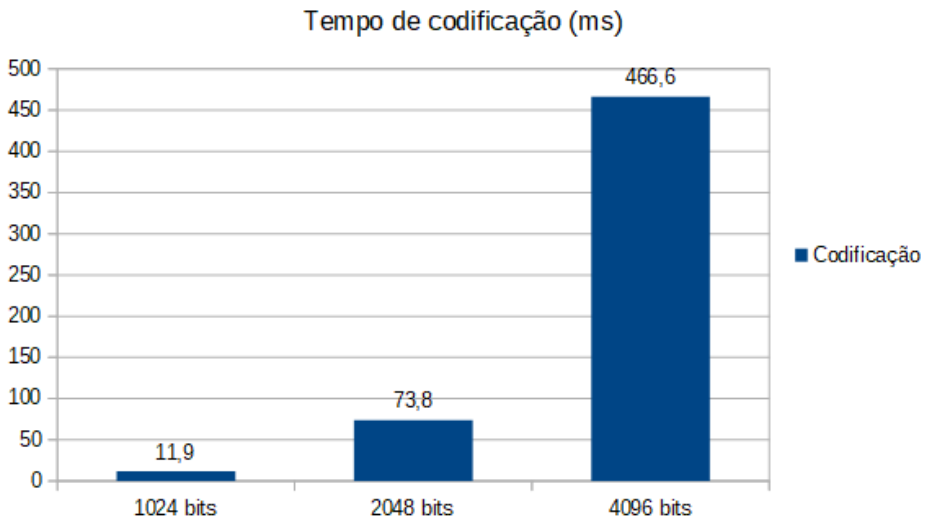

Figura 14 – Tempo para codificação de Andrade e Silva[1]

A figura 14 apresenta os dados do tempo de codificação em uma plataforma sem restrições através da implementação em C, dessa maneira, em comparação com os resultados de uma chave de apenas 8 bits como apresentado na figura 11 que leva em torno de 1ms, o algoritmo no Arduino alem de suas grandes limitações, possui uma velocidade de codificação bem mais baixa.

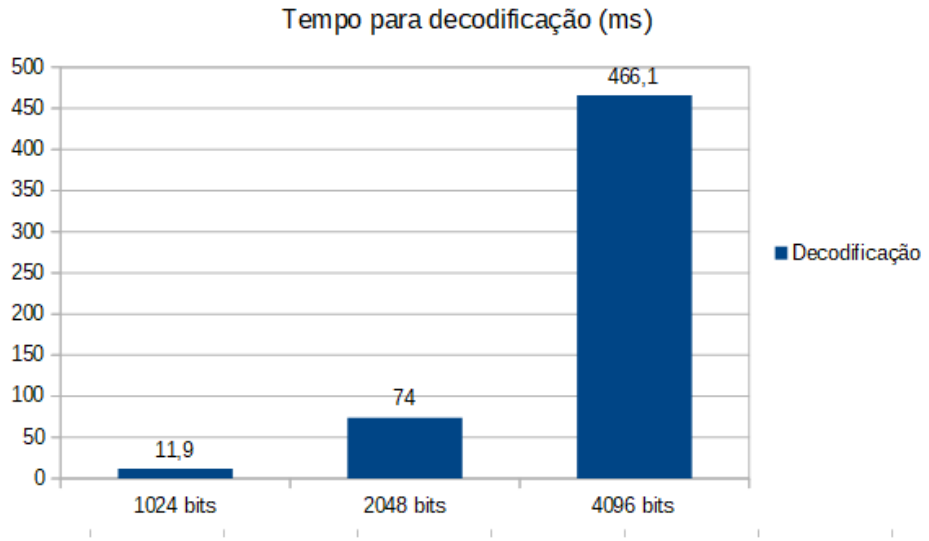

Figura 15 – Tempo para decodificação de Andrade e Silva[1]

A maior diferença vem no tempo de decodificação como apresentado na figura 15, ao relacionar com os dados obtidos neste projeto, onde uma chave de apenas 8 bits, utilizando uma mensagem de 16 caracteres leva até 1756ms para a decodificação total da mensagem na figura 12. O maior valor obtido na pesquisa de Andrade e Silva[1], foi de apenas 466,1ms ao aplicar uma chave de 4096 bits em uma mensagem com 100

caracteres. Com isso, nota-se a tamanha dificuldade da plataforma Arduino em processar a decodificação de uma mensagem com o algoritmo RSA.

