UNIVERSIDADE DE FORTALEZA

CENTRO DE CIÊNCIAS TECNOLÓGICAS

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

AMANDA CORREIA VERAS

ESTUDO DA PERFORMANCE DE ALGORITMOS DE CRIPTOGRAFIA EM DISPOSITIVOS DE INTERNET DAS COISAS

FORTALEZA – CEARÁ

2018

AMANDA CORREIA VERAS

ESTUDO DA PERFORMANCE DE ALGORITMOS DE CRIPTOGRAFIA EM DISPOSITIVOS DE INTERNET DAS COISAS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Centro de Ciências Tecnológicas da Universidade de Fortaleza, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientador: Bruno Lopes Alcantara Batista

FORTALEZA – CEARÁ

2018

AGRADECIMENTOS

Agradeço, primeiramente, aos meus pais e minha irmã, que sempre acreditaram em mim, me deram todo o suporte e incentivo que eu precisava para seguir em frente, e que fizeram de tudo por mim.

Ao meu orientador, Bruno Lopes, que me ensinou muito, deu a oportunidade de construir este trabalho, apoiou em todas as etapas, forneceu tudo que precisei e sempre me encorajou a fazer um trabalho de qualidade.

Aos meus amigos que me acompanharam nessa jornada, por acreditarem em mim e sempre me ajudarem quando eu precisei, por todos os ensinamentos e incentivo que me deram durante esta etapa. Ao Paulo Roberto, por toda a ajuda que me deu sempre que precisei durante o curso e no desenvolvimento deste trabalho.

Aos professores e coordenadores do curso de Engenharia da Computação, por todo o conhecimento que me passaram e me ajudaram a crescer, e à Universidade de Fortaleza (UNIFOR), pela formação de qualidade.

RESUMO

Este trabalho apresenta o estudo da performance de algoritmos de criptografia em plataformas de Internet das Coisas. O presente trabalho tem como objetivo estudar o funcionamento de algoritmo de criptografia simétrico AES e assimétrico RSA e aplicá-los a ambientes de Internet das Coisas, para que se possa avaliar o impacto na performance dos mesmos. Assim como, aplicar algoritmos de criptografia na camada de rede, na tentativa de garantir a segurança dos dados trocados em um ambiente de Internet das Coisas. Através do estudo, foi verificado que algoritmos assimétricos possuem maior impacto na performance do dispositivo, pois se baseam em cálculos complexos. Com isso, foram escolhidas plataformas utilizadas em prototipagem para mensurar o impacto no processamento. Ao realizar os testes, foi possível provar o impacto na rede e ajudar, através dos dados coletados, a escolher o algoritmo que melhor se adequa ao ambiente de Internet das Coisas, assim como, às necessidades de segurança dos mesmos.

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

ABSTRACT

This work presents a study of the performance of cryptography algorithms in Internet of Things platforms. The work presented here has a goal of study the operation of symmetric cryptography AES and asymmetric RSA and apply them in Internet of Things environments, to evaluate the impact in the performance. As well as apply cryptography algorithms in the network layer, in the attempt to assure the security of the data exchanged in a Internet of Things environment. Through the study, it was verified that asymmetric algorithms has more impact in the performance of the device, because they are based in complex calculus. Thereby, some platforms used in prototyping were chosen to evaluate the impact in performance. After performing the tests, it was possible to prove the impact in the network and help, through the collected data, to choose the algorithm that best suits in the Internet of Things environment, as well as their security needs.

Keywords: Internet of Things. Information Security. Cryptography

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A Internet das coisas (IoT) é uma infraestrutura de rede global, em que todo objeto que está conectado à Internet possui um identificador e pode se comunicar com qualquer outro dispositivo na rede. Esses objetos podem ser computadores, smartphones, carros, câmeras e até mesmo utensílios domésticos, como geladeiras. Devido às inúmeras aplicações, a segurança e a integridade dos dados deve ser uma prioridade. A maior vantagem da Internet das Coisas é de que todos os dispositivos que estão conectados na rede podem ser alcançáveis, seja para se comunicar ou para acessar outros objetos. No entanto, essa facilidade de comunicação também acarreta um problema de segurança, os tornando vulneráveis a muitos ataques.

Estamos vivendo a era da informação, com a Internet presente em todos os aspectos das nossas vidas. Por causa do avanço desenfreado da tecnologia, e a procura por soluções práticas e rápidas, acaba-se não abordando aspectos como segurança dos dados. Por causa do baixo poder computacional de dispositivos da Internet das Coisas, acaba-se não tratando a segurança dos dados como uma prioridade. Em um mundo onde todos possuem acesso à rede, todos os dispositivos conectados à ela estão sujeitos a invasões, seja para tomar controle sobre o dispositivo, seja para roubo de dados. Se os dispositivos de Internet das Coisas (IoT) não estiverem protegidos, eles se tornam alvos fáceis (HUSAMUDDIN; QAYYUM, 2017) .

O número de usuários na Internet já passa de 4 bilhões. No Brasil, 66% da população possui acesso à Internet, 139 milhões de brasileiros (We are Social, 2018). Segundo a Gartner (2018), em 2018, já existem 11 bilhões de dispositivos IoT interconectados. O aumento de dispositivos conectados à internet gera mais brechas de segurança para que os ataques continuem aumentando. Alguns casos de ataque já foram observados, comprometendo serviços e a segurança de indivíduos.

O poder computacional de celulares e dispositivos IoT são bem menores do que de servidores ou notebooks, mas ainda são alvos de mineração de criptomoedas por *malware*. Esse tipo de ameaça pode afetar o desempenho de dispositivos IoT, ao consumir memória e processamento. Para evitar esse tipos de exploração, deve-se atualizar regularmente a versão do *firmware* e mudar as credenciais de acesso padrão, para evitar acesso não autorizado (MERCES, 2018).

Encontrou-se uma falha em marca-passos da marca Abbott, a qual deixava esses dispositivos vulneráveis a ataques, causando risco de perda repentina de bateria. Por volta de 465.000 pacientes foram afetados e foi necessário fazer uma atualização urgente de *firmware*,

para impedir a exploração dessa vulnerabilidade. A atualização pode ser feita de forma remota, mas existe risco de falha de atualização. Essa vulnerabilidade foi causada pela autenticação utilizada nos marca-passos que não era suficientemente complexa e estava usando 24-bit RSA (VAAS, 2018).

O uso de criptografia pode comprometer o funcionamento de dispositivos IoT, pois estes possuem recursos limitados, e o processo de criptografia requer poder de processamento. Criptografias de chave-pública, como RSA, não são recomendadas para dispositivos IoT, por causa do tamanho da chave utilizada. Isto torna inviável, utilizando os algoritmos de criptografia disponíveis no mercado, prover confidencialidade, integridade e disponibilidade simultaneamente (NARU; SAINI; SHARMA, 2017).

1.1 MOTIVAÇÃO

Em 2020, estima-se que mais de 20 bilhões de dispositivos IoT estarão interconectados (GARTNER, 2018). Esses dispositivos são amplamente aplicados em vários setores, como monitoramento do ambiente, automação residencial, industrial, transportes e sistema de saúde (HUSAMUDDIN; QAYYUM, 2017) .

Como esses dispositivos requerem cada vez menos intervenção humana, eles ficam mais expostos a ataques, que podem comprometer o funcionamento correto desses componentes. Afetando a rede e o próprio dispositivo, causando problemas de segurança e privacidade em dispositivos IoT (ISHA; LUHACH, 2016).

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O obejtivo principal deste trabalho é avaliar o impacto do uso de algoritmos de criptografia simétricos e assimétricos tradicionais em plataformas de Internet das Coisas.

1.2.2 Objetivos Específicos

a) Realizar estudo sobre plataformas de prototipagem de Internet das Coisas escolhidas.
b) Realizar estudo sobre bibliotecas de criptografia que possam ser aplicadas às plataformas de Internet das Coisas.

c) Definir casos de teste para avaliar a performance das plataformas escolhidas.
d) Avaliar impacto de algoritmos de criptografia simétricos e assimétricos sobre as plataformas escolhidas.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 INTERNET DAS COISAS

O termo em inglês *Internet of Things* (IoT) foi proferido pela primeira vez em 1999 por Kevin Ashton, pesquisador britânico do Massachusetts Institute of Technology (MIT), sendo considerado assim o primeiro especialista a usar o termo (TALWANA; HUA, 2016), sejam os eletrodomésticos em nossas residências ou equipamentos espalhados pela cidade (ATZORI; IERA; MORABITO, 2010) .

De forma análoga e conceitual estamos cada dia mais próximos da conexão do mundo físico e o digital por meio da *web*, através de um paradigma que preconiza o mundo físico com objetos embarcados com sensores e atuadores, conectados por redes sem fio (*Wi-Fi*) usando a Internet, constituindo assim um ecossistema de rede capaz de realizar processamentos, capturar dados e assim, reagir a partir de uma tomada de decisão.

A IoT é um expoente no campo tecnológico que contribui para o avanço de aplicações para novos campos do conhecimento, a exemplo das cidades inteligentes (*Smart cities*), nos quais o uso de sensoriamento e comunicação busca prover melhores serviços de valor agregado, ou não, para setores administrativos e seus cidadãos (BASSI *et al.*, 2013) .

Os dispositivos IoT podem ser aplicados em diversos segmentos (Figura 1), entre eles, cabe destacar a monitoração de ambiente, em que utilizando sensores, podemos monitorar a qualidade da água e do ar. A partir disso, é possível acompanhar em tempo real a qualidade, e tomar as devidas melhorias necessárias.

Esses dispositivos são aplicados no setor industrial, no qual podem-se utilizar sensores e sistemas de controle capazes de atuar e otimizar processos em tempo real. Com o intuito de acompanhar a produção em uma fábrica e observar quais aspectos devem ser melhorados para se obter uma melhor produção.

Outra área de aplicação seria no setor de transporte, em que os dispositivos IoT são amplamente aplicados em *Self-driving cars* (Carros autônomos). Por meio de tecnologias como sensores, atuadores e GPS, esses carros monitoram o ambiente que está ao seu redor e fazem o papel de motorista. Utilizando essas tecnologias, estes podem se comunicar com outros que estão ao redor e mapear o ambiente, para tomar as melhores decisões possíveis.

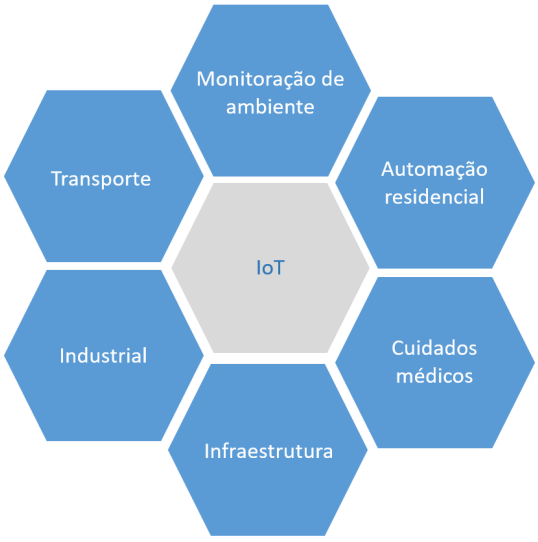
Podemos aplicar também em cuidados médicos, pois através de sensores, o dispositivo é capaz de coletar os sinais vitais do paciente e mandar diretamente para o médico

responsável. Fazendo com que o médico possa analisar esses dados e manter um histórico do paciente. Além de que podemos utilizar também atuadores, como em uma bomba de insulina, que pode verificar os sinais do paciente, e liberar uma certa quantidade de insulina para o mesmo.

Um outro campo de aplicação é na automação residencial, no qual se trata de utilizar um sistema de controle para automatizar as funções de controle do ambiente, capaz também de integrar os dispositivos presentes. Pode controlar além de eletrodomésticos, aspectos do ambiente, como luz, janela, porta, entre outros. Além disso, pode sensoriar o ambiente e enviar informações como gás, energia e água para os prestadores destes serviços.

No setor de rodovias, os dispositivos IoT podem atuar na gestão de infraestrutura, em que seria capaz monitorar e controlar operações de infraestrutura, como estradas, pontes e vias férreas, sem a necessidade de um funcionário localmente. Conseguindo assim, buscar melhorias no serviço e aumentar a sua qualidade (HUSAMUDDIN; QAYYUM, 2017) .

Figura 1 – Aplicações IoT



Fonte – Elaborado pelo autor

Diversos avanços no campo tecnológico nos últimos anos possibilitaram o surgimento da IoT, tais como sensores sem fio, comunicação móvel e computação em nuvem. Entretanto diversos desafios ainda devem ser superados para impulsionar a difusão da tecnologia como a necessidade de um visão mais rígida com relação a segurança dessas interfaces (BASSI *et al.*, 2013) .

Um sistema IoT pode conter diversos sensores interconectados a fim de monitorar um ambiente, e com isso, desempenhar alguma função e mostrar ao usuário a informação coletada.

Para que esse fluxo aconteça, o dado que trafega dentro do sistema IoT passa por 3 estágios: Sensoriamento, Atuação e Gerenciamento.

No Sensoriamento, os sensores existentes neste sistema estão capturando as informações relacionadas ao ambiente. Essa fase é responsável apenas pela coleta de dados. Já na fase de Atuação, as informações coletadas no sensoriamento são analisadas, a fim de disparar alguma função. Por exemplo, se existir um sensor de temperatura, e quando chegar a 50 ° C ele deve acender uma luz em um painel, é nesse momento da atuação que essa ação será disparada.

A etapa de Gerenciamento é responsável por mostrar ao usuário as informações coletadas, e as ações que foram efetuadas, para que o usuário possa acompanhar o desempenho do sistema (KHAN *et al.*, 2012). Podemos verificar o fluxo de dados em um sistema IoT através de sua arquitetura. Segundo Wu, Lu e Ling (2010), A arquitetura IoT é divida em cinco camadas: Percepção, Rede, Middleware, Aplicação e Negócios (Figura 2).

- Percepção: É onde estão presentes os objetos físicos e de sensoriamento, camada responsável pela coleta de dados;
- Rede: Transfere a informação dos objetos físicos para o sistema de processamento;
- Middleware: Tem como finalidade receber os dados da camada de rede, processá-los e tomar decisões baseadas nesses dados;
- Aplicação: Provê gerenciamento de toda a aplicação (Smart health, Smart city, Smart car, etc), baseado nas informações processadas pela camada de Middleware;
- Negócios: Responsável pelo gerenciamento completo do sistema IoT, incluindo aplicação e serviços. Constrói modelos de negócio, como gráficos e fluxogramas. Analisando os resultados, pode-se ajudar a determinar futuras ações e estratégias de negócio.

Figura 2 – Arquitetura IoT

Fonte – Elaborado pelo autor

2.2 CRIPTOGRAFIA

Com o crescimento acelerado da tecnologia, assim como da conectividade entre os dispositivos, aumentaram também as vulnerabilidades e a exposição a ameaças. Se um dispositivo sofre algum tipo de ataque, e estiver conectado à rede, toda a rede estará comprometida. Segundo Husamuddin e Qayyum (2017), os principais requisitos quanto a segurança são: Integridade, Confidencialidade, Autenticidade e Disponibilidade.

- Integridade é responsável por assegurar que os dados enviados sejam os mesmos a serem recebidos. Tem como finalidade, impedir que esses dados sejam alterados de forma indevida, pois pode perder confiabilidade;
- Confidencialidade se detém ao fato de que as informações trocadas entre dispositivos devem manter o sigilo, ou seja, outros usuários ou dispositivos que não fazem parte dessa comunicação, não devem acessá-la;
- A autenticidade garante que as mensagens recebidas são confiáveis, pois permite que se possa confirmar o remetente da mesma;
- Os dados em um dispositivo IoT devem estar sempre disponíveis para que os usuários possam acessá-lo.

Como foi relatado por Jay Radcliffe, um pesquisador na área de cibersegurança, onde se verificou uma falha de segurança em uma bomba de insulina, na qual era controlada por um controle remoto. A falha de segurança previa que um hacker poderia interceptar a mensagem enviada pelo controle remoto e alterar a dosagem de insulina liberada pela bomba, o que pode colocar a vida do paciente em risco. Isso acontecia porque a mensagem enviada não era encriptada, ou seja, a integridade dos dados não foi garantida e nem a autenticidade, pois não se podia confirmar o remetente da mensagem (FINKLE, 2016).

