PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA E DE ARTES
GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

JOÃO ALAN PIMENTA DE MELLO ERNESTO

COMPARAÇÃO ENTRE RSA E ECC NO ÂMBITO DA CERTIFICAÇÃO DIGITAL

GOIÂNIA
2023

JOÃO ALAN PIMENTA DE MELLO ERNESTO

COMPARAÇÃO ENTRE RSA E ECC NO ÂMBITO DA CERTIFICAÇÃO DIGITAL

Trabalho de Conclusão de Curso apresentado à Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, referente a parte dos requisitos para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador(a): Prof(a). Ma. Angélica da Silva Nunes.

GOIÂNIA
2023

JOÃO ALAN PIMENTA DE MELLO ERNESTO

COMPARAÇÃO ENTRE RSA E ECC NO ÂMBITO DA CERTIFICAÇÃO DIGITAL

Este Trabalho de Conclusão de Curso julgado adequado para obtenção do título de Bacharel em Engenharia de Computação, e aprovado em sua forma final pela Escola de Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, em _____/_____/_____.

_____
Prof.ª Ma. Ludmilla Reis Pinheiro dos Santos
Coordenadora de Trabalho de Conclusão de Curso

Banca Examinadora:

_____
Orientadora: Prof.ª Ma. Angélica da Silva Nunes

_____
Prof. Me. Fernando Gonçalves Abadia

_____
Prof. Me. Wilmar Oliveira de Queiroz

GOIÂNIA
2023

DEDICATÓRIA

Dedico a Deus por me dar força e ajudar a alcançar meus objetivos. A meus pais por serem meus maiores incentivadores e que nunca duvidaram de mim. A minha família por sempre me apoiar.

AGRADECIMENTOS

A professora Ma. Angélica da Silva Nunes, orientadora acadêmica, que teve papel fundamental na elaboração deste trabalho, com seu apoio e dedicação.

A família do meu amigo Ronan Neto que me amparou.

Aos meus colegas que me auxiliaram na trajetória acadêmica, em específico Gabriel Diniz e Watyson Soares.

Então Samuel pegou uma pedra e a ergueu entre Mispá e Sem; e deu-lhe o nome de Ebenézer, dizendo: "Até aqui o Senhor nos ajudou".

1 Samuel 7:1

RESUMO

Este trabalho compara o RSA e o ECC no âmbito da certificação digital, através de pesquisas biográficas e implementação de códigos na linguagem Python. Com ênfase a segurança, desempenho, interoperabilidade e suporte a padrões. O objetivo é mostrar a importância da segurança e o desempenho em certificados digitais que podem estar vulneráveis a ataques, focando nos dois algoritmos citados. Para o estudo de caso foi implementado dois códigos na linguagem Python, que tem por objetivo medir o tempo de criptografia e decriptografia das mensagens, a partir do tamanho das chaves e das mensagens. Como resultado, foi possível analisar como é o comportamento das duas criptografias quando são utilizadas em diversos casos.

Palavras-Chave: RSA. ECC. Certificado digital. Segurança. Criptografia. Decriptografia.

ABSTRACT

This work compares the RSA and the ECC in the field of digital certification, through biographical research and implementation of codes in the Python language. With an emphasis on security, performance, interoperability and standards support. The objective is to show the importance of security and performance in digital certificates that may be related to attacks, focusing on the two algorithms mentioned. For the case study, two codes were implemented in the python language, which aims to measure the encryption and decryption time of messages, based on the size of keys and messages. As a result, it was possible to analyze how the two encryptions behave when used in different cases.

Keywords: RSA. ECC. Digital Certificate. Security. Cryptography. Decryption.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

AES-GCM	*Advanced Encryption Standard - Galois/Counter Mode,* Padrão de Criptografia Avançada - Modo Galois/Contador
AC	Autoridade Certificadora
AR	Autoridade de Registro
CRL	*Certificate Revocation List,* Lista de Certificados Revogados
DAS	*Digital Signature Algorithm,* Algoritmo de Assinatura Digital
DLP	*Discrete Logarithm Problem*, Problema do Logaritmo Discreto
ECC	*Elliptic curve cryptography, Criptografia de curva elíptica*
ECDSA	Elliptic Curve Digital Signature Algorithm, Algoritmo de Assinatura Digital com Curvas Elípticas
HKDF	*Hierarchical Key Derivation Function,* Função de Derivação de Chave Hierárquica
ICP	Infraestrutura de Chaves Públicas
ITI	Instituto Nacional de Tecnologia da Informação
ITU	*International Telecommunication Union,* União Internacional das Telecomunicações
LSEC	Laboratório de Sistemas Embarcados Críticos
NIST	*National Institute of Standards and Technology,* Instituto Nacional de Padrões e Tecnologia
ONU	Organizações das Nações Unidas
PUCGO	*Pontifícia Universidade Católica de Goiás*
PKI	*Public Key Infrastructure,* Infraestrutura de Chave Pública
RSA	*Rivest Shamir Adleman*
SHA-1	*Secure Hash Algorithm 1*, Algoritmo Hash de Segurança 1

SUMÁRIO

1 INTRODUÇÃO

Desde os primórdios, o homem busca formas e meios para se defender e proteger seus bens. Inicialmente, se utilizavam de abrigos, pedras e o fogo para proteção. Posteriormente, o homem foi evoluindo e se aprimorando na arte da segurança. Essencialmente, a palavra “segurança” tem como origem o termo (*secura*) do latim, que significa “sem preocupações” (MATOS, 2007).