5.2 Validação

A validação foi feita através do uso da ferramenta RSA Calculator [52], esta calculadora de RSA provê um passo a passo de cada valor até que se obtenha as chaves pública e privada, e desta maneira, também validando o par de chaves. Para dar início à validação, primeiro executou-se os algoritmos de geração de chaves, de codificação e decodificação. Ao gerar o par de chaves, obteve-se os valores necessários para que se chegue à chave pública e privada, sendo esses valores os responsáveis por validar o algoritmo implementado utilizando a ferramenta, esses valores são:

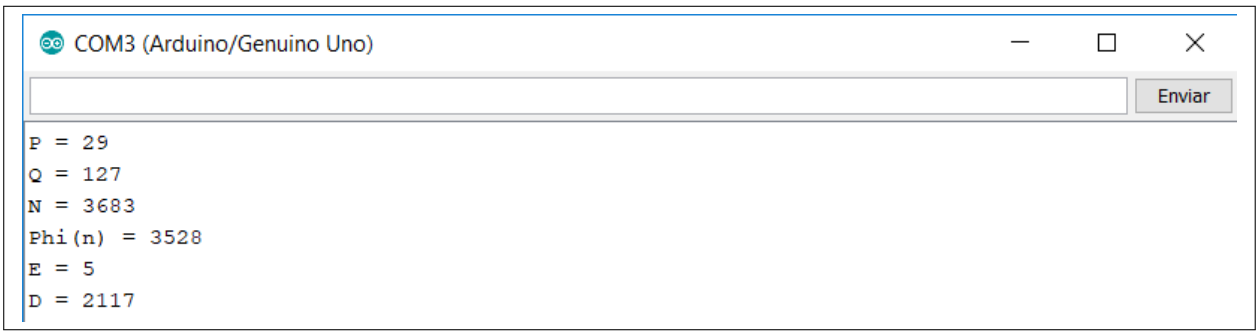

Figura 16 – Valores resultantes do algoritmo implementado.

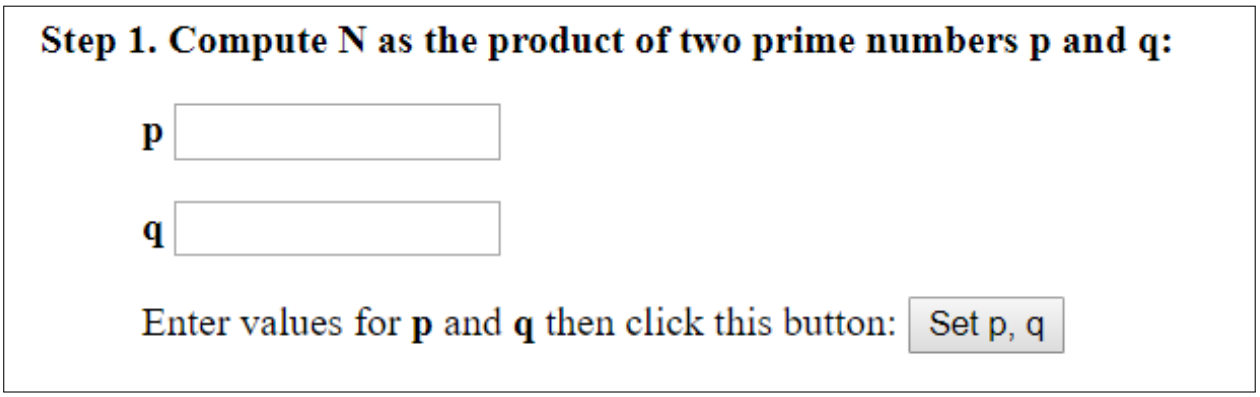

Figura 17 – Passo inicial para validação.

Como pode ser observado na imagem 17, o primeiro passo da geração de chaves, sendo ele encontrar p e q, e o segundo passo computar o módulo n, pode ser utilizado nesta ferramenta. Primeiro basta inserir o valor de p e q gerado e utilizado. Como exemplo de validação, serão utilizados os primos p e q de valor 29 e 127. Ao inserir os valores e utilizar o botão "Set p, q" como visto na figura 17, a calculadora automaticamente calculará o valor do módulo n, resultado este, visível na figura 18.