Um outro ataque foi recentemente verificado, que também fere aos principais requisitos da segurança. Recentemente, foi verificada uma falha de segurança em uma boneca, que foi vendida como uma boneca interativa capaz de conversar com a criança. Como a boneca era conectada com a Internet, e essa conexão era vulnerável, ela poderia ser facilmente hackeada, e utilizada como dispositivo para monitoramento das crianças, em que se poderia ouvir tudo o que a boneca estava recebendo de informação. Deviado a conexão vulnerável, o hacker conseguiria acessar as informações armazenadas pela boneca, obtendo facilmente acesso a rede Wi-Fi onde a boneca estivesse conectada (GIBBS, 2015).

Contudo, os dispositivos IoT possuem restrições quanto a memória e processamento,

o que faz com que, muitas vezes, essas recomendações quanto a segurança não sejam abordadas (ISHA; LUHACH, 2016) .

Criptografia faz parte do campo de segurança e é uma técnica utilizada para esconder o conteúdo da mensagem, torná-la incompreensível (PAAR, 2010) . A mensagem a ser enviada é transformada em um texto cifrado, utilizando como entrada o texto claro e a chave de encriptação. Dessa maneira, apenas o remetente e o destinatário da mensagem conseguirão compreendê-la. Os algoritmos de criptografia podem ser, simétrico ou assimétrico.

O algoritmo de criptografía simétrico, utiliza uma única chave para encriptar e decriptar os dados. Apenas o receptor e o emissor da mensagem devem possuir a chave secreta, com ela, eles poderão enviar mensagens cifradas e decriptar as mensagens recebidas. As partes entram em acordo para definir uma única chave, que será a chave da sessão.

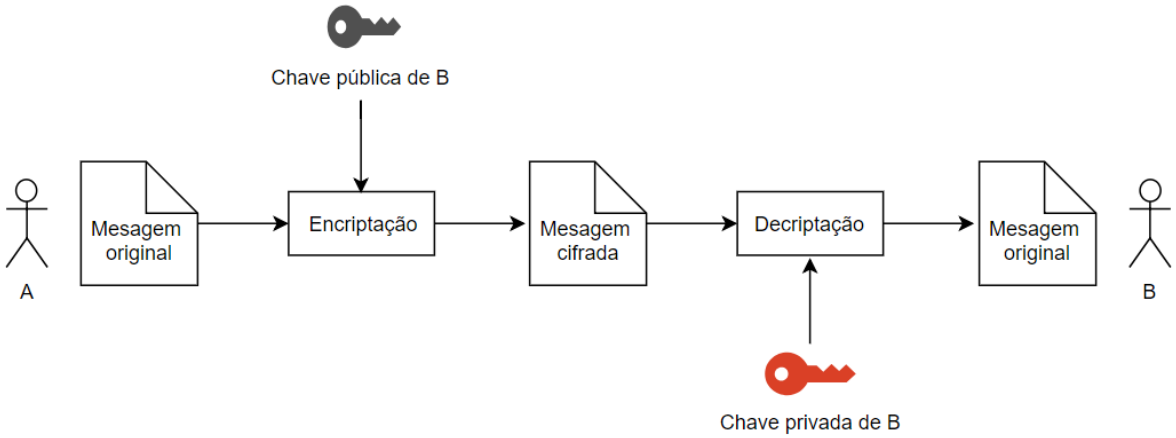
A encriptação simétrica possui dois obstáculos, distribuição de chave e assinatura digital. A encriptação simétrica requer que antes da comunicação entre os dois usuários, eles devem compartilhar a chave secreta, ou que essa chave seja adquirida utilizando um centro de distribuição de chaves. Mas o centro de distribuição, estaria violando um dos princípios da criptografia, o sigilo. Pois outro elemento da rede teria acesso a chave e não apenas os interessados na comunicação.

A criptografia assimétrica, utiliza o conceito de duas chaves complementares, uma chave pública e uma privada. Dado que, é computacionalmente inviável determinar uma chave a partir da outra. No processo de comunicação, cada usuário gera um par de chaves, disponibilizando a chave pública para o outro usuário e mantendo a chave privada sob sigilo.

Para que A envie uma mensagem para B, primeiro A deve encriptar a mensagem utilizando a chave pública que foi gerada por B e lhe foi enviada. Quando B receber a mensagem encriptada, ele poderá utilizar sua chave privada para decriptá-la, como pode ser visto na Figura 3. Fazendo com que, apenas B que possui a chave privada possa ler a mensagem, pois a chave pública só consegue encriptar a mensagem, e não decriptá-la. Com isso, foi possível solucionar o problema de distribuição de chaves.

O método de assinatura digital tem como objetivo utilizar uma assinatura para comprovar o remetente da mensagem. Para que a mensagem seja descriptografada, o remetente terá que utilizar a chave pública disponibilizada pelo emissor. Com isso, ele consegue comprovar o emissor da mensagem, pois a chave privada depende diretamente da chave pública. Utilizando criptografía assimétrica, conseguimos solucionar os dois problemas descritos (STALLINGS, 2015).

Figura 3 – Criptografia assimétrica



Fonte – Elaborado pelo autor

2.2.1 AES

O padrão de criptografia simétrico Advanced Encryption Standard (AES), é uma cifra de bloco que utiliza como entrada um bloco de 128 bits de dados e uma chave que pode variar de tamanho entre 128/192/256 bits, essa chave do AES é gerada de forma pseudo-aleatória (DAEMENA; RIJMEN, 1999).

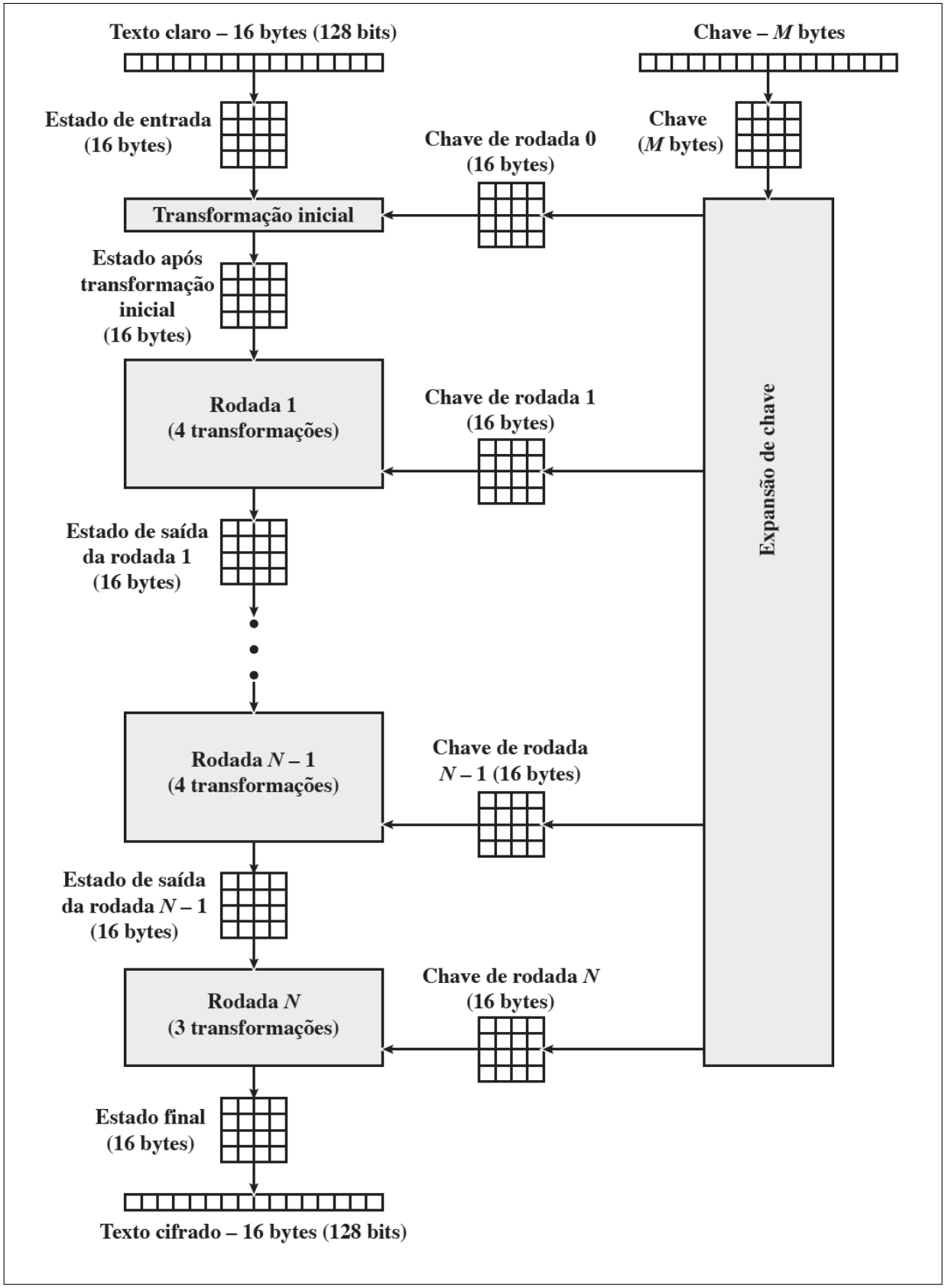
Uma cifra de bloco é um algoritmo que opera em blocos de bits de tamanho fixo, e que encripta todo o texto simples de uma única vez com a mesma chave. Com isso, o padrão AES realiza rodadas, em que a quantidade de rodadas depende do tamanho da chave. Essa prática tem como objetivo prover confusão e difusão.

- A confusão tem como intuito fazer com que a relação entre a chave e o texto cifrado seja a mais complexa possível.
- A difusão é uma operação em que o texto cifrado possui uma grande dependência sobre o texto simples.

No caso de se utilizar a chave de 128 bits, serão necessárias 10 rodadas. Cada rodada possui quatro camadas: Inclusão da chave de rodada, Substituição de bits, Deslocar linhas e Embaralhar colunas. Antes de se iniciar o processo, a chave AES (128-bit) é expandida, a fim de se obter 11 chaves de rodada, formando assim 11 chaves de 128-bit. São geradas 11 chaves, pois uma é utilizada para uma transformação inicial, e as outras 10 são utilizadas uma em cada rodada. A Figura 4 mostra de forma simplificada o processo de encriptação.

Para demonstrar o cálculo utilizado em cada rodada do AES iremos transmitir a

Figura 4 – Processo de encriptação do AES



Fonte – (STALLINGS, 2015)

mensagem “CRIPTOGRAFIA AES”, passando pelas quatro transformações, como mostrado na Figura 4.

Primeiro, devemos transformar a mensagem em hexadecimal (Tabela 1), de acordo com a tabela ASCII.

Tabela 1 – Mensagem

C	R	I	P	T	O	G	R	A	F	I	A		A	E	S
43	52	49	50	54	4F	47	52	41	46	49	41	20	41	45	53

Fonte – Elaborado pelo autor

E colocar os valores dispostos em forma de uma matriz (indicada abaixo), para que os cálculos possam ser realizados.

$$\begin{bmatrix} 43 & 54 & 41 & 20 \\ 52 & 4F & 46 & 41 \\ 49 & 47 & 49 & 45 \\ 50 & 52 & 41 & 53 \end{bmatrix}$$

2.2.1.1 Incluir chave de rodada

Nesse momento, é realizada uma operação XOR[1] com a chave de rodada e o bloco atual bit a bit. Sendo esse o único dos estágios em que a chave é utilizada, e o único que provê segurança. Supondo que a chave da rodada seja a que está sendo mostrada na matriz abaixo, podemos demonstrar como esse cálculo é realizado.

$$\begin{bmatrix} 43 & 54 & 41 & 20 \\ 52 & 4F & 46 & 41 \\ 49 & 47 & 49 & 45 \\ 50 & 52 & 41 & 53 \end{bmatrix} \oplus \begin{bmatrix} A1 & B1 & C1 & D1 \\ A2 & B2 & C2 & D2 \\ A3 & B3 & B3 & D3 \\ A4 & B4 & C4 & D4 \end{bmatrix} = \begin{bmatrix} E2 & E5 & 80 & F1 \\ F0 & FD & 84 & 93 \\ EA & F4 & 8A & 96 \\ F4 & E6 & 85 & 87 \end{bmatrix}$$

2.2.1.2 Substituição de bytes

Nessa etapa, cada elemento (byte) é substituído, utilizando a tabela pré-definida do AES (S-box). Como mostrado na matriz abaixo, temos o exemplo em que os 16 bytes (128 bits)

[1] XOR: Operação sobre dois ou mais valores lógicos, que produz valor ’1’ se, e somente se, apenas um dos valores for ’1’, se a quantidade for ímpar ().

da mensagem são mapeados na forma de uma matriz. Com isso, cada elemento possui um valor x e um valor y, o primeiro valor $a_{1,1} = E2$, onde cada valor representa um byte $(x,y) = (E,2)$. Utilizando a S-box (Figura 5) podemos encontrar o valor de x e o y correspondente, $(E,2) = (98)$.

Figura 5 – S-box de encriptação

		y														
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8ª
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Fonte – (STALLINGS, 2015)

$$\begin{bmatrix} E2 & E5 & 80 & F1 \\ F0 & FD & 84 & 93 \\ EA & F4 & 8A & 96 \\ F4 & E6 & 85 & 87 \end{bmatrix} \rightarrow \begin{bmatrix} 98 & D9 & CD & A1 \\ 8C & 54 & 5F & DC \\ 87 & BF & 7E & 90 \\ BF & 8E & 97 & 17 \end{bmatrix}$$

2.2.1.3 Deslocar linhas

O processo de Deslocar linhas realiza a permutação dos bytes, onde cada linha é deslocada à esquerda como mostrado na matriz abaixo.

$$\begin{bmatrix} 98 & D9 & CD & A1 \\ 8C & 54 & 5F & DC \\ 87 & BF & 7E & 90 \\ BF & 8E & 97 & 17 \end{bmatrix} \rightarrow \begin{bmatrix} Permanece \\ Desloca\ 1\ coluna \\ Desloca\ 2\ colunas \\ Desloca\ 3\ colunas \end{bmatrix} \rightarrow \begin{bmatrix} 98 & D9 & CD & A1 \\ 54 & 5F & DC & 8C \\ 7E & 90 & 87 & BF \\ 17 & BF & 8E & 97 \end{bmatrix}$$

2.2.1.4 Embaralhar colunas

Na etapa de Embaralhar colunas acontece tomando uma única coluna da matriz, e fazendo uma multiplicação utilizando Corpos Finitos (GF(2^8)) para fazer com que cada byte na entrada afete todos os bytes da saída. Utilizando a matriz que foi obtida no final da etapa anterior (Deslocar Linhas), deve-se multiplicá-la por uma matriz padrão.

$$\begin{bmatrix} 98 & D9 & CD & A1 \\ 54 & 5F & DC & 8C \\ 7E & 90 & 87 & BF \\ 17 & BF & 8E & 97 \end{bmatrix} \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \rightarrow \begin{bmatrix} s_{0,1} & s_{0,2} & s_{0,3} & s_{0,4} \\ s_{1,1} & s_{1,2} & s_{1,3} & s_{1,4} \\ s_{2,1} & s_{2,2} & s_{2,3} & s_{2,4} \\ s_{3,1} & s_{3,2} & s_{3,3} & s_{3,4} \end{bmatrix}$$

Para se obter os valores da matriz, a multiplicação é feita seguindo a regra indicada nas equações abaixo.

$$s_{0,j} = (02 \oplus s_{0,j}) \oplus (03 \oplus : s_{1,j}) \oplus s_{2,j} \oplus s_{3,j}$$
$$s_{1,j} = s_{0,j} \oplus (02 \oplus s_{1,j}) \oplus (03 \oplus s_{2,j}) \oplus s_{3,j}$$
$$s_{2,j} = s_{0,j} \oplus s_{1,j} \oplus (02 \oplus s_{2,j}) \oplus (03 \oplus s_{3,j})$$
$$s_{3,j} = (03 \oplus s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (02 \oplus s_{3,j)}$$

Seguindo as equações acima, podemos obter o valor $s_{0,1}$:

$$s_{0,1} = (02 \oplus 98) \oplus (03 \oplus 54) \oplus 7E \oplus 17$$
$$s_{0,1} = 2B \oplus A8 \oplus 7E \oplus 17$$
$$s_{0,1} = 69$$

Após realizar o cálculo acima para gerar cada um dos 16 valores da matriz, obtém-se a matriz abaixo.

$$\begin{bmatrix} 98 & D9 & CD & A1 \\ 54 & 5F & DC & 8C \\ 7E & 90 & 87 & BF \\ 17 & BF & 8E & 97 \end{bmatrix} \rightarrow \begin{bmatrix} 69 & 67 & F7 & FE \\ A5 & 73 & 72 & EF \\ 09 & 67 & 84 & 4B \\ B7 & 16 & 10 & 8A \end{bmatrix}$$

Comparando a entrada da mensagem com o que foi obtido no final da rodada, podemos verificar como a mensagem foi alterada com apenas uma rodada.

$$\begin{bmatrix} 43 & 54 & 41 & 20 \\ 52 & 4F & 46 & 41 \\ 49 & 47 & 49 & 45 \\ 50 & 52 & 41 & 53 \end{bmatrix} \rightarrow \begin{bmatrix} 69 & 67 & F7 & FE \\ A5 & 73 & 72 & EF \\ 09 & 67 & 84 & 4B \\ B7 & 16 & 10 & 8A \end{bmatrix}$$

A camada de Substituição de bytes é responsável pelo processo de confusão. As camadas de Deslocar linhas e de Embaralhar colunas produzem o processo de difusão. Isso acarreta o efeito avalanche, em que a mudança de um bit no texto simples, pode gerar grandes mudanças no texto cifrado. Com isso, obtém-se resultados do algoritmo muito diferentes, aumentando a quantidade de chaves possíveis a serem pesquisadas por um atacante (PAAR, 2010).