Essa definição no contexto da segurança da informação está relacionada com a ideia de se proteger dados e informações de algum individuo ou organização. Para o *National Institute of Standards and Technology*, Instituto Nacional de Padrões e Tecnologia (NIST, 1995 apud Stallings, 2014, p. 7), no *Computer Security Handbook* (Livro de Bolso de Segurança de Computadores), a segurança de computadores pode ser definida como:

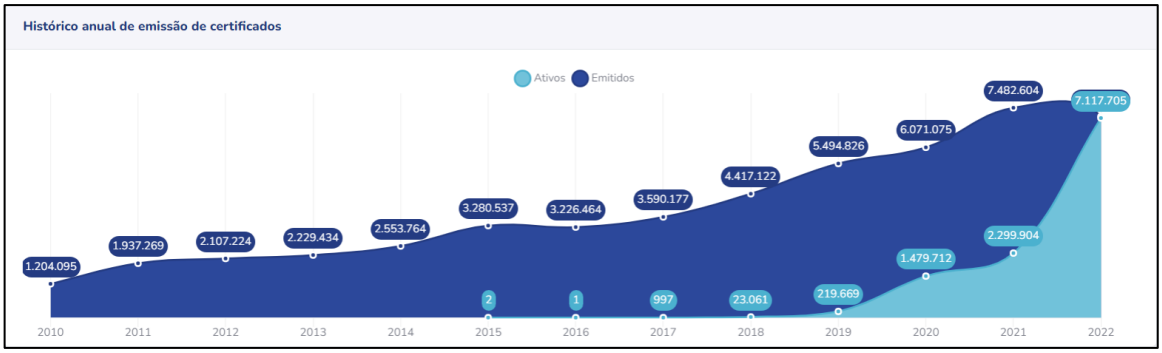
> A proteção oferecida a um sistema de informação automatizado para atingir os objetivos apropriados de preservação da integridade, disponibilidade e confidencialidade de ativos de sistemas de informação (incluindo hardware, software, firmware, informações/dados e telecomunicações).

Dentro da área de segurança, existem mecanismos para proteger as informações, fisicamente ou logicamente. Os controles físicos limitam a entrada de pessoas indesejadas em um determinado lugar. Exemplo: portas, trancas, guardas, dentre outros. Os controles lógicos são aqueles que limitam o acesso de usuários no ambiente eletrônico. Exemplo: criptografia, assinatura digital, sistemas biométricos, dentre outros.

A partir do avanço da tecnologia, serviços que precisavam ser feitos muita das vezes através de cartórios, passaram a ser feitos de forma *online*. Não é mais necessário um indivíduo ir ao cartório e assinar ou autenticar documentos, é preciso apenas de um certificado digital válido que comprove que, de fato, foi ele que assinou o documento. Logo, é relevante saber como esse certificado valida a assinatura de uma pessoa, quem a valida e de que forma se dá isso.

A emissão de certificados digitais vem aumentando. De acordo com *site* do Instituto Nacional de Tecnologia da Informação (ITI), em 2019 foram emitidos 5,4 milhões de certificados, no ano seguinte, foram emitidos 6 milhões e de 2021 para 2022, teve um salto de quase 1,5 milhão de novos certificados emitidos, como mostra a Figura 1. (ITI, 2022).

Figura 1 - Gráfico com histórico anual de emissão de certificados



Fonte: ITI, 2022

Existem diversas alternativas de proteger um certificado digital. Um deles e o mais utilizado é o *Rivest Shamir Adleman* (*RSA*), que utiliza da dificuldade de fatorar o produto de dois enormes números primos. (Carvalho, 2001). O *Elliptc Curve Cryptography,* Criptografia de Curva Elíptica (ECC), utiliza da dificuldade em encontrar um logaritmo distinto dentro de uma curva elíptica aleatória. (ENTRUST, 2022).

Neste trabalho, são comparadas as duas formas de criptografia que são utilizados na certificação digital, o RSA e o ECC.

Diante do contexto apresentado, este trabalho pretende responder a seguinte questão: Quais são as vantagens e desvantagens das aplicabilidades do ECC e RSA como instrumentos de segurança para o certificado digital?

1.1 Objetivo Geral

- Identificar as vantagens e desvantagens das aplicabilidades do ECC e RSA como instrumentos de segurança para certificado digital.

1.2 Objetivos Específicos

- Contextualizar segurança de sistemas e seus principais elementos;
- Apresentar as funcionalidades segurança de certificados digitais;
- Diferenciar e comparar ECC com o RSA;
- Apresentar o resultado da comparação.

1.3 Metodologia

Esta pesquisa quanto a natureza, é um resumo de assunto, pois tem como objetivo sistematizar a área do conhecimento para o âmbito acadêmico (WAZLAWICK, 2014).