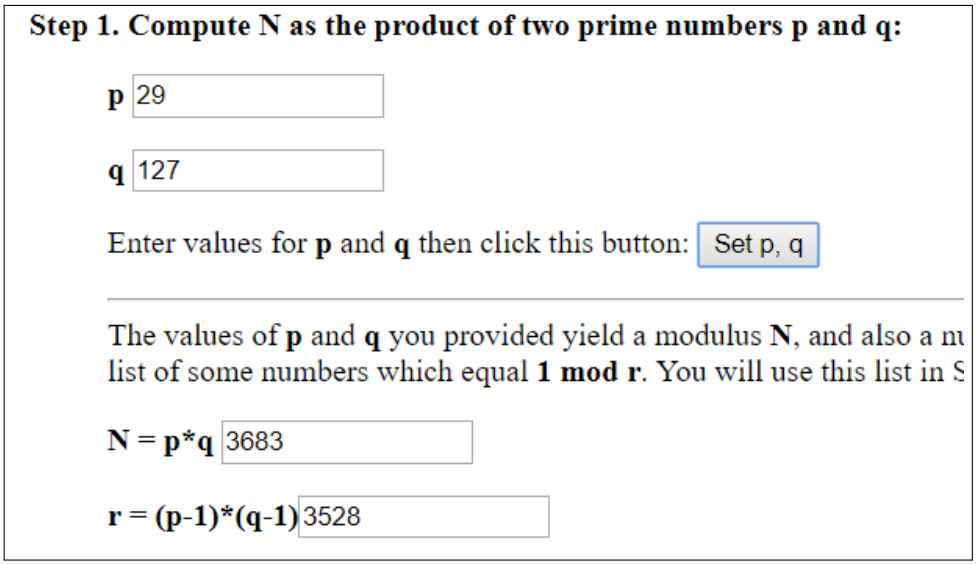

Figura 18 – Valor resultante da ferramenta de validação

Para prosseguir com a validação, todos os valores até agora gerados e calculados procederam com os da calculadora. No próximo passo, é necessário encontrar 2 números que são co primos ao módulo N e o qual $ed = 1 mod r$, passo esse, necessário para encontrar d.

Os valores gerados pelo algoritmo sendo e e d respectivamente são 5 e 2117, onde, sendo o valor do expoente público e o qual em conjunto com o módulo n compõe a chave pública, resultando em (3683, 5). Enquanto o expoente privado, responsável por compor a chave privada em conjunto com o módulo, torna o valor da chave privada em (3683, 2117). Ao inserir os valores de e e d em seus campos, o resultado deve ser todos os valores anteriormente encontrados, e segundo o manual da calculadora, o algoritmo e os valores estão totalmente consistentes e corretos caso ao fim dos resultados, as seguintes mensagens sejam exibidas: "e*d mod r = 1", "e and r are relatively prime" e "d and r are relatively prime", como visto na figura 19.