Existem cinco modos de operação do algoritmo AES: ECB, CBC, CFB, OFB e CTR. maneira de aplicar uma cifra de bloco ao texto claro.

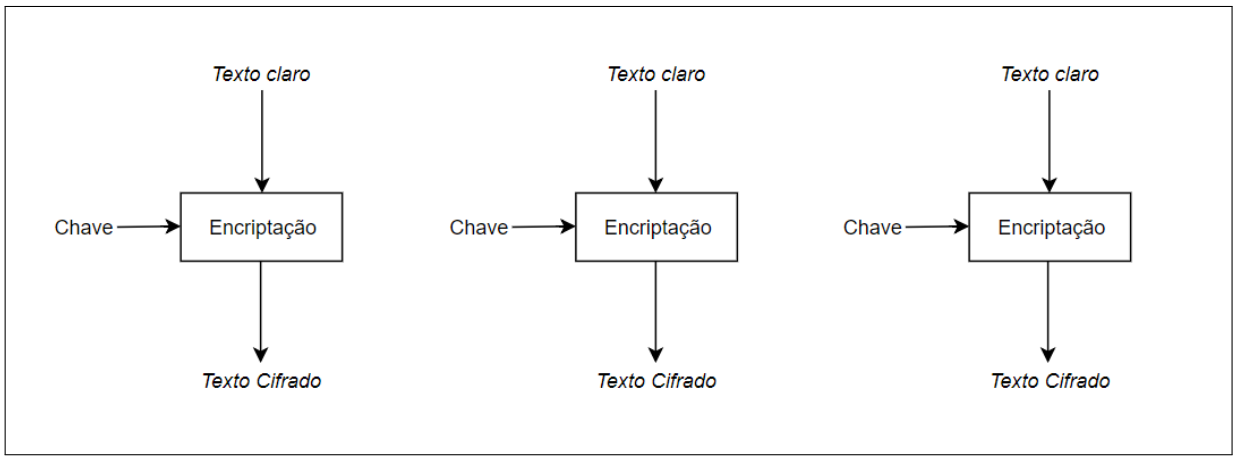
2.2.1.5 Modo Eletronic Codebook

No modo de operação Eletronic Codebook (ECB), a mensagem a ser encriptada é separda em blocos de b bits, de forma que, se a quantidade de bits da mensagem não for um múltiplo de b, o bloco será completado com zeros. O modo de operação ECB é o mais simples, pois os blocos são encriptados separadamente, e o os blocos também são decriptados separadamente, como pode ser visto na Figura 6. Esse modo possui a vantagem de que se algum bloco for perdido na transmissão, ainda será possível descriptografar os blocos, pois eles são independentes. Além disso, a encriptação e decriptação pode ser feita de forma paralela, reduzindo tempo de processamento. Apesar das vantages, esse modo se torna mais susceptível a ataques, pois os mesmos blocos, ao serem encriptados, geram os mesmo blocos cifrados (Se a chave for a mesma) (STALLINGS, 2015).

2.2.1.6 Modo Cipher Block Chaining

No modo CBC, Figura 7, a entrada do algoritmo de encriptação é um XOR do bloco de texto claro atual e do bloco de texto cifrado anterior, a mesma chave é utilizada para cada bloco. Com isso, o bloco atual depende não apenas do texto claro, mas também de todos os blocos anteriores. Além disso, é utilizado um vetor de inicialização (VI) na encriptação do primeiro bloco, como pode ser visto na Figura 6. Aumentando assim a proteção, pois além da

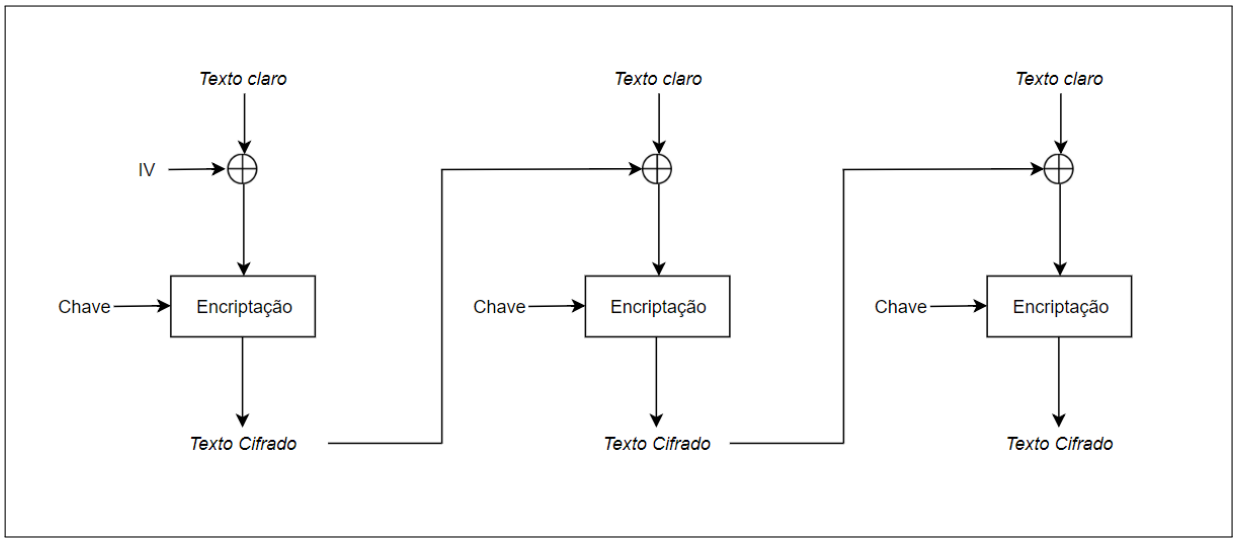
Figura 6 – Modo ECB



Fonte – Elaborado pelo autor

chave de criptografia, também se utiliza o VI, gerando também uma maior difusão, devido a interdependência dos blocos (STALLINGS, 2015).

Figura 7 – Modo CBC



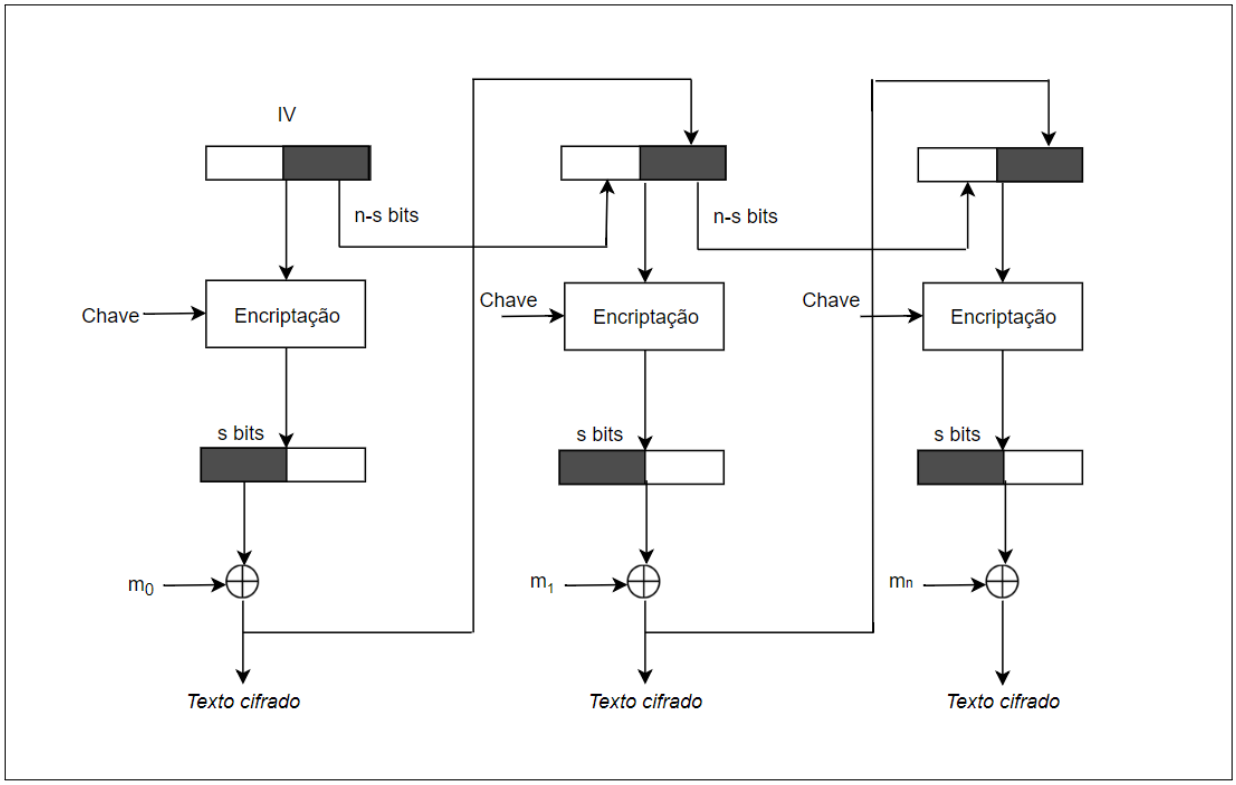
Fonte – Elaborado pelo autor

2.2.1.7 Modo Cipher Feedback

O modo de operação Cipher Feedback (CFB), Figura 8, funciona como uma cifra de fluxo, em que em vez da mensagem ser criptografada em blocos de bits, ela é feita em grupos de 128 bits. Na primeira parte, o vetor de inicialização (VI) é encriptado utilizando a chave de criptografia, depois, parte do texto cifrado (s bits) realizar um XOR com um bloco da mensagem. Na segunda parte, o resto do vetor de inicialização (n-s bits) é adicionado ao texto cifrado

resultante da etapa anterior, e será encriptado, repetindo o processo até que toda a mensagem seja encriptada (STALLINGS, 2015).

Figura 8 – Modo CFB



Fonte – Elaborado pelo autor

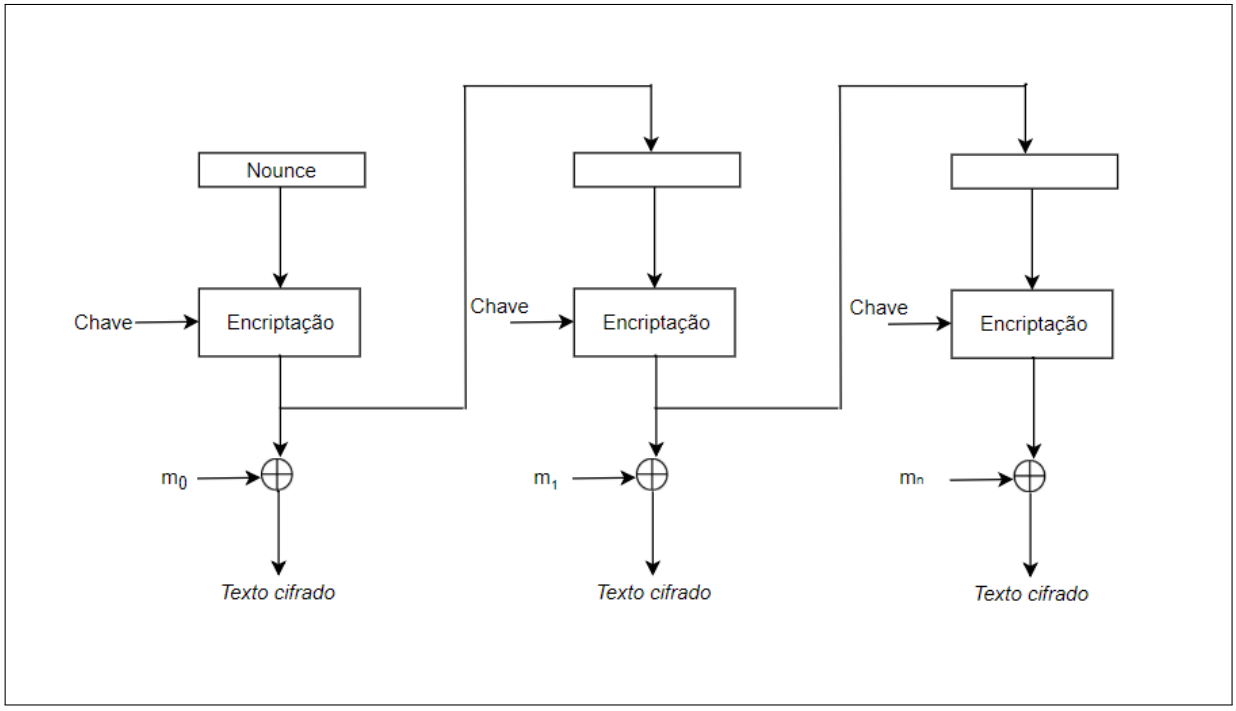
2.2.1.8 Modo Output Feedback

O modo Output Feedback (OFB), Figura 9, também possui um vetor de inicialização, mas aqui é chamou de Nounce, e deve ser gerado a cada nova encriptação. Na primeira fase, assim como no modo CFB, o Nounce é encriptado utilizando a chave de criptografia. Após esse processo, é realizado um XOR com o que foi gerado da encriptação com um bloco da mensagem, gerando o texto cifrado. A entrada do próxima fase será o resultado da encriptação da fase anterior (STALLINGS, 2015).

2.2.1.9 Modo Counter

O modo Counter (CTR), Figura 10, é largamente utilizado em redes de ATM (asynchronous transfer mode) e IPsec (IP security). Utiliza-se um contador, que possui o mesmo tamanho do bloco de texto claro, o valor do contador deve ser diferente para cada bloco de texto

Figura 9 – Modo OFB



Fonte – Elaborado pelo autor

claro que é encriptado. O valor do contador é gerado, e para cada bloco, o valor é incrementado em 1. Para produzir o texto cifrado, o resultado da encriptação do contador passa por um XOR com o bloco da mensagem. Possui a vantagem de que pode ser feita em paralelo em múltiplos blocos de texto, e é mais simples de implementar do que os modos anteriores (STALLINGS, 2015).

2.2.2 RSA

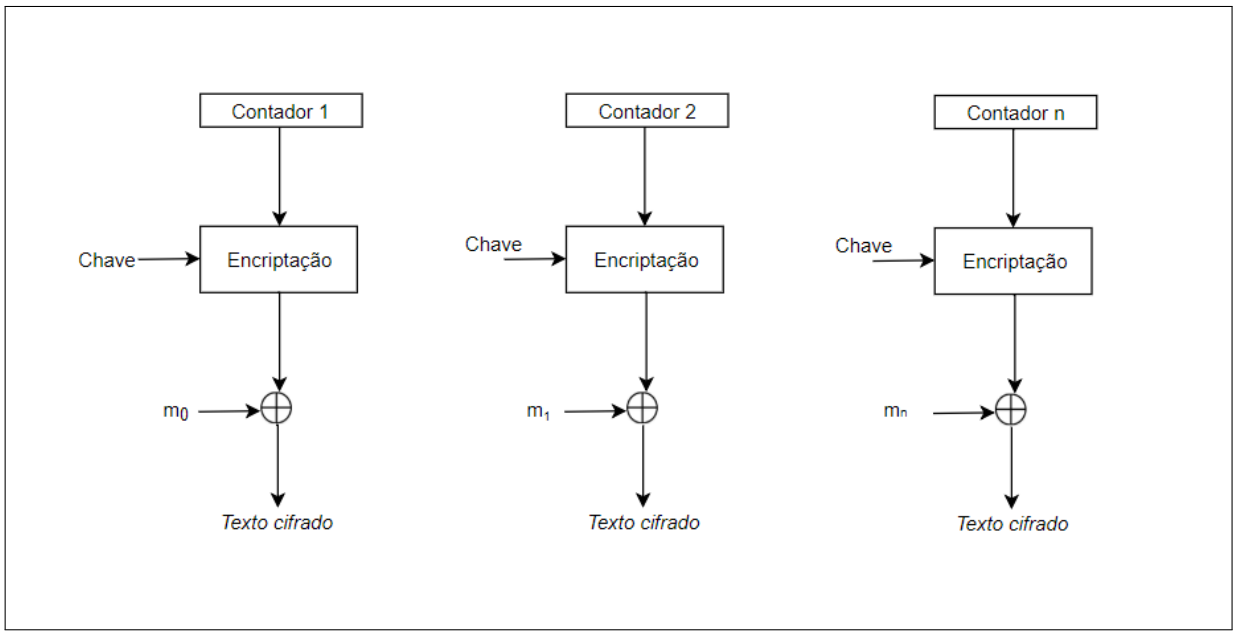
RSA é um algoritmo de criptografia assimétrico, desenvolvido por Ron Rivest, Adi Shamir e Len Adleman em 1977. Este algoritmo utiliza números primos para determinar as chaves públicas e privadas (RIVEST; SHAMIR; ADLEMAN, 1997).