Quanto aos objetivos, é uma pesquisa descritiva, pois são descritas as características dos objetos de estudo, haverá comparações entre eles e um resultado demostrando os pontos positivos e negativos de cada criptografia. (WAZLAWICK, 2014).

Quanto aos procedimentos técnicos, é uma pesquisa bibliográfica, pois manipula dados e faz comparações entre eles com objetivo de se obter resultados quanto a utilidade de cada tipo de criptografia. (WAZLAWICK, 2014)

1.4 Estrutura da Monografia

No Capítulo 2, é feita a fundamentação teórica sobre a segurança de computadores, com seus principais conceitos e sua importância.

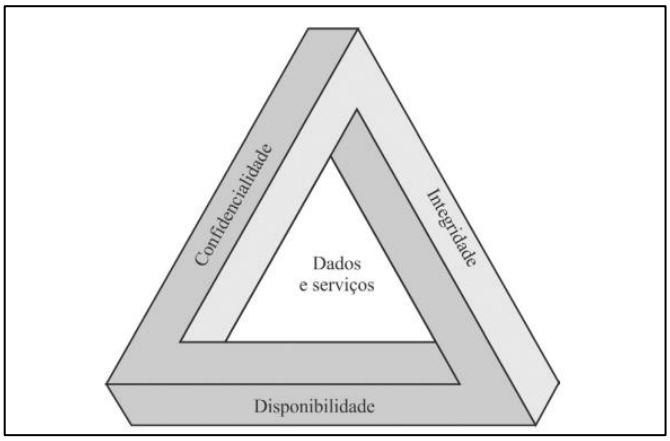
O Capítulo 3 aborda a comparação entre RSA e ECC a partir do tamanho da chave por SSL, o ambiente da implementação, a comparação entre o RSA e o ECC em relação ao tamanho das chaves, a difusão do RSA e ECC pelo mundo, a difusão do RSA e ECC no Brasil, o tempo de criptografia e decriptografia para vários tamanhos de chave, o tempo de criptografia e decriptografia para vários tamanhos de mensagem que foram utilizadas como base para a elaboração e compreensão dos testes que foram realizados.

No capítulo 4 são apresentadas as considerações finais sobre a importância da comparação entre o RSA e o ECC no âmbito da Certificação Digital e as sugestões de trabalhos futuros.

2 REFERENCIAL TEÓRICO

A segurança no meio digital traz consigo a tríade de requisitos para a proteção de dados como mostra a Figura 2. São eles: confidenciabilidade, integridade e disponibilidade. Além desses três, diversos autores incluem também a autenticidade e o não repúdio.

Figura 2 - A tríade de requisitos de segurança



Fonte: STALLINGS, 2014

A confidenciabilidade assegura a privacidade do usuário e dos seus dados, podendo assim, ficarem disponíveis apenas para pessoas permitidas. A integridade sustenta a veracidade dos dados enviados e a sua alteração de forma correta, feita apenas por pessoas responsáveis. A disponibilidade traz a garantia que o sistema funcionará sempre que for necessário e que o acesso é feito de forma segura. (STALLINGS, 2014)

Além da tríade de segurança, há ainda a autenticidade que tem o propósito de provar que, de fato, as mensagens foram enviadas através do usuário, sem ninguém se passar por ele. Por fim, o não repúdio que tem como objetivo assegurar que o autor não refute a autoria e a assinatura do documento. (STALLINGS, 2014)

2.1 Serviços de segurança

Os serviços de segurança são encargos que garantem a proteção do sistema. Existem diversos serviços para a segurança e a *International Telecommunication Union*, União Internacional das Telecomunicações (ITU) que é uma agência da Organização das Nações Unidas (ONU) responsável por regulamentar e padronizar

as telecomunicações internacionais e desenvolveu recomendações para os serviços de segurança, são elas:

- Autenticação;
- Controle de acesso;
- Confidencialidade de dados;
- Integridade de dados;
- Irretratabilidade.

A autenticação é a certeza de que a mensagem foi enviada por quem de fato afirma ser. Por exemplo, a Pontifícia Universidade Católica de Goiás (PUCGO) envia um *e-mail* para um determinado aluno. Como o estudante pode garantir que aquele *e-mail* de fato veio da secretaria da universidade? Ele consegue essa garantia através do domínio do *e-mail*. O domínio geralmente é o nome da instituição que vem logo após o caractere @ em um endereço de *e-mail*. A PUCGO tem o seu domínio registrado no *site* nic.br. Portanto, todo endereço de *e-mail* referente a ela, contém o sufixo pucgo.edu.br. Dessa forma, o aluno tem a certeza de que aquele *e-mail* foi enviado através da instituição de ensino.

O controle de acesso determina quem pode e quem não pode acessar um determinado sistema, uma determinada página ou aplicação. Somente pessoas previamente cadastradas tem o acesso e devidamente autenticadas.