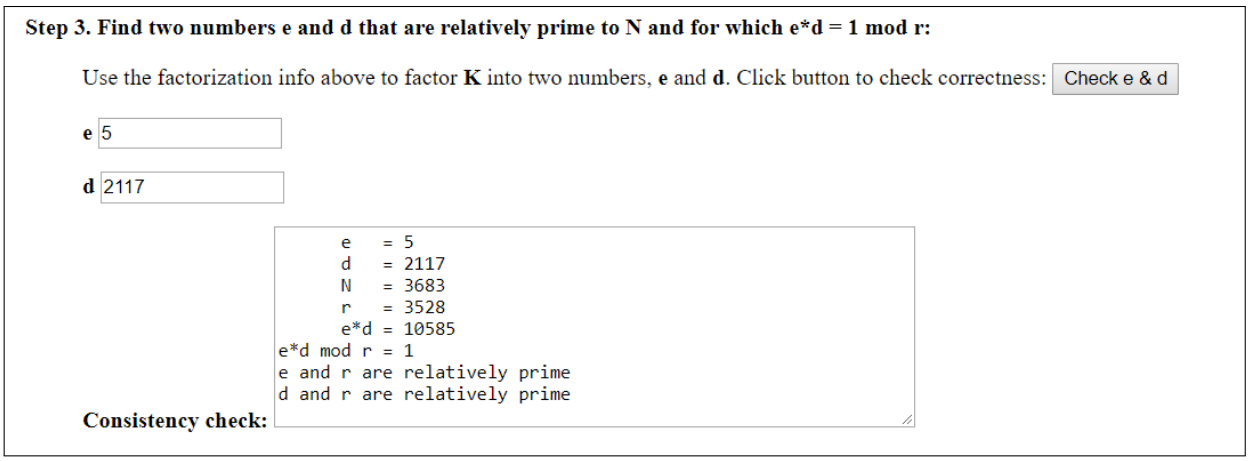

Figura 19 – Passo de validação dos expoentes público e privado.

Para validar os algoritmos de codificação e decodificação, utilizou-se a última etapa da ferramenta RSA Calculator. A qual é possível inserir um valor numérico e vê-lo codificado e decodificado a partir dos valores encontrados acima pela calculadora. Para isto, utilizou as chaves geradas pelo algoritmo 19, sendo elas a chave publica (3683, 5), e a chave privada (3683, 2117), e o valor de mensagem "f", valor este correspondente à 102 na tabela ASCII, gerando o seguinte resultado presente na figura 20.

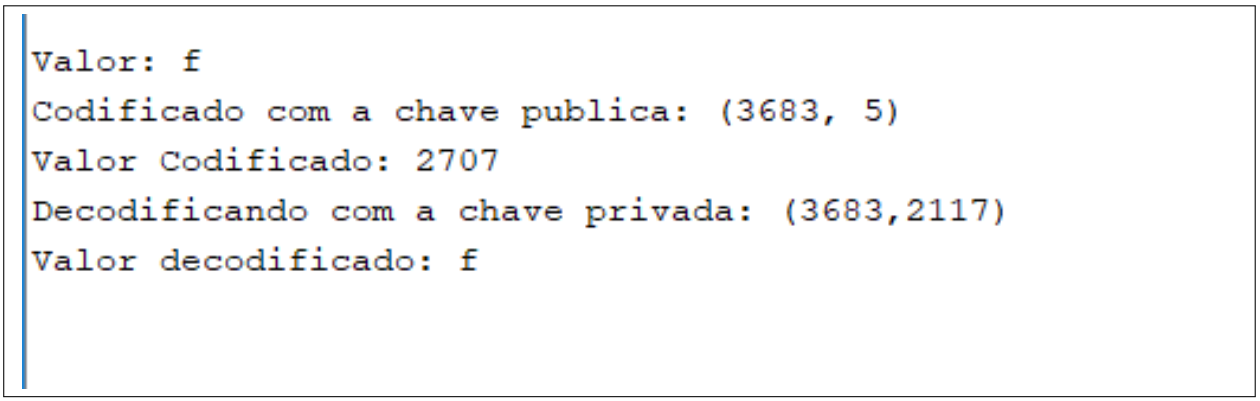

Figura 20 – Valor codificado e decodificado gerado pelo algoritmo.

Apenas com os mesmos valores de p e q, a calculadora RSA foi capaz de gerar os mesmos valores, desde o módulo aos expoentes público e privado, e também as chaves pública e privada. Desta maneira, estes mesmos valores foram utilizados para também validar os métodos e algoritmos implementados de codificação e decodificação, o qual resultaram nos valores vistos na Figura 20, ao utilizar a calculadora para codificar e decodificar o valor 102 (f), obteve-se o mesmo resultado anteriormente alcançado pelo algoritmo proposto e implementado, como pode ser conferido na figura 21.