Os cálculos do algortimo RSA são baseados em duas proposições matemáticas, Teorema fundamental da aritmética e a Função totiente de Euler.

2.2.2.1 Teorema Fundamental da Aritmética

O Teorema fundamental da aritmética afirma que: "Todo natural maior que 1 ou é primo ou pode ser escrito de maneira única, a menos da ordem dos fatores, como um produto de primos".

Figura 10 – Modo CTR



Fonte – Elaborado pelo autor

Para encontrarmos esses fatores em que o conceito do teorema se refere, devemos fatorar o número (n), de forma a obter a fatoração canônica de n em primos. Podemos obter utilizando a regra abaixo:

$$n = p_1^{x_1} p_2^{x_2} \ldots p_k^{k_1}$$

Com isso, pode-se ver que todo número possui uma forma única de ser escrito, baseado nos números primos (STEIN, 2017).

2.2.2.2 Função Totiente de Euler

A função totiente de Euler ($\Phi(n)$), é definida como o número de inteiros positivos que são menores do que ou iguais a n, além de que, esses inteiros não compartilham fator comum com n.

Para encontra o totiente, deve-se realizar o MDC (Máximo divisor comum) com todos os valores menores que n, utilizando como exemplo $n = 6$:

$$MDC(0,6) = 6$$
$$MDC(1,6) = 1$$
$$MDC(2,6) = 2$$
$$MDC(3,6) = 3$$
$$MDC(4,6) = 2$$

$$MDC(5,6) = 1$$

Existem dois números que são relativamente primos à 6, 5 e 1. Portanto, ($\Phi(n)$)= 2.

O totiente de um número é dado pela Equação abaixo:

$$\Phi(n) = n\prod_{p|n}(1-\frac{1}{p})$$

Utilizando $n = 120$ como exemplo, podemos encontrar o valor do $\Phi(120)$ utilizando como base o Teorema fundamental da aritmética descrito no tópico 2.2.2.1.

$$\text{Se } n = 120, \text{ e } n = p_1^{x_1} p_2^{x_2} ... p_k^{k_1}$$
$$120 = 2^3 \text{ x } 3 \text{ x } 5$$
$$\Phi(120) = (2-1)\ 2^{3-1}\ (3-1)\ 3^{1-1}\ (5-1)\ 5^{1-1}$$
$$\Phi(120) = 32$$

Se não souber o valor dos primos que geram o número n, o cálculo se torna extenso e computacionalmente inviável (PAAR, 2010).

2.2.2.3 Algoritmo RSA

Segundo Rivest, Shamir e Adleman (1997), a representação matemática desse algoritmo está descrita abaixo, seguido de um exemplo prático simples:

1. Selecionar dois números primos quaisquer p e q
2. Calcular n, sendo $n = p \times q$
3. Calcular $\Phi(n) = (p-1)(q-1)$
4. Selecionar e, tal que $MDC(\Phi(n), e) = 1$, e $1 < e < \Phi(n)$
5. Calcular d, sendo $d = e^{-1} \pmod{\Phi(n)}$
6. Determinar a chave pública e a chave privada, sendo:

$$PU = \{e, n\}, \text{ chave pública}$$
$$PR = \{d, n\}, \text{ chave privada}$$

Supondo que dois usuários desejam se comunicar, para simplificar, iremos nomear esses usuários de Alice e Bob. Esses nomes são comumente utilizados por conveniência nas explicações técnicas em criptografia. Onde Alice deseja enviar uma mensagem criptografada para Bob.

Para se iniciar a comunicação, Bob deve determinar sua chave pública e privada seguiremos os passos acima demonstrados.

1. Selecionaremos de forma arbitrária dois números primos

$$p = 47$$
$$q = 59$$

2. Será calculado o valor de n, com base no valor de p e q

$$n = p \times q$$
$$n = 47 \times 59$$
$$n = 2773$$

3. Calcularemos o valor do totiente de n

$$\Phi(n) = (p-1)(q-1)$$
$$\Phi(n) = (47-1)(59-1)$$
$$\Phi(n) = 2668$$

4. Escolheremos o valor de e com base na regra descrita anteriormente

$$e = 17$$

5. Calcularemos o valor de d

$$d = e^{-1} \pmod{\Phi(n)}$$
$$d = 17^{-1} \pmod{2668}$$
$$d = 157$$

6. Com isso, conseguimos determinar a chave pública e a chave privada:

$$PU = \{17, 2773\}, \text{chave pública}$$
$$PR = \{157, 2773\}, \text{chave privada}$$

Após determinar a chave pública e chave privada, Alice poderá transmitir a mensagem “CRIPTOGRAFIA RSA”, no qual as letras serão representadas por: espaço = 00, A = 01, B = 02, C = 03,..., Z = 25. Para encriptar os dados, o RSA separa os dados em blocos contendo duas letras cada:

CRIPTOGRAFIA RSA = 0318 0916 2015 0718 0106 0901 0018 1901

Para isso, Alice deverá utilizar a chave pública de Bob, e a equação mostrada abaixo para encriptar cada bloco da mensagem.

$C = M^e \bmod n$, onde C é o texto cifrado e M é o texto simples

Começaremos com o primeiro bloco 0318:

$$C = 318^{17} \bmod 2773, \text{ sendo}$$
$$C = 578$$

Realizando o mesmo cálculo acima para todos os blocos da mensagem, obteremos o texto cifrado:

0578 2239 0774 2663 0195 2617 1883 1281

Dessa forma, Alice poderá enviar para Bob o texto cifrado, e Bob poderá descriptografar o texto utilizando a sua chave privada e a equação mostrada abaixo:

$M = C^d \bmod n$, onde C é o texto cifrado e M é o texto simples

Começaremos pelo primeiro bloco 0578:

$$M = 578^{157} \bmod 2773, \text{ sendo}$$
$$M = 318$$

Realizando o mesmo cálculo acima para todos os blocos do texto cifrado, Bob conseguirá descriptografar a mensagem de Alice. Concluímos que, apenas quem possui a chave privada conseguirá compreender o conteúdo da mensagem. Pois para determinar a chave privada, o usuário irá precisar saber o valor de n, o qual, para ser obtido, deve ser fatorado para encontrar os valores de p e q.

Consideramos que os números primos utilizados são apenas a fim de demonstração, e usualmente os números escolhidos para determinar a chave (p e q) devem ter no mínimo 2^{512} bits, fazendo com que gere uma chave (n) de no mínimo 2^{1024} bits. O motivo da criptografia RSA ser tão robusta é que os números primos utilizados são tão grandes, que o resultado de sua multiplicação produz um número que torna impossível fatorá-lo utilizando o poder computacional disponível atualmente. Na Figura 11, podemos observar o tempo que levará para fatorar o valor n e encontrar os números primos correspondentes.

Com isso, conseguimos compreender a complexidade da criptografia RSA. Entretanto, a utilização da criptografia assimétrica torna o sistema mais lento, pois é baseado em cálculos extensos (PAAR, 2010).

Figura 11 – Número de operações por tempo

Dígitos	Número de operações	Tempo
50	$1{,}4\ X\ 10^{10}$	3,9 horas
75	$9{,}0\ X\ 10^{12}$	104 dias
100	$2{,}3\ X\ 10^{15}$	74 anos
200	$1{,}2\ X\ 10^{23}$	$3{,}8\ X\ 10^{9}$ anos
300	$1{,}5\ X\ 10^{29}$	$4{,}9\ X\ 10^{15}$ anos
500	$1{,}3\ X\ 10^{39}$	$4{,}2\ X\ 10^{25}$ anos

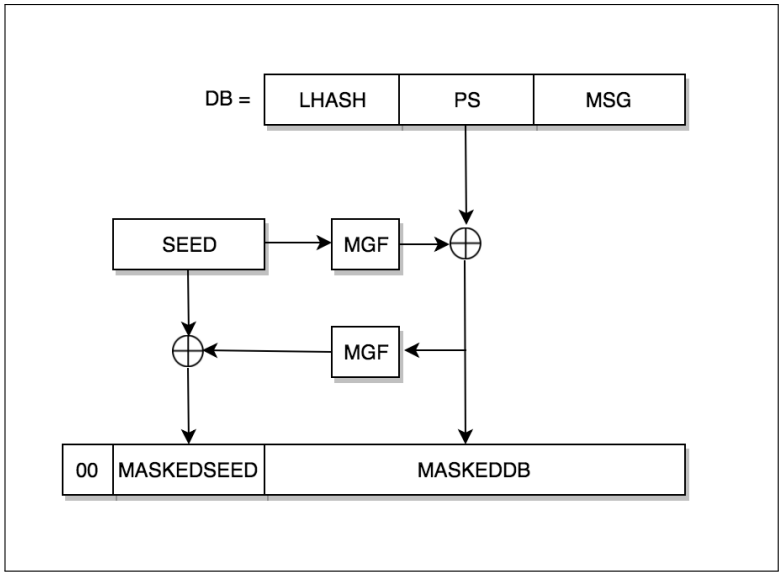
Fonte – (RIVEST; SHAMIR; ADLEMAN, 1997)

2.2.2.3.1 PKCS#1 2.0

PKCS (Public-Key Cryptography Standards) são padrões que definem diretrizes para a implementação de algoritmos de criptografia de chave pública. Neste trabalho iremos utilizar o PKCS#1 2.0, em que implementa o OAEP (Optimal Asymmetric Encryption Padding). OEAP é um esquema de preenchimento, utilizado em conjunto com o algoritmo RSA para prover a expansão da mensagem.

Em que o preechimento é feito de acordo com o esquema descrito na Figura 12. Onde o *Data Block* (DB) contem o LHASH, o PS (que são os bits definidos de preenchimento) e a mensagem (MSG). Depois são realizadas operações de XOR e de MGF (Mask Generating Function) para a obtenção da mensagem final que será encriptada utilizando o algoritmo de criptografia RSA (RSA Laboratories, 2003).

Figura 12 – Esquema OAEP



Fonte – Adaptado de (RSA Laboratories, 2003)

3 METODOLOGIA

3.1 DESCRIÇÃO DO AMBIENTE

Uma aplicação de Internet das Coisas é, essencialmente, composta por: dispositivos, redes de comunicação e sistemas de controle. A combinação desses três componentes formam um sistema IoT. Dispositivos são a tecnologia que está embarcada, composta por chips, sensores e atuadores. Esses dispositivos se comunicam através das redes de comunicação, na qual são responsáveis por fazer essa conexão entre os componentes do sistema. Essas redes podem ser Wi-fi, Bluetooth ou NFC.

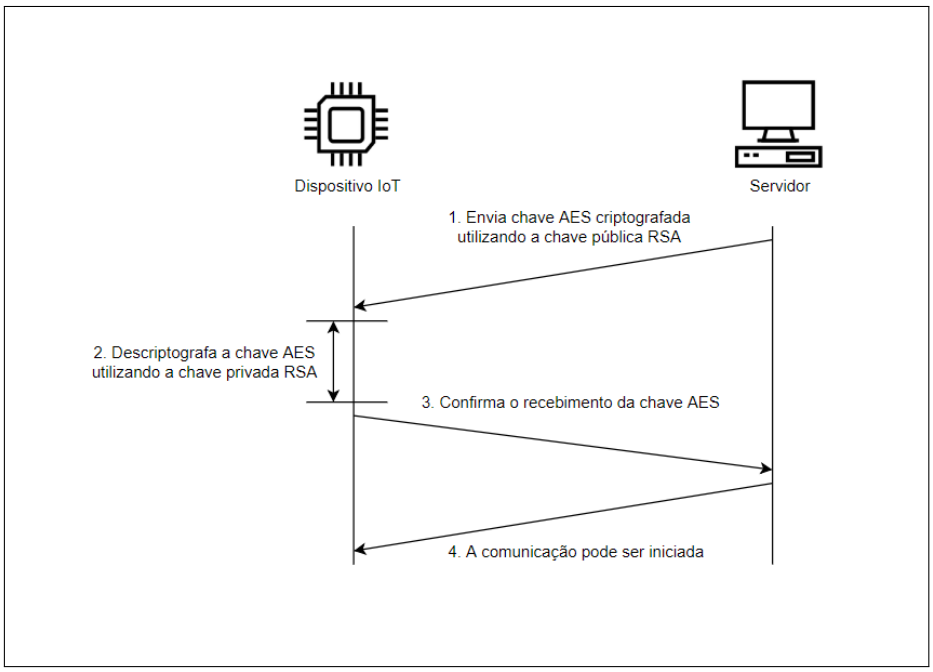
Mas não é suficiente que o dispositivo apenas se conecte à Internet, ou que ele possa se comunicar com outros dispositivos na rede. Ele deve poder enviar esses dados obtidos para que eles possam ser processados, a fim de desempenhar alguma função. Por isso, se faz necessário o uso de sistemas de controle. Esses sistemas podem estar na nuvem, e são utilizados para coletar dados, processar dados ou enviar dados de atuação.

Atuaremos apenas na camada de rede do sistema IoT, garantindo que essa troca de dados entre os componentes do sistema deve ser feita de forma segura, para isso, pode-se utilizar criptografia. Com a criptografia, podemos garantir que os dados enviados e recebidos são íntegros, assegurando os princípios da segurança.

Dado que a utilização de algoritmos simétricos traz problemas como tempo de execução, gerenciamento e troca de chaves e não garantem autenticação. Assim como, algoritmos assimétricos são lentos, devido ao tamanho das chaves utilizadas. Isso traz a necessidade de um algoritmo que possa aproveitar as melhores características de ambos os algoritmos, para garantir economia de energia e processamento, e ainda garantir integridade, autenticidade e confidencialidade (ISHA; LUHACH, 2016).

Para que essa troca de dados seja segura, utilizaremos criptografia híbrida (Figura 13), combiando as melhores características das criptografias AES e RSA. Na tentativa de economizar energia e processamento, serão combinados os dois algoritmos. Utilizaremos o algoritmo simétrico AES para gerar as chaves de criptografia e para encriptar os dados a serem transmitidos. Utilizando juntamente o algoritmo assimétrico RSA para encriptar a chave única do AES. Um dos problemas do algoritmo simétrico é a troca de chaves, que deve ser feita em um ambiente seguro. Utilizaremos criptografia RSA para realizar essa troca de chaves, e garantir que ela não seja descoberta.

Figura 13 – Fluxo de comunicação



Fonte – Elaborado pelo autor

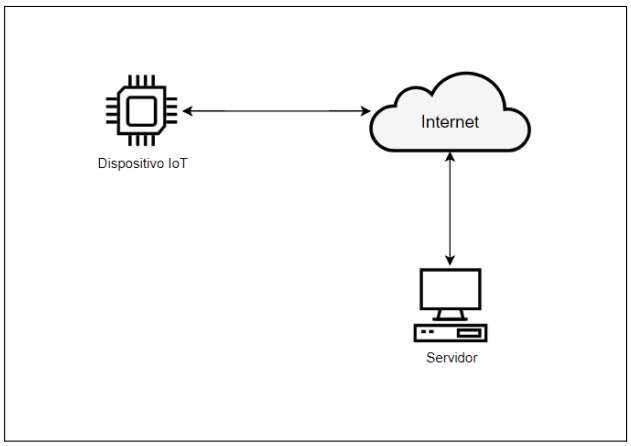
Será feito um estudo investigativo a fim de comparar o desempenho dos hardwares utilizados juntamente com os algoritmos de criptografia (simétrico, assimétrico e híbrido), para que se possa prover um melhor custo benefício quando for escolher o algoritmo de criptografia a ser utilizado no sistema.