A confidencialidade de dados tem como objetivo proteger a transmissão e o fluxo de dados. Logo, todo dado que circula em um determinado sistema tem a sua devida proteção. A integridade de dados garante que os dados que foram recebidos são os mesmos que foram enviados, sem nenhuma alteração. A irretratabilidade confirma que os dados foram enviados através do remetente. Sendo assim, o usuário que enviou alguma informação não pode afirmar que não foi ele que enviou (STALLINGS, 2014).

2.2 Mecanismos de segurança

Além dos serviços de segurança, a ITU aborda os mecanismos de segurança, e aqui vale ressaltar dois: a criptografia e a assinatura digital.

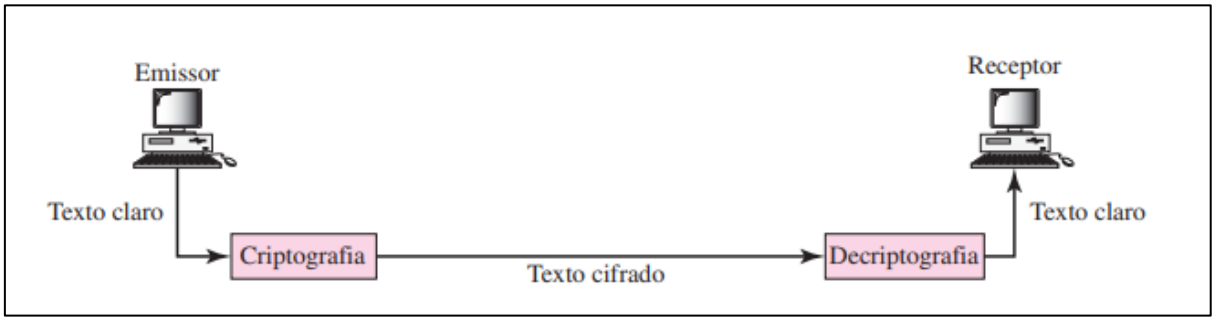
A criptografia utiliza de algoritmos matemáticos para embaralhar informações. Desta forma, as informações ficam inacessíveis para quem não tem acesso à chave de criptografia (STALLINGS, 2014).

A assinatura digital tem como objetivo provar a origem e a integridade, para que assim, fique livre de falsificação. Além disso, toda assinatura digital necessita de um sistema de chave pública, o qual, é abordado ao decorrer deste trabalho (STALLINGS, 2014).

2.3 Criptografia

A criptografia é o ato do emissor utilizar de um algoritmo para proteger, embaralhar uma mensagem (FOROUZAN, 2010).

Figura 3 - Componentes da criptografia



Fonte: FOROUZAN, 2010

A criptografia, se faz necessário a utilização de alguns componentes, as quais são: texto claro, texto cifrado, cifra e a chave, conforme mostra a Figura 3. O texto claro é a mensagem, o dado na sua forma original, antes de ser transformado. O texto cifrado é a mensagem, o dado após ser cifrado. As cifras são os algoritmos de criptografia e decriptografia. A chave é um número ou números aleatórios que se faz necessário para operar a criptografia e a decriptografia (FOROUZAN, 2010).

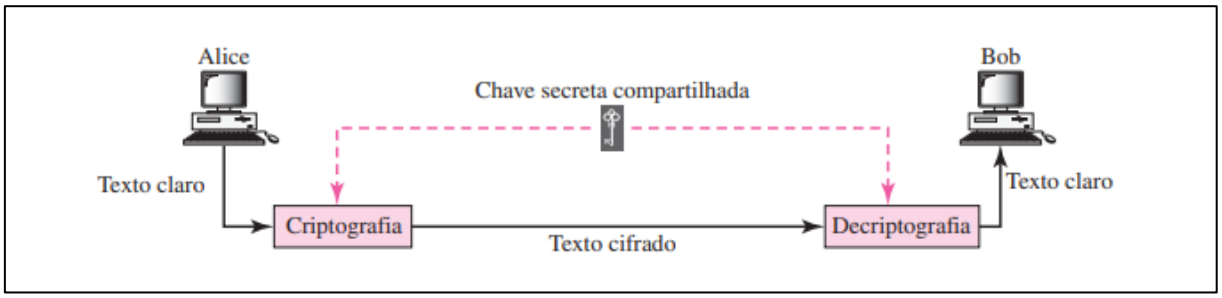
2.3.1 Tipos de criptografia

Existem dois tipos de criptografia: a criptografia de chave simétrica e a criptografia de chave assimétrica (FOROUZAN, 2010).

2.3.2 Criptografia de chave simétrica

Na criptografia de chave simétrica, a mesma chave é utilizada tanto para criptografar quanto para decriptografar os dados. Isso significa que a chave usada para codificar as informações é idêntica à chave utilizada para decodificar, garantindo assim a reversibilidade do processo, conforme mostra a Figura 4 (FOROUZAN, 2010).