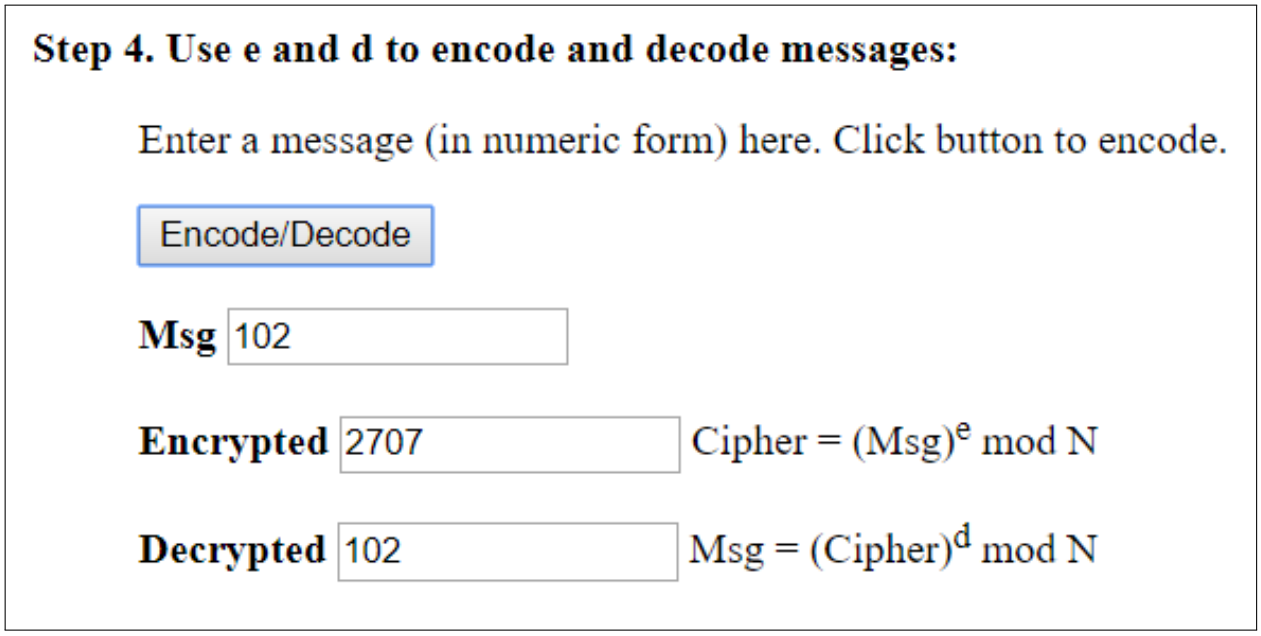

Figura 21 – Valor codificado e decodificado gerado pela ferramenta de validação.

6 CONCLUSÃO

Durante o presente trabalho, foi apresentada a tecnologia da Internet das Coisas, assim como seu papel na sociedade e na evolução da tecnologia também foram mostrados. A capacidade que esta tecnologia tem de revolucionar é muito grande, devido ao fator de poder utiliza-la para gerar novas soluções a partir de ”coisas” as quais já estamos acostumados em nosso dia a dia. Todos estes fatores, em conjunto com a flexibilidade e preço ao se utilizar dispositivos da tecnologia IoT, tornam esse paradigma muito poderoso. Também foram apresentadas algumas dificuldades, Limitações, restrições e problemas desta tecnologia.

Quando fala-se em IoT, frequentemente depara-se com suas limitações, sendo elas: seu tamanho ou seus recursos limitados (baixa capacidade de memória, pouco poder de processamento, arquitetura). Todos esses fatores quando se trata de recursos, geram dificuldades e problemas para utilização da tecnologia e de suas plataformas. Uma das principais áreas que sofrem com isso é a segurança e criptografia, pois nesse caso costuma-se utilizar muitos recursos computacionais, quantidades não pensadas para pequenos dispositivos de IoT.

Entre os principais problemas que IoT enfrenta, a segurança é um dos maiores devido às limitações dos dispositivos que integram a Internet das Coisas. Em razão destes dispositivos não serem aptos a executar um mecanismo de segurança completo tradicional já existente em outras plataformas como computadores em razão ao consumo excessivo de recursos para um dispositivo desse porte, o objetivo deste projeto foi implementar o sistema criptográfico RSA na plataforma Arduino e fazer a análise de utilização do mesmo.

O desenvolvimento do projeto acarretou diversas dificuldades de implementação, fazendo com que houvesse a necessidade de adaptações e encontrar soluções de maneira que não afetasse o algoritmo base, nem a forma como a criptografia funciona através de suas características envolvendo números primos.

Os resultados atingidos indicam que a plataforma Arduino possui diversas limitações, as quais resultaram em muitos problemas durante a implementação do RSA na plataforma, com isso, em diversos momentos houve a necessidade de fazer adaptações e mudanças para se encaixar devidamente.