O ambiente IoT (Figura 14) que será aplicado neste trabalho, contará com um servidor e um cliente que irão se comunicar para que se possa avaliar o desempenho desses dispositivos no processo de encriptação e decriptação das mensagens. O servidor utilizado será uma máquina MacBook Pro, com processador 3,1 GHz Intel Core i5 e memória 8 GB 2133 MHz LPDDR3. Os clientes utilizados neste estudo serão o Arduino, a Raspberry Pi e a Intel Galileo, que serão descritos na sessão 3.2.

3.2 PLATAFORMAS IOT

Foi feito um estudo sobre as plataformas que foram escolhidas, para que fosse possível avaliar o impacto da criptografia em ambientes de IoT, foram escolhidas três plataformas de IoT diferenes. Os dispositivos escolhidos para fazer o estudo são: Raspberry Pi, Intel Galileo e Arduino. A escolha desses dispositivos se deve ao fato de que são amplamente utilizados para fazer prototipagem, possuem arquitetura, capacidades de processamento e memória diferentes, podendo assim, avaliar o impacto da criptografia simétrica e assimétrica em diferentes ambientes.

Figura 14 – Interface de comunicação



Fonte – Elaborado pelo autor

3.2.1 Arduino

Arduino foi inicialmente desenvolvido em 2004 para auxiliar estudantes que não possuíam conhecimento em eletrônica ou microcontroladores, com isso se tornou a principal ferramenta de prototipagem utilizada no mercado e no meio acadêmico. Atualmente, com o crescimento do mercado de dispositivos IoT, o Arduino se tornou uma ferramenta indispensável para a construção desses dispositivos. O crescimento da plataforma se deve ao fato de ser um hardware livre, onde a sua comunidade está sempre contribuindo e de possuir baixo custo.

Existem variações da plataforma Arduino no mercado, mas a utilizada neste trabalho será o Arduino Uno R3 (Figura 15). Em que é um microcontrolador baseado no processador ATmega328, com uma velocidade de clock de 16MHz. O processador conta com uma SRAM de 2KB, EEPROM de 1KB e uma memória flash de 32KB. Possui 14 pinos digitais de E/S, 6 pinos analógicos, uma conexão USB e uma entrada de energia (5V).

Para programar o Arduino, utiliza-se o software Arduino IDE, na qual possui uma interface simples e intuitiva, foi escrito em Java e é um software open-source. O software torna possível escrever e fazer o upload do código para controlar os componentes da placa (ARDUINO, 2018).

3.2.2 Raspberry Pi

Raspberry Pi é um computador de pequeno porte criado e registrado pela Raspberry Pi Foundation, situada no Reino Unido e sem fins lucrativos que promove e ampara o ensino de programação para crianças. Aliada ao baixo custo e a versatilidade o foco do auxílio ao ensino

Figura 15 – Interface Arduino Uno R3

Fonte – (ARDUINO, 2018)

de crianças logo foi ampliado com o surgimento de diversas aplicações de jogos, multimídia e até pequeno servidores.

Sua estrutura de hardware passa por diversas modificações e a atual denominada Raspberry Pi 3 Model B (Figura 16) foi lançada em Fevereiro de 2016. Conta com um processador quad-core ARMv8 trabalhando com a frequências de 1.2GHz com possibilidade de overclock e arquitetura 64bits e memória RAM de 1GB. O grande diferencial dessa versão além do poder de processamento está na presença de uma interface Wireless 802.11n embarcada e Bluetooth 4.1 nos padrões BLE (Bluetooth Low Energy).

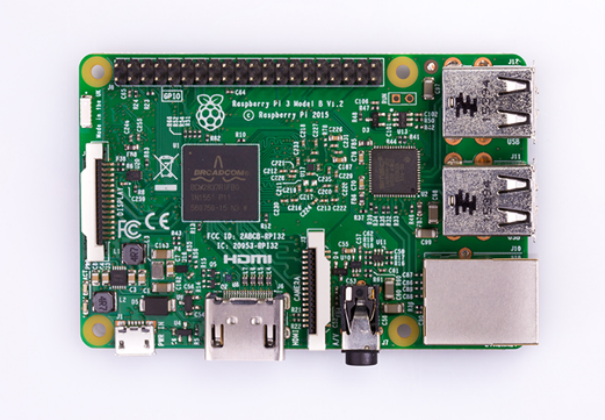
Raspberry Pi funciona como um computador, portanto é necessário ser instalado um Sistema Operacional (SO), como Linux ou Windows. Neste projeto será utilizado o Sistema Operacional Linux. Há também variadas distribuições do SO Linux que podem ser instaladas no Raspberry Pi. Para utilização no projeto foi escolhido o GNU/Linux Debian versão Raspbian devido a sua facilidade de utilização. Pode-se obter uma imagem do SO pronta para utilização no site oficial do Raspberry Pi, a imagem vem estruturada para ser salva em algum padrão de memória flash, Pen drive ou cartão SD (Raspberry Pi Foundation, 2017).

3.2.3 Intel Galileo

Galileo (Figura 17) é a primeira placa microcontroladora baseada no Intel Quark SoC X1000, 400 MHz 32-bit Intel Pentium. É a primeira placa baseada na arquitetura Intel, feita para ter uma pinagem compatível, tanto em hardware como em software com as shields do Arduino Uno R3. A Intel Galileo pode ser programada utilizando o mesmo software do Arduino, ou pode instalar um Sistema Operacional Linux utilizando um microSD de até 32 GB.

A placa Galileo suporta shields arduino com 3.3V e 5V. Conta com 512 KB de

Figura 16 – Interface Raspberry Pi

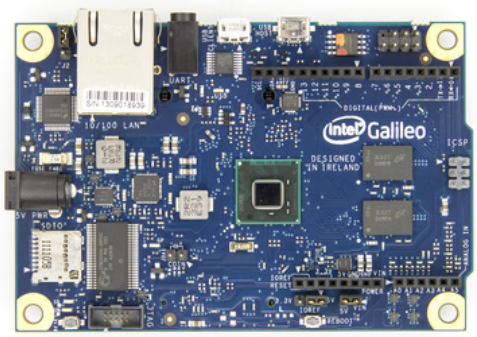


Fonte – (RASPBERRY PI FOUNDATION, 2016)

SRAM embarcada, 256 MB de DRAM, memória flash de 8 MB, conector Ethernet 10/100, PCI Express, porta USB, 14 pinos digitais e 6 analógicos (INTEL, 2018).

Neste projeto será utilizado o Linux, na distribuição Projeto Yocto, na versão X.X. O Projeto Yocto é um projeto open-source, criado em 2010, que tem como finalidade construir sistemas Linux customizados para embarcados, independente da arquitetura de hardware utilizada (Yocto Project, 2018).

Figura 17 – Interface Intel Galileo



Fonte – (INTEL, 2018)

3.3 DESCRIÇÃO DO PROJETO

Para a Raspberry Pi e a Intel Galileo foi escolhida a linguagem Python, por ser uma linguagem simples e que está sendo amplamente utilizada para prototipagem e testes. Como o intuito deste trabalho é fazer uma avaliação do impacto da criptografia e não o desenvolvimento de uma nova biblioteca, foi utilizada a biblioteca já existente PyCrypto. Na qual, é uma

coleção de módulos, em Python, que implementa diversos algoritmos e protocolos criptográficos (PYCRYPTO, 2012). Com essa biblioteca foi possível implementar os módulos de criptogrfia em AES e RSA no ambientes escolhidos.

Para o Arduino foi implementado utilizando a linguagem própria, na qual é baseada em C++, juntamente com a biblioteca de AES que pode ser encontrada no GitHub (SPANOS, 2018). Na qual, é uma biblioteca de criptografia que implementa o algoritmo de criptografia AES efetivamente. Não foi encontrada uma biblioteca de criptografia RSA funcional para o ambiente Arduino, portanto, não será avaliada a criptografia RSA nessa plataforma.

Para a realizar a comunicação entre o dispositivo IoT e o servidor, foi utilizada a interface de software Socket. Socket é uma interface entre a camada de aplicação e a de transporte dentro de um hospedeiro (KUROSE; ROSS, 2014). Para utilizar os Sockets, deve-se fazer uso da arquitetura Cliente-Servidor. Dessa forma, o cliente cria um Socket e se conecta ao servidor, para que possam enviar e receber mensagens. Da mesma forma, o servidor aceita essa conexão do cliente. Neste trabalho, o dispositivos IoT é o cliente.

A fim de avaliar o avaliar a performance de algoritmos de criptografia simétricos e assimétricos, foram definidos casos de testes, que irão testar os seguintes aspectos:

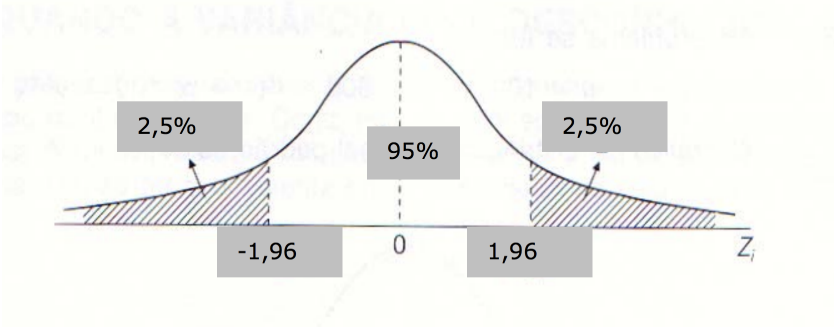
- Avaliar o algoritmos de criptografia simétrico AES nas plataformas escolhidas (Raspberry Pi, Intel Galielo e Arduino), utilizando as chaves de criptografia de 128 bits e 256 bits, e utilizando diferentes tamanhos de mensagens.
- Avaliar o algoritmos de criptografia assimétrico nas plataformas escolhidas (Raspberry Pi e Intel Galielo), utilizando as chaves de criptografia de 1024 bits e 2048 bits, e utilizando diferentes tamanhos de mensagens.
- Avaliar o algoritmos de criptografia híbrido nas plataformas escolhidas (Raspberry Pi e Intel Galielo), utilizando a chave de criptografia AES de 256 bits e RSA de 2048 bits.
- Realizar os testes e avaliar os resultados obtidos a partir de um intervalo de confiança, onde cada parâmetro será calculado 30 vezes, a fim de obter uma maior precisão e confiabilidade nos dados obtidos.

3.4 INTERVALO DE CONFIANÇA

O intervalo de confiança tem como intuito determinar um intervalo a partir de uma amostra da população total para chegar a um estimativa onde se o teste for repetido, o valor obtido estaria dentro deste intervalo baseado no erro calculado.

A estimação do intervalo de confiança consiste na fixação de dois valores, na qual será a probabilidade $(1-\alpha)$ de que o valor a ser obtido esteja entre o intervalo, o valor α determina a porcentagem de erro aceitável. Em que o valor obtido terá 95% de chance de estar dentro no intervalo determinado, aumentando assim, a confiabilidade nos dados obtidos. Na Figura 18, pode-se visualizar o intervalo onde os dados poderão se encontrar.

Figura 18 – Intervalo de Confiança



Fonte – (CORREA, 2003)

Para se obter o intervalo de confiança devemos utilizar a fórmula abaixo:

$$P\left(-Z\frac{\alpha}{2} \leq Z \leq Z\frac{\alpha}{2}\right) = 1-\alpha$$

Onde Z pode ser definido como:

$$Z = \frac{x' - \mu}{\frac{\sigma}{\sqrt[2]{n}}}$$

$x' =$ média dos valores da amostra

$\mu =$ valor obtido

$\sigma =$ desvio padrão

$n =$ quantidade de valores da amostra

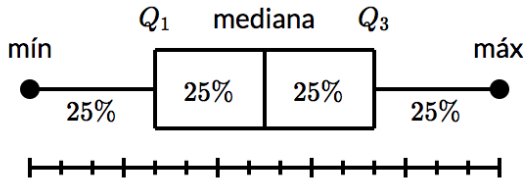
Neste trabalho, iremos utilizar o intervalo de confiança com 5% de erro ($\alpha = 5\%$) e 95% de possibilidade do valor obtido estar dentro do intervalo (CORREA, 2003).

Para calcular o intervalo de confiança deve-se definir uma distribuição normal dos dados. Uma distribuição normal é uma distribuição de probabilidade contínua para uma variável aleatória. Para uma distribuição amostral se aproximar de uma distribuição normal, a amostra coletada deve ser maior ou igual a 30 (MORETTIN, 2010).

3.5 DIAGRAMA DE CAIXA

Utilizaremos o diagrama de caixas para apresentar os dados coletados. O diagrama de caixas (Figura 19) é composto por cinco elementos: mínimo, primeiro quartil, mediana, terceiro quartil, e máximo. Com este diagrama, é possível visualizar os dados coletados da amostra, sumarizando a distribuição dos mesmos (WILLIAMSON; PARKER; KENDRICK, 1989).

Figura 19 – Digrama de caixa



Fonte – (WILLIAMSON; PARKER; KENDRICK, 1989)

Em que, os elementos representam:

- mín: Representa o menor valor da amostra
- $Q1$: Simboliza o primeiro quartil (quartil $= 1/4$ do total)
- mediana: Corresponde ao valor que está no meio da amostra
- $Q3$: Simboliza o terceiro quartil
- máx: Representa o maior valor da amostra

4 RESULTADOS

4.1 COMPARATIVO AES

Na Tabela 12 podemos encontrar os resultados que foram obtidos ao realizar a encriptação AES no Arduino. Foram enviadas mensagens para o servidor de tamanhos variados, utilizando os três tamanhos possíveis de chave (128 bits, 192 bits e 256 bits), para que fosse possível visualizar o desempenho do Arduino com esse algoritmo. Foram encriptadas mensagens de 128 bits até 2048 bits, pois o Arduino é um microcontrolador de 8 bits, conseguindo endereçar variáveis de 0 a 255 bytes (2^8). Da mesma forma, o Arduino recebeu mensagens encriptadas que o servidor enviou, podendo visualizar na Tabela 3 o tempo necessário para descriptografar esses dados. Foram utilizados os mesmos tamanhos de chaves e tamanhos das mensagens.

Tabela 2 – Encriptação AES Arduino

Mensagem/Chave	128 bits	192 bits	256 bits
16 bits	$0,001045s \pm 0,00000077$	$0,001212s \pm 0,0000011$	$0,001427s \pm 0,0000012$
32 bits	$0,001707s \pm 0,0000012$	$0,001996s \pm 0,0000008$	$0,002333s \pm 0,0000010$
64 bits	$0,003020s \pm 0,0000015$	$0,003551s \pm 0,0000012$	$0,004139s \pm 0,0000012$
128 bits	$0,005634s \pm 0,0000016$	$0,006663s \pm 0,0000014$	$0,007747s \pm 0,0000013$
256 bits	$0,010856s \pm 0,0000017$	$0,012887s \pm 0,0000015$	$0,014964s \pm 0,0000016$

Fonte – Elaborado pelo autor

Tabela 3 – Decriptação AES Arduino

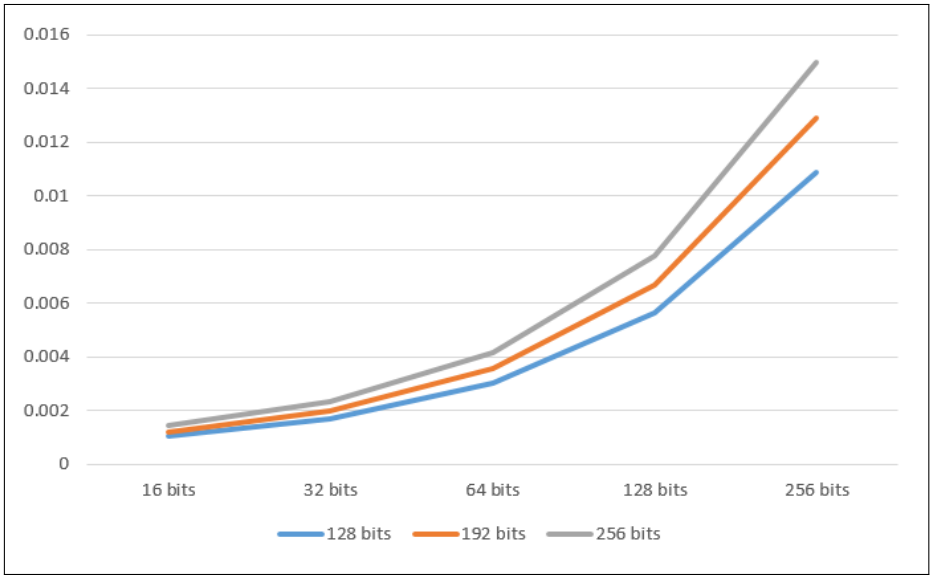
Mensagem/Chave	128 bits	192 bits	256 bits
16 bits	$0,001099s \pm 0,0000006$	$0,001282s \pm 0,0000010$	$0,001504s \pm 0,00000011$
32 bits	$0,001810s \pm 0,0000010$	$0,002122s \pm 0,0000008$	$0,002477s \pm 0,0000014$
64 bits	$0,003220s \pm 0,0000004$	$0,003797s \pm 0,0000011$	$0,004424s \pm 0,0000013$
128 bits	$0,006033s \pm 0,0000010$	$0,007138s \pm 0,0000009$	$0,008286s \pm 0,00000010$
256 bits	$0,011646s \pm 0,0000011$	$0,013812s \pm 0,0000017$	$0,016019s \pm 0,0000012$

Fonte – Elaborado pelo autor

Como pode ser visto nas tabelas acima, o tempo de encriptação é maior que o decriptação, pois no processo de encriptação utilizando o modo de operação CBC, além de realizar operações com a chave de criptografia, é inserido o VI (Vetor de Inicialização), o que torna o processo mais lento.