Figura 4 - Criptografia de chave simétrica

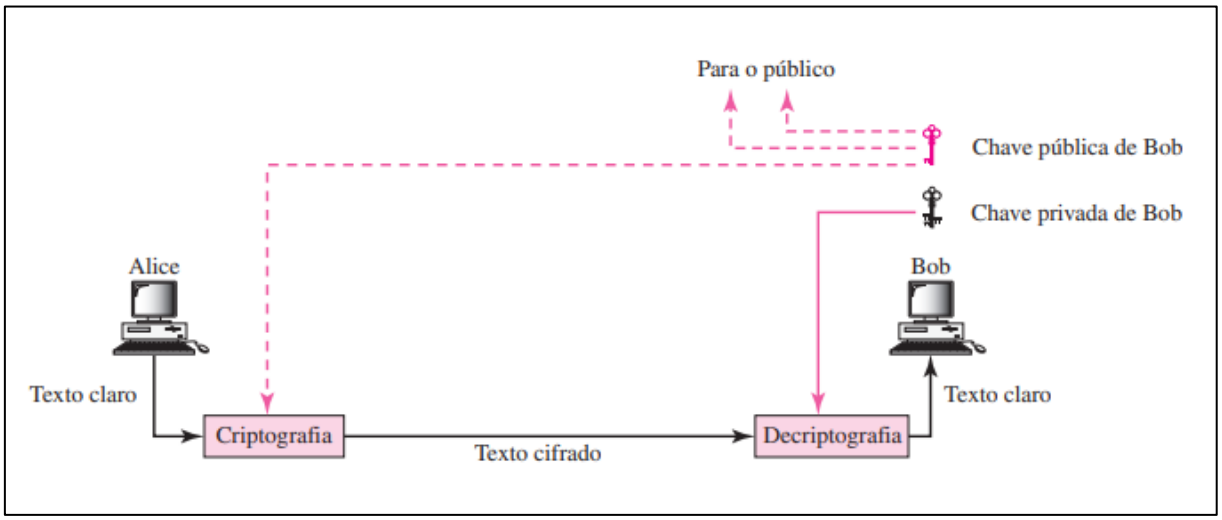


Fonte: FOROUZAN, 2010

2.3.3 Criptografia de chave assimétrica

A criptografia de chave assimétrica envolve o uso de duas chaves distintas no processo de criptografia e decriptografia. Essas chaves são conhecidas como chave privada e chave pública. A chave privada é mantida em posse exclusiva do destinatário ou receptor dos dados, enquanto a chave pública é divulgada e pode ser acessada pelo público em geral, conforme mostra a Figura 5 (FOROUZAN, 2010).

Figura 5 - Criptografia de chave assimétrica



Fonte: FOROUZAN, 2010

Essa abordagem, conforme descrita por Forouzan (2010), permite uma troca segura de informações. O remetente utiliza a chave pública do destinatário para criptografar os dados, garantindo que apenas o destinatário, com a posse da chave privada correspondente, seja capaz de decriptografar e acessar os dados originais. A divulgação da chave pública possibilita que qualquer pessoa criptografe informações para enviar ao destinatário, mantendo a confidencialidade dos dados transmitidos.

A integridade da mensagem pode ser promovida de outra maneira, além da relevância do conteúdo, por meio da assinatura. Nesse contexto, o remetente criptografa a mensagem usando sua chave privada antes de enviá-la ao destinatário. Ao receber a mensagem, o destinatário a decriptografa para verificar se foi realmente enviada pela pessoa alegada. Portanto, embora o conteúdo da mensagem possa ser acessado por qualquer pessoa, a autoria da mensagem será identificada.

A criptografia de chave assimétrica, como destacado por Forouzan (2010), é uma estratégia eficiente para garantir a segurança e a privacidade das comunicações, fornecendo um meio seguro para a troca de informações entre remetentes e destinatários.

Existem diversos algoritmos criptográficos de chave assimétrica, porém, esse trabalho descreve apenas dois deles, que são o RSA e o ECC.

2.3.4 RSA

Em 1977, foi desenvolvido um algoritmo de chave pública por Ron Rivest, Adi Shamir e Len Adleman, por isso, se dá o nome de RSA que são as letras iniciais dos sobrenomes de cada autor. Segundo Forouzan (2010), esse é o ele é o algoritmo mais utilizado no quesito criptografia de chave pública.

Segundo Forouzan (2010), o RSA funciona da seguinte maneira: Primeiramente, seleciona-se as chaves. Para isso, é preciso seguir algumas etapas:

- Escolhe-se dois números primos grandes, p e q;
- Multiplica-se p por q para encontrar n, logo, $n = p \times q$. n é o módulo para criptografia e decriptografia;
- Multiplicam-se $(p - 1) \times (q - 1)$ para obter Φ;
- Escolhe um número aleatório e, logo após, calcula-se $d = 1 \bmod \Phi$;
- O módulo n e o expoente e são anunciados ao público e o Φ e d são privados;

- Após essas etapas, para um indivíduo enviar uma mensagem criptografada para outra pessoa é necessário seguir essa formula: $C = p^e \ (mod\ n)$, no qual o *C* é o texto criptografado, *n* e *e* são o módulo e o expoente público;
- Para a pessoa que recebeu o texto cifrado, para fazer a decriptografia, é necessário utilizar a chave privada *d* na formula: $P = C^d \ (mod\ n)$.