Muitos problemas foram causados em razão ao tamanho da memória do dispositivo e também pela limitação de bits que a arquitetura AVR de 8 bits oferece, desta maneira, o limite para trabalhar com números grandes foi de 64 bits de forma nativa utilizando tipos de variáveis nativos da linguagem C, e quando se trata de outras bibliotecas, o limite de uso sobe para 256 bits, porém, isto resulta com mais facilidade em problemas de

memória, devido à grande fragmentação da memória e ao grande uso que essas bibliotecas necessitam para armazenar e fazer cálculos com números tão grandes e complexos.

A Fragmentação é um grande problema, pois devido à pequena quantidade de recursos computacionais presentes em Arduino, não há gerenciamento de memória para corrigir isso, a unidade central de gerenciamento de memória não está presente.

Uma das maiores dificuldades dessa pesquisa foi a validação dos resultados obtidos. Isso se dá ao fato de que ao implementar um sistema criptográfico, é necessário um tempo maior para aplicação e obtenção de resultados, principalmente em uma plataforma Arduino com as limitações de IoT. Desta maneira, utilizou-se uma ferramenta [52] responsável por comprovar e provar a consistência do algoritmo implementado. A ferramenta utilizada consiste em uma calculadora que faz a verificação de todos os dados e valores para que se obtenha o par de chaves, e também utilizando destes mesmo valores e chaves, verifica a integridade do algoritmo de codificação e decodificação.

Como trabalhos futuros, propõe-se a melhoria da capacidade do tamanho da chave na plataforma Arduino, valores acima de 64 bits, assim como a aplicação do algoritmo de decodificação com chaves maiores que 16 bits de maneira que não haja problemas de memória ou limitação de bits devido à plataforma utilizar microcontrolador AVR de 8 bits. Assim como melhorar o uso de dados para o cálculo de decodificação devido à seu expoente de ordem elevada.

REFERÊNCIAS

[1] ANDRADE, R. S.; SILVA, F. d. S. Algoritmo de criptografia rsa: análise entre a segurança e velocidade. *Eventos Pedagógicos*, v. 3, n. 3, p. 438–457, 2012.

[2] ASHTON, K. That "internet of things" thing. *RFID Journal*, 2009.

[3] AL., L. Y. et. *The Internet of Things: From RFID to the Next-Generation Pervasive Networked Systems*. New York, NY: Auerbach Publications, 2008.

[4] ELKHODR, M.; SHAHRESTANI, S.; CHEUNG, H. The internet of things: Vision amp; challenges. In: *IEEE 2013 Tencon - Spring*. [S.l.: s.n.], 2013. p. 218–222.

[5] KRANENBURG, R. V.; BASSI, A. Iot challenges. *Communications in Mobile Computing*, v. 1, n. 1, p. 9, 2012.

[6] O'BRIEN, H. M. The internet of things. *Internet Law*, v. 19, n. 12, p. 1–20, 2016.

[7] ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. p. 2787–2805, 10 2010.

[8] WORTMANN, F.; FLüCHTER, K. Internet of things: Technology and value added. Gabler, v. 57, n. 3, p. 221 – 224, 2015–06. ISSN 0937-6429. Received 29 January 2015, Accepted 2 March 2015, Published online 27 March 2015.

[9] GARCIA-MORCHON, O. et al. Securing the ip-based internet of things with hip and dtls. In: *Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks*. New York, NY, USA: ACM, 2013. (WiSec '13), p. 119–124. ISBN 978-1-4503-1998-0. Disponível em: <http://doi.acm.org/10.1145/2462096.2462117>.

[10] HOSSAIN, M.; HASAN, R.; SKJELLUM, A. Securing the internet of things: A meta-study of challenges, approaches, and open problems. In: *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*. [S.l.: s.n.], 2017. p. 220–225.

[11] Zhou, W.; Zhang, Y.; Liu, P. The Effect of IoT New Features on Security and Privacy: New Threats, Existing Solutions, and Challenges Yet to Be Solved. *ArXiv e-prints*, fev. 2018.

[12] GEEKPWN. *IoT devices have a large number of low-level loopholes*. 2017. Disponível em: <http://www.sohu.com/a/129188339_198147>. Acesso em: 17.06.2018.