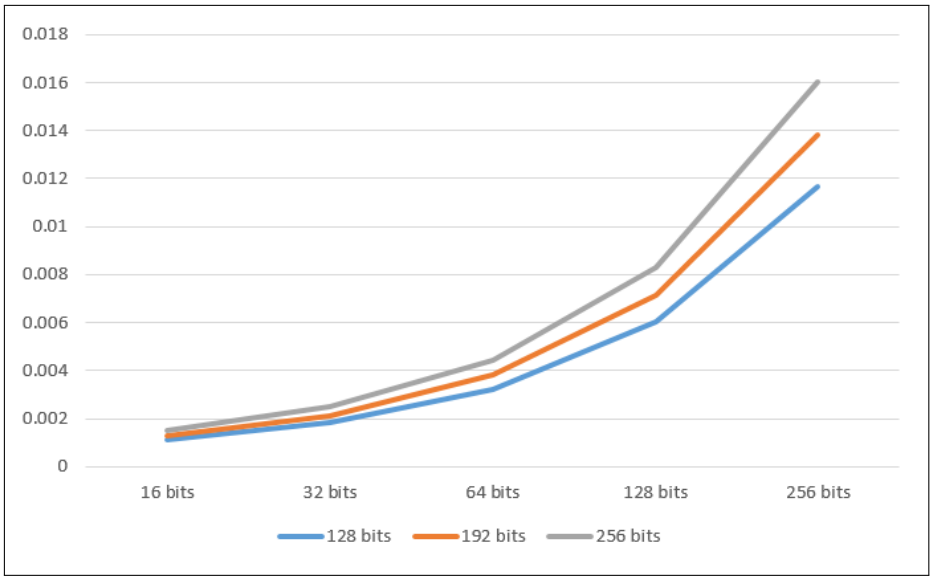
Na Figura 20, podemos visualizar o impacto da chave de criptografia na encriptação de diferentes tamanhos de mensagens enviadas para o servidor. Na Figura 21, podemos visualizar o impacto da chave de criptografia na decriptação de diferentes tamanhos de mensagens enviadas para o servidor.

Figura 20 – Gráfico de encriptação AES para a plataforma Arduino



Fonte – (Elaborado pelo autor, 2018)

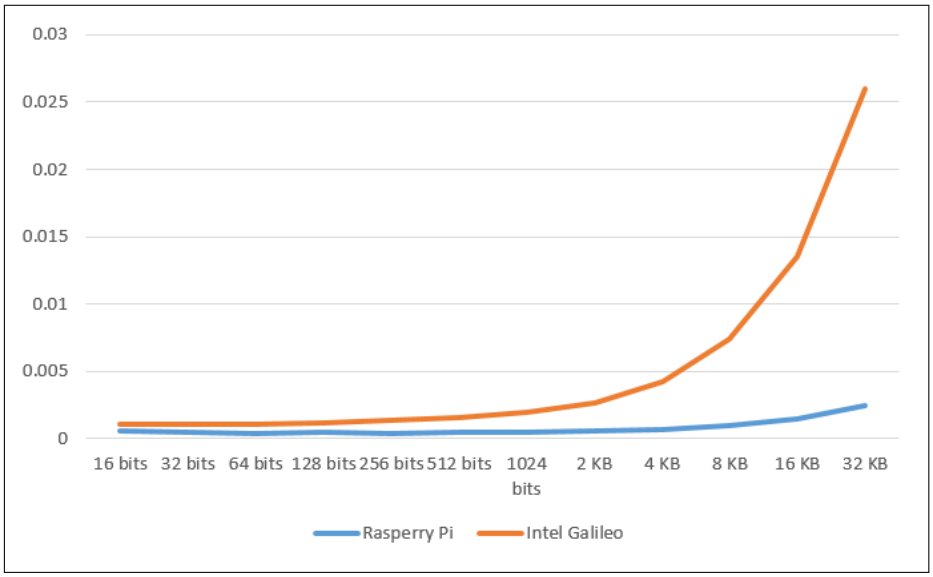
Figura 21 – Gráfico de decriptação AES para a plataforma Arduino



Fonte – (Elaborado pelo autor, 2018)

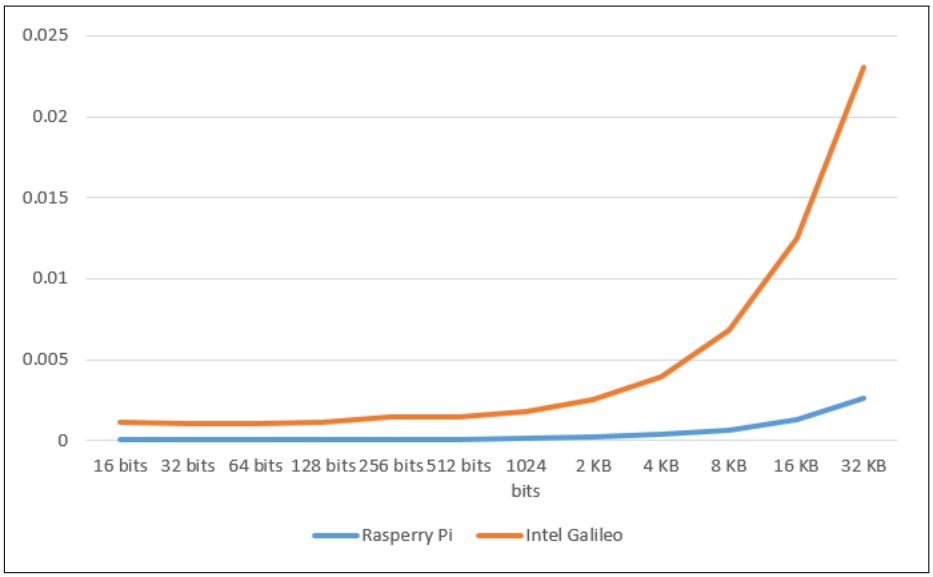
Os resultados que foram obtidos ao realizar a encriptação e a decriptação do AES na Raspberry Pi e na Intel Galileo podem ser encontrados no Apêndice A. Foram enviadas mensagens para o servidor de tamanhos variados, utilizando dois tamanhos de chave (128 bits e 256 bits), mas diferente do Arduino, foi possível utilizar mensagens de maior tamanho, já que a Raspberry Pi opera com 32 bits. A fim de comparação, pode-se visualizar na Figura 22 e na Figura 23, os desempenhos da Raspberry Pi e da Intel Galileo no processo de encriptação e decriptação utilizando a chave de 128 bits.

Figura 22 – Gráfico de encriptação AES para as plataformas Raspberry Pi e Intel Galileo utilizando chave de 128 bits



Fonte – (Elaborado pelo autor, 2018)

Figura 23 – Gráfico de decriptação AES para as plataformas Raspberry Pi e Intel Galileo utilizando chave de 128 bits



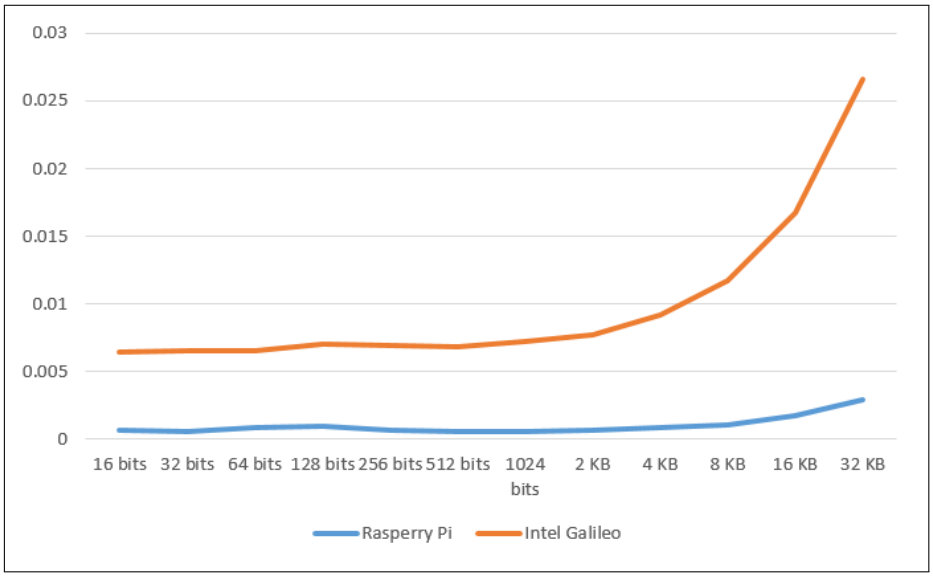
Fonte – (Elaborado pelo autor, 2018)

Na Figura 24 e na Figura 25, pode-se visualizar os desempenhos da Raspberry Pi e da Intel Galileo no processo de encriptação e decriptação utilizando a chave de 256 bits.

A Raspberry Pi possui melhor desempenho, pois possui um processador de 1.2GHz, enquanto a Intel Galielo possui um processador de 400MHz.

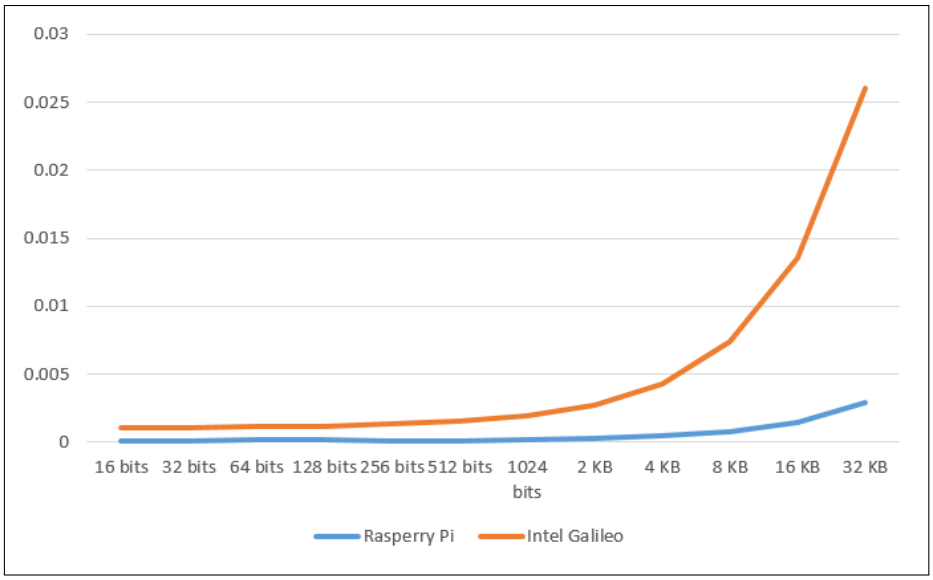
Podemos visualizar na Figura 26 a disposição das amostras coletadas, através do diagrama de caixa, para a plataforma Raspberry Pi, utilizando o algoritmo de criptografia AES

Figura 24 – Gráfico de encriptação AES para as plataformas Raspberry Pi e Intel Galileo utilizando chave de 256 bits



Fonte – (Elaborado pelo autor, 2018)

Figura 25 – Gráfico de decriptação AES para as plataformas Raspberry Pi e Intel Galileo utilizando chave de 256 bits



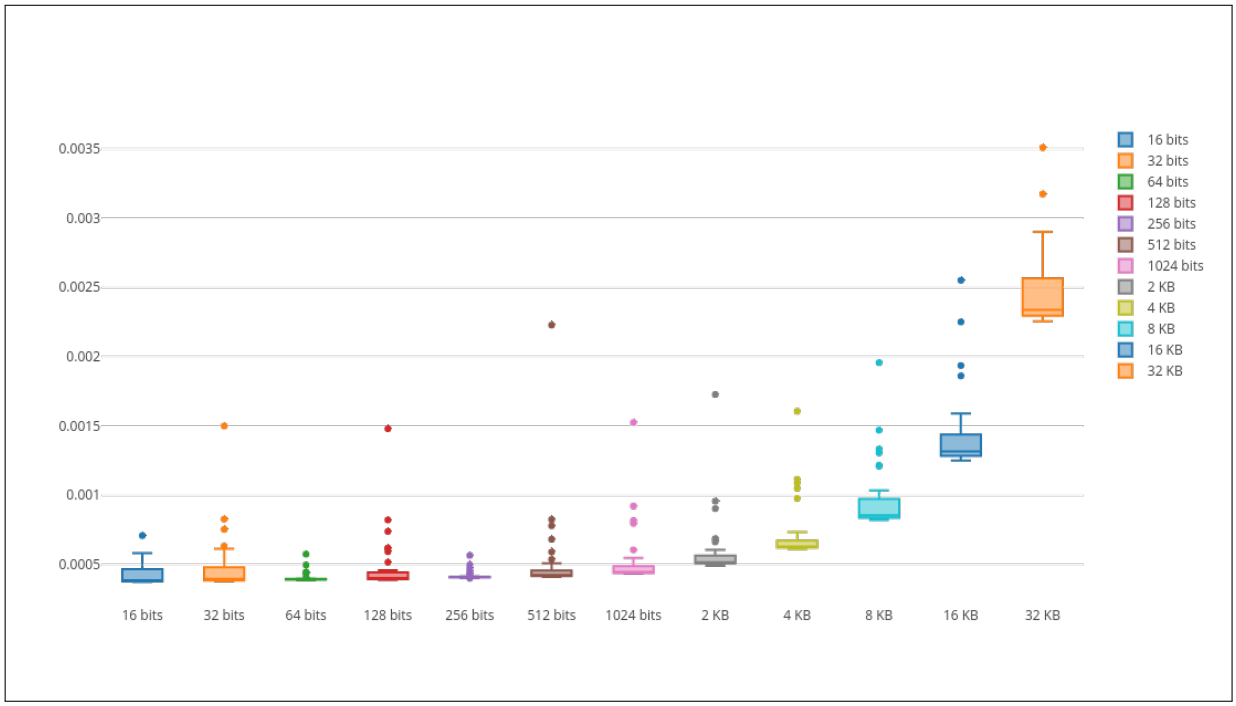
Fonte – (Elaborado pelo autor, 2018)

com uma chave de 128 bits.

Podemos visualizar na Figura 27 a disposição das amostras coletadas, através do diagrama de caixa, para a plataforma Intel Galileo, utilizando o algoritmo de criptografia AES com uma chave de 128 bits.