Existe uma condição para que o RSA funcione, o valor de *P* necessita ser menor que o valor de *n*.

Forouzan (2010) menciona que há vantagens e desvantagens na utilização do RSA. As vantagens são:

- Útil para mensagens curtas;
- Muito utilizado para assinatura digital e autenticação.

As desvantagens do RSA são:

- Muito lento para mensagens longas;
- Chaves muito extensas.

2.3.5 ECC

Em 1985, foi desenvolvido um algoritmo de chave pública por Neal Koblitz e Victor Miller. A criptografia utiliza de curvas elípticas sobre campos finitos. Esses campos finitos são conjuntos feitos por números finitos de elementos. O interessante para o estudo do ECC é o campo finito F_p, pois ele é um conjunto de inteiros módulo *p*, no qual *p* é um número primo. Esse conjunto consiste em inteiros de *0* até $p - 1$ e a aritmética modular que faz as operações matemáticas. (LSEC, 2016)

As curvas elípticas sobre F_p, são conjunto de (*x*,*y*) que satisfazem a equação $y^2 \ mod\ p = x^3 + ax + b \ mod\ p$, com $4a^3 + 27b^2 \neq 0$. Logo, as curvas elípticas singulares não podem ser utilizadas, pois seus resultados não são interessantes para a criptografia. *A*, *b*, *x* e *y* pertencem a F_p junto com um ponto especial que se chama ponto no infinito *O*. (LSEC, 2016)

O ponto base *G* usado na criptografia das curvas elípticas possui ordem *n* e um cofator *h*. Quanto maior é o número primo, mais pontos existem na curva elíptica. Isso é importante, pois a mensagem criptografada é mapeada em ponto desse. Logo, quanto mais pontos, melhor. (LSEC, 2016).

O *Discrete Logarithm Problem*, Problema do Logaritmo Discreto (DLP) é o que garante a segurança do ECC. Esse problema consiste em *P* e *Q* que são valores

conhecidos. É necessário encontrar k tal que $Q = kP$. Já no caso do campo finito F_p, conhecidos a e b, encontrar k tal que $b = a^k \ mod \ p$ (LSEC, 2016).

2.3.5.1 Geração de chaves no ECC

A chave privada no ECC consiste em d, um inteiro escolhido entre 1 e $n - 1$, tal que n é a ordem do ponto base G. A chave pública é $D = dG$. (LSEC, 2016)

A criptografia de um texto claro se dá através do emissor criptografar a mensagem M, que foi mapeada em um ponto com a sua chave privada e pública do receptor. $C = M + d_e D_r$. (LSEC, 2016)

A mensagem cifrada C chega no receptor e, para que ele possa decriptografar a mensagem, é necessário usar a sua chave privada e a chave pública do emissor. $M = C – d_r D_e$. (LSEC, 2016)

2.3.5.2 ECDSA

Segundo o LSEC (2016), *Digital Signature Algorithm*, Algoritmo de Assinatura Digital (DAS) é o algoritmo que, juntando-o com o ECC, formam o *Elliptic Curve Digital Signature Algorithm*, Algoritmo de Assinatura Digital com Curvas Elípticas (ECDSA). Essa junção, permite que o indivíduo confirme a autenticidade do emissor através da sua chave pública. A forma de funcionamento é:

- Aplica-se uma função de *hash* na mensagem Z;
- Escolhe-se um inteiro k aleatoriamente entre 0 e $n – 1$;
- Calcula o ponto $P = kG = (X_p, y_p)$ e $r = x_p \ mod \ n \neq 0$;
- Calcula-se $s = k^{-1} (z + rd_e) \ mod \ n \neq 0$, onde ($r$, s) é a assinatura do emissor.

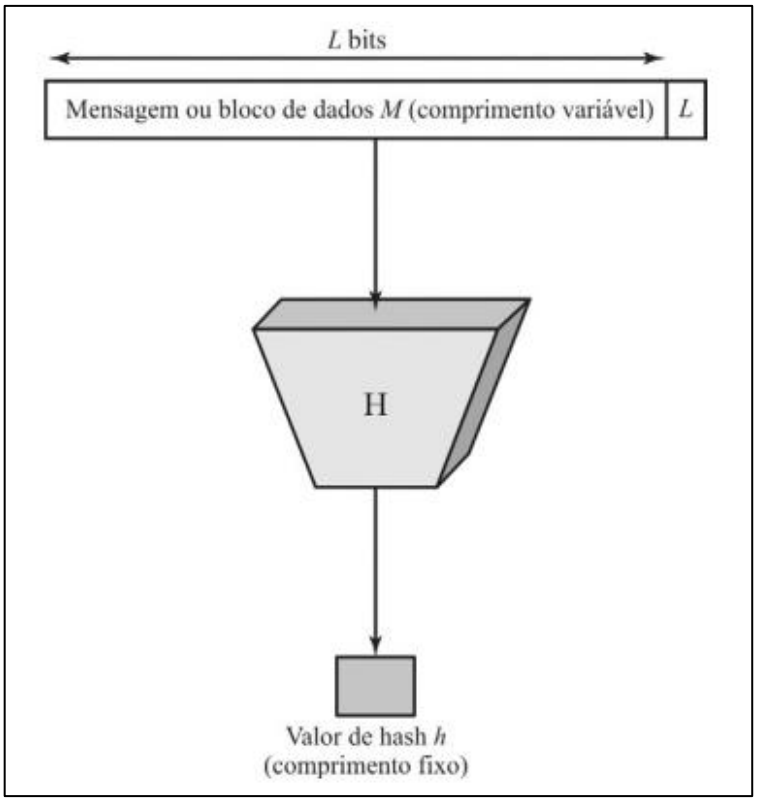
Para verificar se a assinatura é válida, o LSEC (2016) destaca que o receptor tem que fazer os cálculos:

- Calcula-se o inteiro de $u_1 = s^{-1} z \ mod \ n$;
- Calcula-se o inteiro de $u_2 = s^{-1} r \ mod \ n$;
- Calcula-se o Ponto $P = u_1 G + u_2 D_e = (x_p, y_p)$;
- A assinatura é válida se r *for igual a* $x_p \ mod \ n$.

2.4 Função de *hash*

A função de *hash* é um algoritmo matemático que recebe um dado de tamanho variado e produz um resumo de tamanho fixo criptografado como mostra a Figura 6. Além disso, a função de *hash* garante a integridade da mensagem, mas só não autêntica a mensagem. É necessária a combinação com a cifração convencional que é utilizada quando as partes interessadas compartilham a chave criptográfica. Ou então, a criptografia de chave pública que utilizam chaves diferentes. (STALLINGS, 2014).

Figura 6 - Função de hash em diagrama de bloco

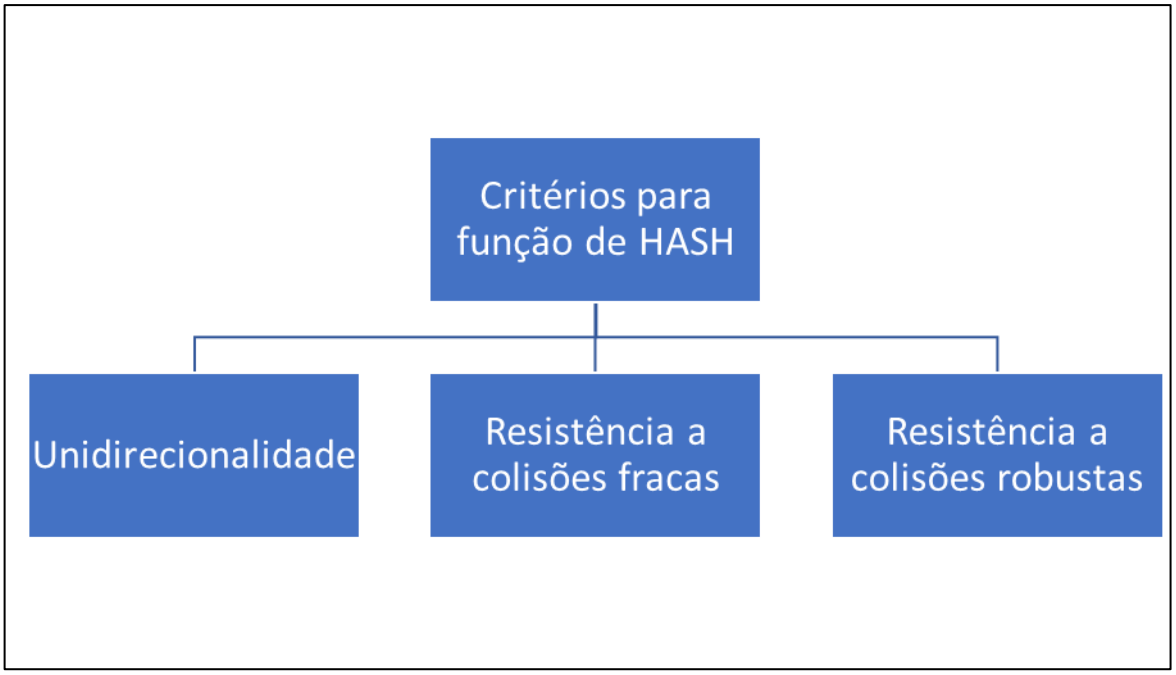


Fonte: STALLINGS, 2014

2.4.1 Critérios para Função de hash

Existem três critérios para a função de *hash* como mostra a Figura 7, são eles: unidirecionalidade, resistência a colisões fracas e resistência a colisões robustas.

Figura 7 - Critérios para função de hash



Fonte: Autoria própria, dados extraídos de FOROUZAN, 2010

A unidirecionalidade garante que, ao se criar um resumo *hash h(M)* de uma mensagem *M*, deve ser impossível encontrar *M* a partir do seu resumo *h(M)*. A resistência a colisões fracas garante que se houver uma entrada *a* e um *hash h*, deve ser difícil, quase impossível achar outra entrada que produza o mesmo *hash h*. A resistência a colisões robustas garante que deve ser difícil, quase impossível encontrar entradas distintas que produzam o mesmo *hash h* (FOROUZAN, 2010).