[13] YOO, Y.; HENFRIDSSON, O.; LYYTINEN, K. Research commentary—the new organizing logic of digital innovation: An agenda for information systems research. *Info. Sys. Research*, INFORMS, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, v. 21, n. 4, p. 724–735, dez. 2010. ISSN 1526-5536. Disponível em: <http://dx.doi.org/10.1287/isre.1100.0322>. Acesso em: 15.06.2018.

[14] BALDWIN, C. Y.; CLARK, K. B. *Design Rules: The Power of Modularity Volume 1*. Cambridge, MA, USA: MIT Press, 1999. ISBN 0262024667.

[15] ADOMAVICIUS JESSE C. BOCKSTEDT, A. G. G.; KAUFFMAN, R. J. Making sense of technology trends in the information technology landscape: A design science approach. v. 32, p. 779–809, 2008.

[16] DAVID, B. *The Transparent Society: Will Technology Force Us to Choose Between Privacy and Freedom*. New York, NY: Perseus Press, 1998.

[17] FLEISCH; WEINBERGER; WORTMANN. *Business models for the internet of things*. 2014. Disponível em: <http://www.iot-lab.ch/wp-content/uploads/2014/11/EN_Bosch-Lab-White-Paper-GM-im-IOT-1_3.pdf>. Acesso em: 15.06.2018.

[18] GUBBI JAYAVARDHANA E BUYYA, R. e. M. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, Elsevier, v. 29, n. 7, p. 1645–1660, 2013.

[19] TIBBETS, L.; TANE, J. 2012. Disponível em: <https://platform.ifttt.com/>. Acesso em: 18.06.2018.

[20] PA, Y. M. P. et al. Iotpot: Analysing the rise of iot compromises. In: *Proceedings of the 9th USENIX Conference on Offensive Technologies*. Berkeley, CA, USA: USENIX Association, 2015. (WOOT'15), p. 9–9. Disponível em: <http://dl.acm.org/citation.cfm?id=2831211.2831220>.

[21] COPOS, B. et al. Is anybody home? inferring activity from smart home network traffic. In: *2016 IEEE Security and Privacy Workshops (SPW)*. [S.l.: s.n.], 2016. p. 245–251.

[22] GOVTECH. *FutureStructure: the new framework for communities (Infographic)*. 2015. Disponível em: <http://www.govtech.com/dc/articles/FutureStructure-The-NewFramework-for-Communities.html>. Acesso em: 18.06.2018.

[23] NAWIR, M. et al. Internet of things (iot): Taxonomy of security attacks. In: *2016 3rd International Conference on Electronic Design (ICED)*. [S.l.: s.n.], 2016. p. 321–326.

[24] GRIFFIN, L. *What is a Spoofing attack?* 2015. Disponível em: <https://study.com/academy/lesson/what-is-a-spoofing-attack-definition-types.html>. Acesso em: 19.06.2018.

[25] KUMAR, H.; SARMA, D.; KAR, A. Security threats in wireless sensor networks. *IEEE Aerospace and Electronic Systems Magazine*, v. 23, n. 6, p. 39–45, June 2008. ISSN 0885-8985.

[26] YıLMAZ, M. H.; ARSLAN, H. A survey: Spoofing attacks in physical layer security. In: *2015 IEEE 40th Local Computer Networks Conference Workshops (LCN Workshops)*. [S.l.: s.n.], 2015. p. 812–817.

[27] ALSAADI, E. Internet of things : Features , challenges , and vulnerabilities authors. In: . [S.l.: s.n.], 2015.

[28] BELAPURKAR, A. et al. *Distributed systems security: issues, processes and solutions*. [S.l.]: John Wiley & Sons, 2009.

[29] TRABELSI, Z.; SALEOUS, H. Teaching keylogging and network eavesdropping attacks: Student threat and school liability concerns. In: *2018 IEEE Global Engineering Education Conference (EDUCON)*. [S.l.: s.n.], 2018. p. 437–444.

[30] BHUSHAN, B.; SAHOO, G.; RAI, A. K. Man-in-the-middle attack in wireless and computer networking 2014; a review. In: *2017 3rd International Conference on Advances in Computing,Communication Automation (ICACCA) (Fall)*. [S.l.: s.n.], 2017. p. 1–6.