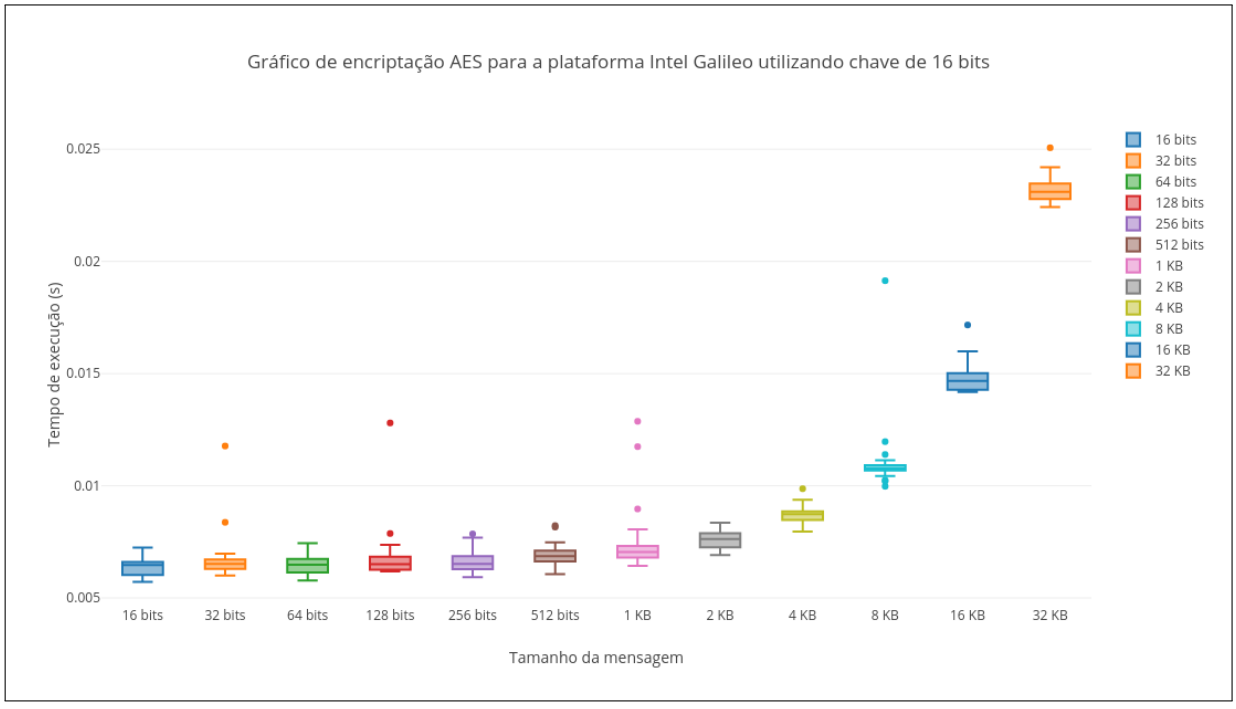
Na Figura 28, podemos visualizar o tempo de encriptação nas três diferentes plataformas, utilizando a chave de 256 bits e mensagens de diferentes tamanhos (128 bits, 256 bits,

Figura 26 – Gráfico de encriptação AES para as plataforma Raspberry Pi utilizando chave de 128 bits



Fonte – (Elaborado pelo autor, 2018)

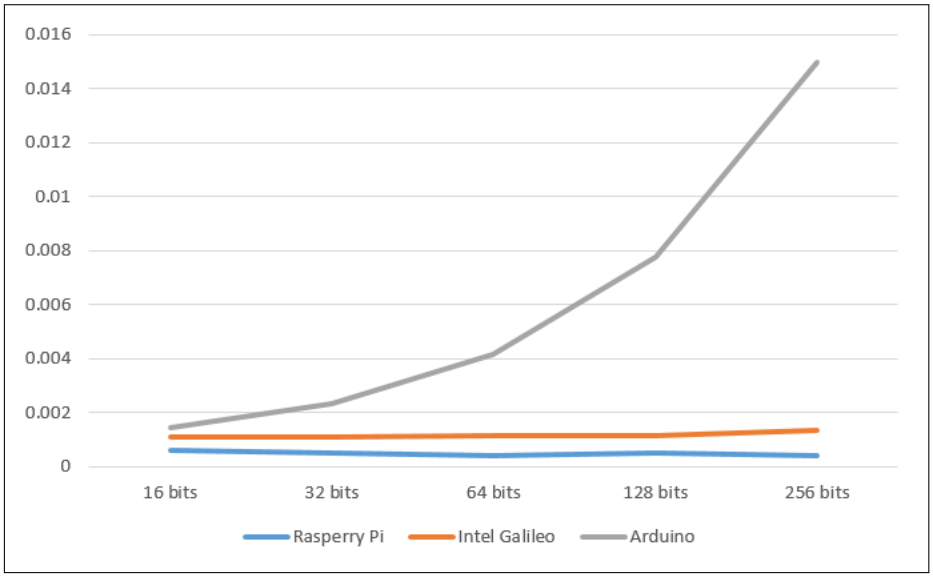
Figura 27 – Gráfico de encriptação AES para as plataforma Intel Galileo utilizando chave de 128 bits



Fonte – (Elaborado pelo autor, 2018)

512 bits, 1024 bits e 2048 bits). A Raspberry Pi continua com melhor performance, pois possui poder de processamento superior à Intel Galileo e ao Arduino (16MHz).

Figura 28 – Gráfico de encriptação AES para as plataformas Arduino, Raspberry Pi e Intel Galileo utilizando chave de 256 bits



Fonte – (Elaborado pelo autor, 2018)

4.2 COMPARATIVO RSA

A fim de realizar o processo de criptografia utilizando o algoritmo assimétrico RSA, foram utilizados três tamanhos de chave 1024 bits, 2048 bits e 4096 bits. O tamanho da chave do algoritmo possui limitações, pois o tamanho da mensagem é limitada pelo tamanho da chave. Neste trabalho foi utilizado o padrão PKCS#1 2.0 do algoritmo RSA, que defininem que o tamanho da chave deva ser igual ao tamanho do arquivo a ser cripografado menos 336 bits, utilizados para realizar preenchimento (padding) definido pelo padrão. Essa inclusão de dados aleatórios tem como finalidade aumentar a segurança do algoritmos RSA contra ataques de força bruta.

Para teste, foram utilizados três tamanhos diferentes de mensagens (688 bits, 1712 bits e 3760 bits) correspondendo ao tamanho da chave. Segue abaixo a fórmula para determinação da chave:

$$MLen \leq KLen - 336bits$$

Com isso, podemos vizualizar na Tabela 4 o tempo necessário para a Raspberry Pi realizar o processo de encriptação, e na Tabela 5 o processo de decriptação.

Tabela 4 – Encriptação RSA Raspberry Pi

Chave/Mensagem	688 bits	1712 bits	3760 bits
1024 bits	$0,006s \pm 0,000415$		
2048 bits		$0,015s \pm 0,000129$	
4096 bits			$0,043s \pm 0,00047$

Fonte – Elaborado pelo autor

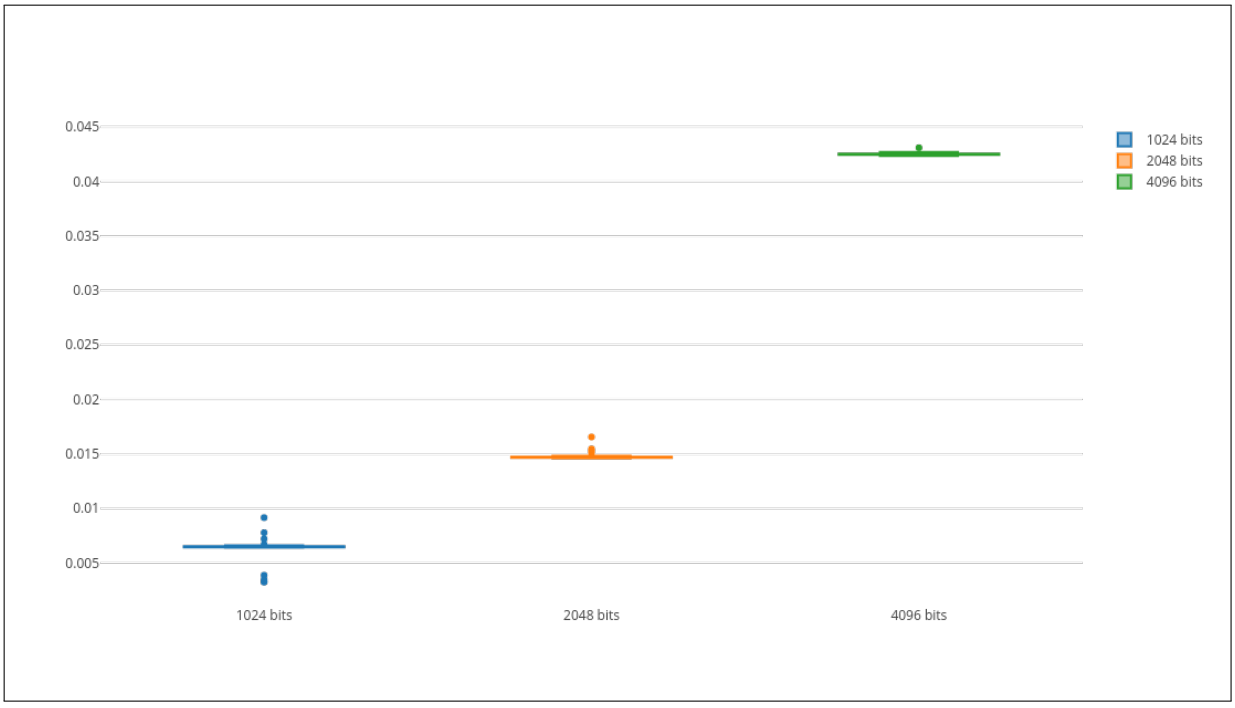
Tabela 5 – Decriptação RSA Raspberry Pi

Chave/Mensagem	688 bits	1712 bits	3760 bits
1024 bits	$0,017s \pm 0,001003$		
2048 bits		$0,072s \pm 0,000371$	
4096 bits			$0,402s \pm 0,008752$

Fonte – Elaborado pelo autor

Podemos visualizar na Figura 29 a disposição das amostras coletadas, utilizando diagrama de caixa, para a plataforma Raspberry Pi, utilizando o algoritmo de criptografia RSA.

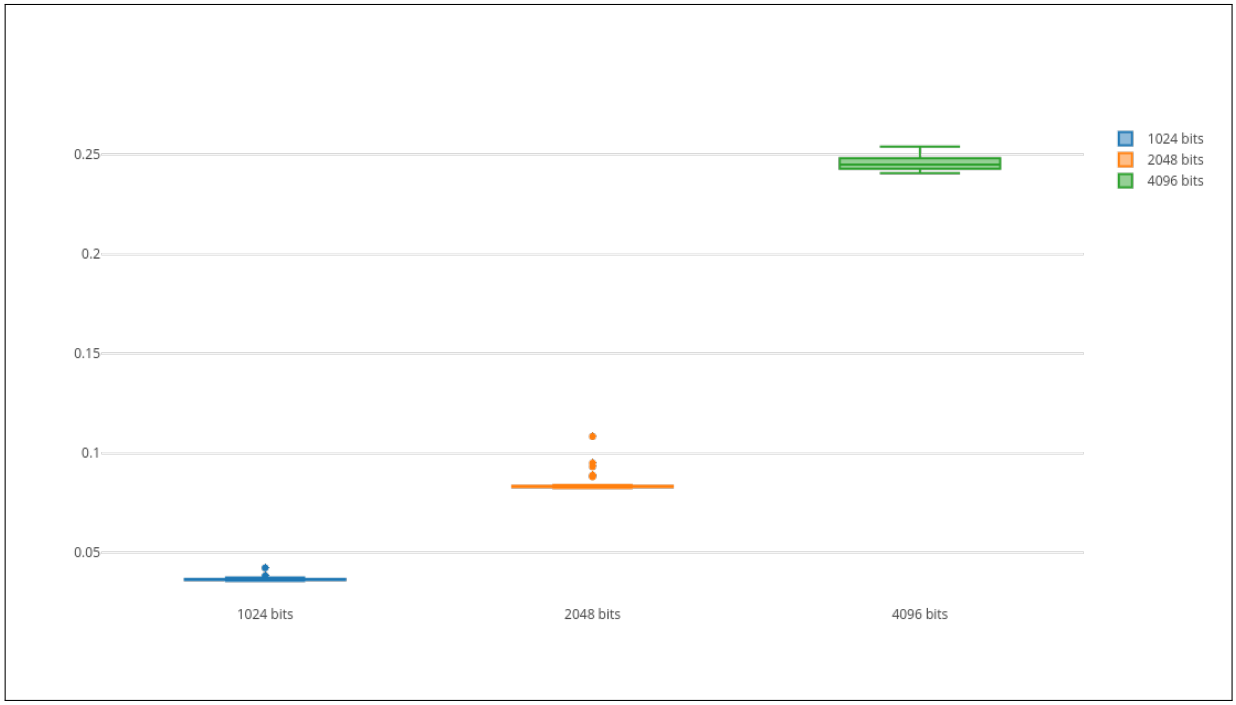
Figura 29 – Gráfico de encriptação RSA para a plataforma Raspberry Pi



Fonte – (Elaborado pelo autor, 2018)

Podemos visualizar na Figura 30 a disposição das amostras coletadas, utilizando diagrama de caixa, para a plataforma Intel Galileo, utilizando o algoritmo de criptografia RSA.

Figura 30 – Gráfico de encriptação RSA para a plataforma Intel Galileo



Fonte – (Elaborado pelo autor, 2018)

No algoritmo de criptografica assimétrico RSA o tempo de decriptação é maior do que o de encriptação. Isso acontece porque na encriptação, utiliza-se a chave pública e na decriptação utiliza-se a chave privada, na qual, possui um maior expoente, tornando o cálculo mais lento e complexo.

Da mesma forma, podemos vizualizar na Tabela 6 o tempo necessário para a Intel Galileo realizar o processo de encriptação, e na Tabela 7 o processo de decriptação.

Tabela 6 – Encriptação RSA Intel Galileo

Chave/Mensagem	688 bits	1712 bits	3760 bits
1024 bits	$0,037s \pm 0,000398$		
2048 bits		$0,085s \pm 0,001801$	
4096 bits			$0,246s \pm 0,001207$

Fonte – Elaborado pelo autor

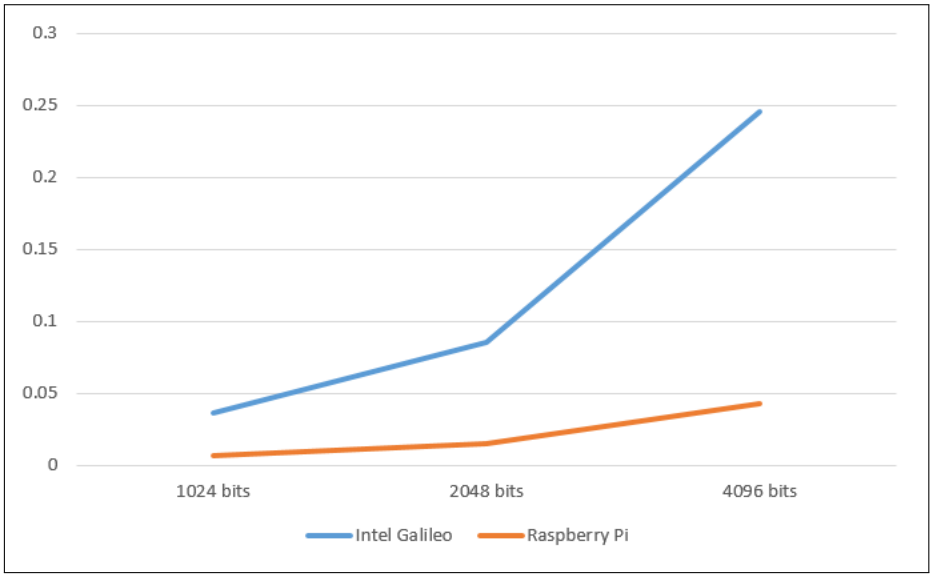
Tabela 7 – Decriptação RSA Intel Galielo

Chave/Mensagem	688 bits	1712 bits	3760 bits
1024 bits	$0,102s \pm 0,001834$		
2048 bits		$0,457s \pm 0,002759$	
4096 bits			$2,825s \pm 0,007519$

Fonte – Elaborado pelo autor

No gráfico abaixo, Figura 31, podemos visualizar o comparativo do processo de encriptação do algoritmo RSA entre as plataformas Raspberry Pi e Intel Galileo.

Figura 31 – Gráfico de encriptação RSA para as plataformas Raspberry Pi e Intel Galileo



Fonte – (Elaborado pelo autor, 2018)

4.3 COMPARATIVO ALGORITMO HÍBRIDO

Com o algoritmo híbrido, podemos obter as vantagens do algoritmo simétrico e assimétrico. Neste trabalho foi utilizados o algoritmo RSA com a chave de 2048 bits e o algoritmo AES com a chave de 256 bits, pois são as recomendadas para obter maior segurança.

Na Tabela 8 podemos visualizar a comparação do desempenho da Raspberry Pi e na Intel Galileo. O desempenho foi mensurado a partir do momento em que o servidor encripta a chave AES, utilizando o algoritmo RSA, e a envia para o cliente (Dispositivos IoT). Até o momento em que o cliente descriptografa a chave AES, utilizando o algoritmo RSA.

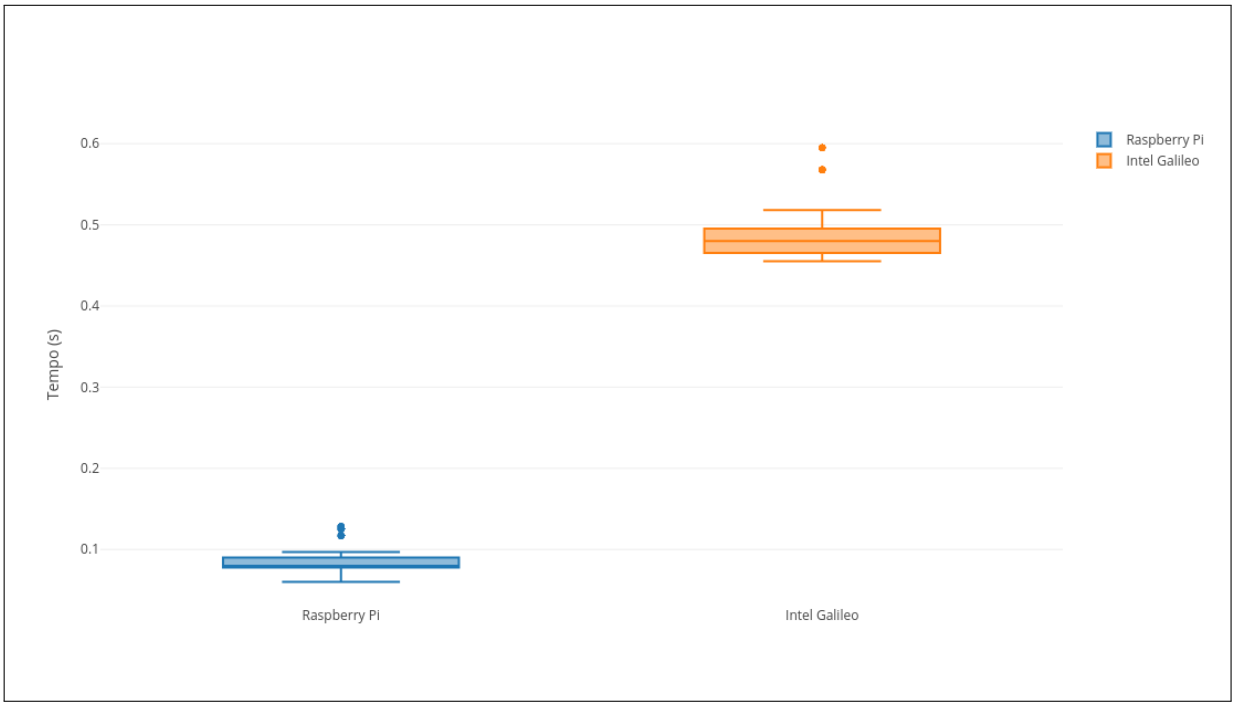
Tabela 8 – Algoritmo Híbrido

Plataforma	Tempo (s)
Raspberry Pi	$0,083 \pm 0,896$
Intel Galileo	$0,473 \pm 0,157$

Fonte – Elaborado pelo autor

Podemos visualizar na Figura 32 o comparativo entre as plataformas Raspberry Pi e Intel Galileo, visualizando a disposição das amostras coletadas.

Figura 32 – Gráfico de encriptação para as plataformas Raspberry Pi e Intel Galileo utilizando o algoritmo híbrido



Fonte – (Elaborado pelo autor, 2018)

Assim como foi verificados nos outros processos, a Raspberry Pi possui um maior desempenho, devido ao maior poder de processamento.

Com base nos resultados que foram obtidos, pode-se escolher qual algoritmo de criptografia se adequa melhor ao ambiente de IoT que está sendo desenvolvido. Pois, após o estudo realizado, é possível verificar as necessidades de segurança do ambiente IoT e escolher um algoritmo de criptografia que não impacte a performance do dispositivos IoT, assim como, da rede que ele estará contido.

5 CONCLUSÕES E TRABALHOS FUTUROS

Foi realizado um levantamento bibliográfico sobre os algoritmos de criptografia AES e RSA, em que auxiliaram em um maior entendimento acerca dos mesmos. Com isso, foi possível entender o funcionamento, a complexidade e a necessidade de cada um deles.

Foi realizado um estudo sobre as plataformas de IoT, o que auxiliou na aplicação dos algoritmos de criptografia em cada uma delas. Tornando possível desenvolver os algoritmos responsáveis pelo processo de criptografia na comunicação entre os dipositivos IoT e o servidor, utilizando bibliotecas de criptografia já existentes.

O estudo realizado sobre as plataformas de IoT e sobre os algoritmos de criptografia, foram de grande importância para a definição dos casos de testes, pois foi possível definir quais seriam os requisitos necessários para a realização dos mesmos. Após definir os casos de testes, foram realizadas as comunicações entre as plataformas IoT e o servidor, conseguindo assim, recolher os dados necessários para a avaliação do impacto dos algoritmos de criptografia.

Através dos dados obtidos, foi possível avaliar o impacto na performance de algoritmos de criptografia tradicionais em ambiente de IoT. De acordo com os dados que foram apresentados, podemos concluir que, idealmente, a plataforma que obteve uma melhor perfomance, tanto em algoritmos de criptografia simétricos quanto assimétricos, foi a Raspberry Pi, devido ao seu melhor poder de processamento. Contudo, deve-se avaliar as necessidades e restrições da solução de IoT, pois a Raspberry Pi possui um maior custo.

Em relação a utilização de algoritmos de criptografia para encriptar mensagens trocadas entre clientes e servidores, deve-se considerar a utilização de algoritmos de criptografia híbridos, pois estes se tornam mais seguros que algoritmos de criptografia simétricos, mas não possuem um grande impacto na performance, pois utilizam o algoritmo RSA apenas para realizar a troca de chaves. O algoritmo de criptografia RSA não é recomendado para a troca de mensagens , pois possui grande impacto na performance, se comparado ao algoritmo de criptografia AES.

Após a avaliação dos dados, pudemos verificar o impacto de cada algoritmo de criptografia em cada ambiente IoT. Podendo assim, auxiliar na escolha de um algoritmo de criptografia para realizar prototipagem de dispositivos IoT, escolhendo o que melhor se adequa ao seu ambiente, com base das necessidades de segurança do mesmo.

5.1 TRABALHOS FUTUROS

É necessário que seja feito estudo acerca de outros algoritmos de criptografia simétricos e assimétricos, para que se possa verificar o impacto de outros algoritmos em sistemas IoT. É desejável que o estudo possa ser feito utilizando outra plataformas de prototipagem, como BeagleBone Black, ESP8286, entre outras, para que se possa visualizar o impacto da criptografia em outros ambiente de IoT.

Após realizar os estudos descritos acima, deve-se desenvolver um algoritmo de criptografia que possa ser aplicado de forma efetiva em ambientes IoT, sem comprometer os recursos dos mesmos.

REFERÊNCIAS

ARDUINO. **Arduino**. 2018. Disponível em: <https://www.arduino.cc>.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. **Computer Networks**, Elsevier BV, v. 54, n. 15, p. 2787–2805, 2010.

BASSI, A.; BAUER, M.; FIEDLER, M.; KRAMP, T.; KRANENBURG, R. V.; LANGE, S.; MEISSNER, S. **Enabling things to talk: Designing IoT solutions with the IoT**. [S.l.]: Springer, 2013.

CORREA, S. M. B. B. **Probabilidade e Estatística**. [S.l.]: Pontifícia Universidade Católica de Minas Gerais, 2003.

DAEMENA, J.; RIJMEN, V. **AES Proposal: Rijndael**. [S.l.], 1999.

FINKLE, J. **JJ warns diabetic patients: Insulin pump vulnerable to hacking**. 2016. <https://www.reuters.com/article/us-johnson-johnsoncyber-insulin-pumps-e/jj-warns-diabetic-patients-insulin-pump-vulnerableto-hacking-idUSKCN12411L>. Acesso 15/04/2017.

GARTNER. **Gartner Says 8.4 Billion Connected "Things"Will Be in Use in 2017, Up 31 Percent From 2016**. 2018. <https://www.gartner.com/newsroom/id/3598917>. Acesso 08/05/2018.

GIBBS, S. **Hackers can hijack Wi-Fi Hello Barbie to spy on your children**. 2015. <https://www.theguardian.com/technology/2015/nov/26/hackers-can-hijack-wi-fi-hello-barbie-to-spy-on-your-children>. Acesso 15/04/2017.

HUSAMUDDIN, M.; QAYYUM, M. Internet of things: A study on security and privacy threats. **2nd International Conference On Anti-cyber Crimes**, IEEE, 2017.

INTEL. **Galileo Datasheet**. 2018. <https://www.intel.com/content/dam/support/us/en/documents/galileo/sb/galileo_datasheet_329681_003.pdf>. Acesso 11/05/2018.

ISHA; LUHACH, A. K. Analysis of lightweight cryptographic solutions for internet of things. **Indian Journal Of Science And Technology**, Indian Society for Education and Environment, v. 9, n. 28, p. 1–7, 2016.

KHAN, R.; KHAN, S. U.; ZAHEER, R.; KHAN, S. Future internet: The internet of things architecture, possible applications and key challenges. **2012 10th International Conference on Frontiers of Information Technology**, IEEE, 2012.

KUROSE, J.; ROSS, K. **Redes de computadores e a internet: Uma abordagem top-down**. [S.l.]: Pearson, 2014.

MERCES, F. **Cryptocurrency-Mining Malware Targeting IoT, Being Offered in the Underground**. 2018. <https://blog.trendmicro.com/trendlabs-security-intelligence/cryptocurrency-mining-malware-targeting-iot-being-offered-in-the-underground/>. Acesso 08/05/2018.

MORETTIN, L. G. **Estatística Básica: Probabilidade e Inferência**. [S.l.]: Pearson, 2010.

NARU, E. R.; SAINI, H.; SHARMA, M. A recent review on lightweight cryptography in iot. **2017 International Conference on IoT in Social, Mobile, Analytics and Cloud (I-SMAC)**, IEEE, 2017.

PAAR, J. P. C. **Understanding cryptography: A textbook for students**. [S.l.]: Springer, 2010.

PYCRYPTO. 2012. <https://www.dlitz.net/software/pycrypto/api/2.6/>. Acesso 15/03/2018.

Raspberry Pi Foundation. **Raspberry Pi**. 2017. Disponível em: <https://www.raspberrypi.org/>.

RIVEST, R. l; SHAMIR, A.; ADLEMAN, L. **A Method for Obtaining Digital Signatures and Public-Key Cryptosystems**. [S.l.], 1997.

RSA Laboratories. **Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1**. 2003. <https://tools.ietf.org/html/rfc3447#section-7.1>. Acesso 16/07/2018.

SPANOS, G. 2018. <https://github.com/spaniakos/AES>. Acesso 15/03/2018.

STALLINGS, W. **Criptografia e segurança de redes: princípios e práticas**. [S.l.]: Pearson, 2015. v. 6.

STEIN, W. **Elementary Number Theory: Primes, Congruences, and Secrets**. 2017. <https://wstein.org/ent/ent.pdf>. Acesso 20/05/2018.

TALWANA, J. C.; HUA, H. J. Smart world of internet of things (iot) and its security concerns. **Ieee International Conference On Internet Of Things (ithings) And Ieee Green Computing And Communications (greencom) And Ieee Cyber, Physical And Social Computing (cpscom) And Ieee Smart Data (smartdata)**, IEEE, p. 240–245, 2016.

VAAS, L. **350,000 cardiac devices need a security patch**. 2018. <https://nakedsecurity.sophos.com/2018/05/04/half-a-million-pacemakers-need-a-security-patch/>. Acesso 08/05/2018.

We are Social. **Digital in 2018: Essential insights into Internet, Social Media, Mobile, and E-Commerce use around the world**. 2018. <https://wearesocial.com/blog/2018/01/globaldigital-report-2018>. Acesso 06/05/2018.

WILLIAMSON, D. F.; PARKER, R.; KENDRICK, J. S. The box plot: A simple visual method to interpret data. **Annals of internal medicine**, Research Gate, 1989.

WU, M.; LU, T.-J.; LING, F.-Y. Research on the architecture of internet of things. **2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)**, IEEE, p. 484–487, 2010.

Yocto Project. 2018. <https://www.yoctoproject.org/>. Acesso 11/05/2018.

APÊNDICES

Tabela 9 – Encriptação AES Raspberry Pi

Mensagem/Chave	128 bits	256 bits
16 bits	$0.0006s \pm 0.000371$	$0.0007s \pm 0.000516$
32 bits	$0.0005s \pm 0.000075$	$0.0006s \pm 0.000121$
64 bits	$0.0004s \pm 0.000013$	$0.0009s \pm 0.000086$
128 bits	$0.0005s \pm 0.000073$	$0.001s \pm 0.000107$
256 bits	$0.0004s \pm 0.000012$	$0.0007s \pm 0.000157$
512 bits	$0.0004s \pm 0.000012$	$0.0007s \pm 0.000157$
1 KB	$0.0005s \pm 0.000114$	$0.0006s \pm 0.000089$
2 KB	$0.0005s \pm 0.000076$	$0.0006s \pm 0.000077$
4 KB	$0.0006s \pm 0.000081$	$0.0007s \pm 0.000148$
8 KB	$0.0007s \pm 0.000075$	$0.0009s \pm 0.000095$
16 KB	$0.001s \pm 0.000086$	$0.0011s \pm 0.000093$
32 KB	$0.0025s \pm 0.000103$	$0.0029s \pm 0.00027$

Fonte – Elaborado pelo autor

Tabela 10 – Decriptação AES Raspberry Pi

Mensagem/Chave	128 bits	256 bits
16 bits	$0.0001s \pm 0.000004$	$0.0001s \pm 0.000003$
32 bits	$0.0001s \pm 0.000008$	$0.0001s \pm 0.000022$
64 bits	$0.0001s \pm 0.000002$	$0.0001s \pm 0.000005$
128 bits	$0.0001s \pm 0.00001$	$0.0002s \pm 0.000007$
256 bits	$0.0001s \pm 0.000002$	$0.0001s \pm 0.00002$
512 bits	$0.0001s \pm 0.000007$	$0.0001s \pm 0.00001$
1 KB	$0.0002s \pm 0.000011$	$0.0002s \pm 0.000022$
2 KB	$0.0002s \pm 0.00001$	$0.0003s \pm 0.000015$
4 KB	$0.0004s \pm 0.000029$	$0.0005s \pm 0.000039$
8 KB	$0.0007s \pm 0.000037$	$0.0008s \pm 0.000053$
16 KB	$0.0013s \pm 0.000048$	$0.0015s \pm 0.00006$
32 KB	$0.0026s \pm 0.000077$	$0.0029s \pm 0.000058$

Fonte – Elaborado pelo autor

Tabela 11 – Encriptação AES Intel Galileo

Mensagem/Chave	128 bits	256 bits
16 bits	$0.0064s \pm 0.000126$	$0.0065s \pm 0.000131$
32 bits	$0.0067s \pm 0.000356$	$0.0066s \pm 0.000331$
64 bits	$0.0065s \pm 0.00014$	$0.0065s \pm 0.000161$
128 bits	$0.0068s \pm 0.000403$	$0.007s \pm 0.000618$
256 bits	$0.0066s \pm 0.000167$	$0.0069s \pm 0.00036$
512 bits	$0.0069s \pm 0.00017$	$0.0069s \pm 0.000123$
1 KB	$0.0074s \pm 0.000476$	$0.0072s \pm 0.000287$
2 KB	$0.0076s \pm 0.00013$	$0.0078s \pm 0.000119$
4 KB	$0.0087s \pm 0.000126$	$0.0092s \pm 0.000318$
8 KB	$0.011s \pm 0.000526$	$0.0117s \pm 0.000521$
16 KB	$0.0148s \pm 0.000214$	$0.0168s \pm 0.000462$
32 KB	$0.0234s \pm 0.000386$	$0.0266s \pm 0.000483$

Fonte – Elaborado pelo autor

Tabela 12 – Decriptação AES Intel Galileo

Mensagem/Chave	128 bits	256 bits
16 bits	$0.0011s \pm 0.000104$	$0.0011s \pm 0.000075$
32 bits	$0.0011s \pm 0.000061$	$0.0011s \pm 0.000031$
64 bits	$0.0011s \pm 0.000053$	$0.0011s \pm 0.000053$
128 bits	$0.0011s \pm 0.000033$	$0.0012s \pm 0.000033$
256 bits	$0.0014s \pm 0.000325$	$0.0014s \pm 0.000059$
512 bits	$0.0015s \pm 0.000045$	$0.0016s \pm 0.00011$
1 KB	$0.0018s \pm 0.000049$	$0.0019s \pm 0.000043$
2 KB	$0.0025s \pm 0.00006$	$0.0027s \pm 0.00005$
4 KB	$0.0039s \pm 0.00008$	$0.0043s \pm 0.000054$
8 KB	$0.0068s \pm 0.000308$	$0.0074s \pm 0.000105$
16 KB	$0.0125s \pm 0.000339$	$0.0135s \pm 0.000042$
32 KB	$0.0231s \pm 0.000338$	$0.026s \pm 0.000172$

Fonte – Elaborado pelo autor