[31] NAYYAR, A.; PURI, V. A review of arduino board's, lilypad's amp; arduino shields. In: *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. [S.l.: s.n.], 2016. p. 1485–1492.

[32] ARDUINO. 2018. Disponível em: <http://www.arduino.cc>. Acesso em: 13.11.2018.

[33] MARGOLIS, M. *Arduino Cookbook*. [S.l.]: O'Reilly Media, Inc., 2011. ISBN 1449313876, 9781449313876.

[34] MCROBERTS, M. *Beginning Arduino*. 1st. ed. Berkely, CA, USA: Apress, 2010. ISBN 1430232404, 9781430232407.

[35] JOSHI, A.; JOSHI, B. A randomized approach for cryptography. In: *2011 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC)*. [S.l.: s.n.], 2011. p. 293–296.

[36] MANDAL, B. et al. A comparative and analytical study on symmetric key cryptography. In: *2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE)*. [S.l.: s.n.], 2014. p. 131–136.

[37] STALLINGS, W. *Cryptography and Network Security: Principles and Practice*. 5th. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2010. ISBN 0136097049, 9780136097044.

[38] RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, ACM, New York, NY, USA, v. 21, n. 2, p. 120–126, fev. 1978. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/359340.359342>.

[39] WU, Y.; WU, X. Implementation of efficient method of rsa key-pair generation algorithm. In: *2017 IEEE International Symposium on Consumer Electronics (ISCE)*. [S.l.: s.n.], 2017. p. 72–73. ISSN 2159-1423.

[40] LI, Y.; LIU, Q.; LI, T. Design and implementation of an improved rsa algorithm. In: *2010 International Conference on E-Health Networking Digital Ecosystems and Technologies (EDT)*. [S.l.: s.n.], 2010. v. 1, p. 390–393.

[41] QING-HAI, B. et al. Research on design principles of elliptic curve public key cryptography and its implementation. In: *2012 International Conference on Computer Science and Service System*. [S.l.: s.n.], 2012. p. 1224–1227.

[42] EISENBARTH, T. et al. A survey of lightweight-cryptography implementations. *IEEE Design Test of Computers*, v. 24, n. 6, p. 522–533, Nov 2007. ISSN 0740-7475.

[43] LEE, J.-H.; LIM, D.-G. Parallel architecture for high-speed block cipher, hight. *International Journal of Security and Its Applications*, v. 8, p. 59–66, 03 2014.

[44] HASIB, A. A.; HAQUE, A. A. M. M. A comparative study of the performance and security issues of aes and rsa cryptography. In: *2008 Third International Conference on Convergence and Hybrid Information Technology*. [S.l.: s.n.], 2008. v. 2, p. 505–510.

[45] KHAN, N. et al. Performance analysis of security algorithms for iot devices. In: *2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*. [S.l.: s.n.], 2017. p. 130–133. ISSN 2572-7621.

[46] SKLAVOS, N.; ZHANG, X. *Wireless Security and Cryptography: Specifications and Implementations*. 1st. ed. Boca Raton, FL, USA: CRC Press, Inc., 2007. ISBN 084938771X, 9780849387715.

[47] SINGH, S. et al. Advanced lightweight encryption algorithms for iot devices: survey, challenges and solutions. *Journal of Ambient Intelligence and Humanized Computing*, May 2017. ISSN 1868-5145. Disponível em: <https://doi.org/10.1007/s12652-017-0494-4>.

[48] OH, S.; KIM, Y. Development of iot security component for interoperability. In: *2017 13th International Computer Engineering Conference (ICENCO)*. [S.l.: s.n.], 2017. p. 41–44. ISSN 2475-2320.

[49] RSA Algorithm. Disponível em: <https://www.di-mgt.com.au/rsa_alg.html>. Acesso em: 17.11.2018.

[50] PUBLIC-KEY Cryptography Standards (PKCS): RSA Cryptography. Disponível em: <https://tools.ietf.org/html/rfc3447>. Acesso em: 17.11.2018.

[51] GAMMON, N. *BigNumber*.

[52] RSA Calculator. Disponível em: <https://www.cs.drexel.edu/~jpopyack/IntroCS/HW/RSAWorksheet.html>. Acesso em: 20.11.2018.