2.4.2 Algoritmos de hash

Existem diversos algoritmos de *hash*, mas o que mais se usa *Secure Hash Algorithm 1*, Algoritmo *Hash* de Segurança 1 (SHA-1). Ele foi desenvolvido através do NIST no ano de 1995 após o SHA que veio no ano de 1993. Inicialmente o SHA-1 gerava *hashes* de 160 *bits*, mas não eram suficientes. Então, no ano de 2002 foi publicado outras três versões com comprimentos de 256, 384 e 512 *bits*. (STALLINGS, 2014).

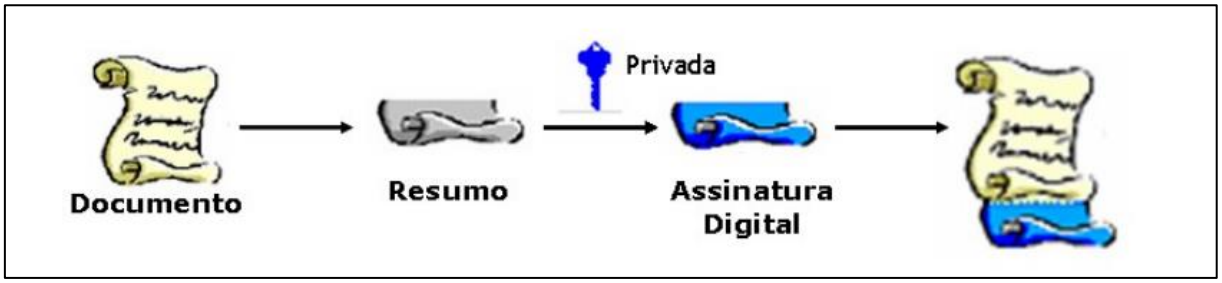
2.5 Assinatura digital

A assinatura digital é utilizada quando a confidencialidade do conteúdo da mensagem não é importante, mas quem a enviou. A criptografia de chave pública é bastante requisitada para prover a autenticação. Para essa finalidade, o emissor gera um resultado *hash* de um documento eletrônico, depois disso, o emissor cifra o resultado *hash* com sua chave privada, associada a uma chave pública constante do seu certificado digital, gerando a assinatura digital, por fim, o documento eletrônico e a assinatura digital ficam associados, para futura validação (ITI, 2009).

Para fazer a verificação, o documento eletrônico e a assinatura digital associada são disponibilizados para o receptor, juntamente com o certificado digital do emissor. Em seguida, o receptor calcula novamente o resultado *hash* do documento eletrônico. Posteriormente, o receptor decifra a assinatura digital com a chave pública do emissor, contida no certificado digital, obtendo o resultado *hash* gerado pelo emissor. Por fim, o receptor compara os resultados *hash* obtidos nos passos anteriores. Se forem iguais, significa que o documento eletrônico está íntegro e que é possível identificar o emissor por meio do certificado digital. Caso contrário, a assinatura digital é considerada inválida (ITI, 2009).

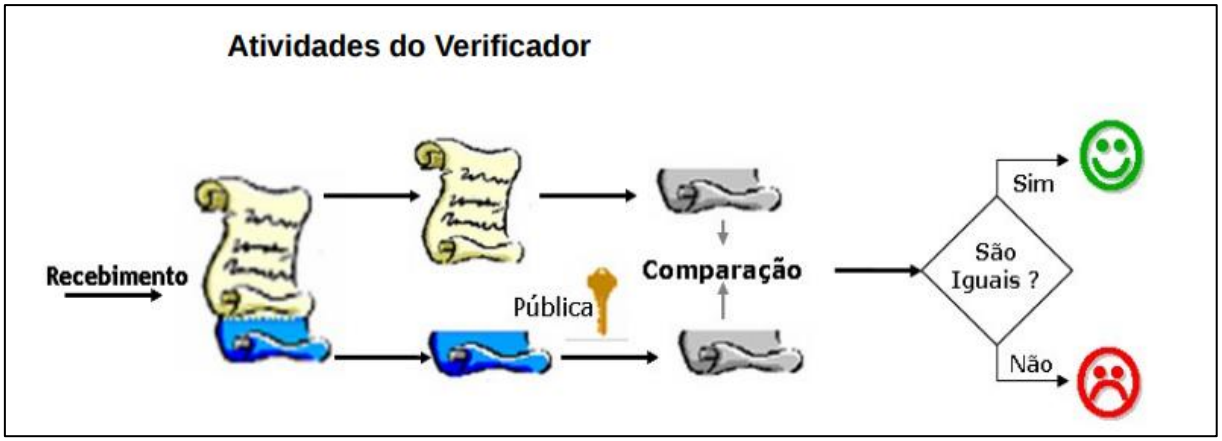
A Figura 8 mostra o processo da criação da assinatura digital e a Figura 9 mostra o processo da verificação da assinatura digital (ITI, 2009).

Figura 8 - Criação da assinatura digital



Fonte: ITI, 2009

Figura 9 - Verificação da assinatura digital



Fonte: ITI, 2009

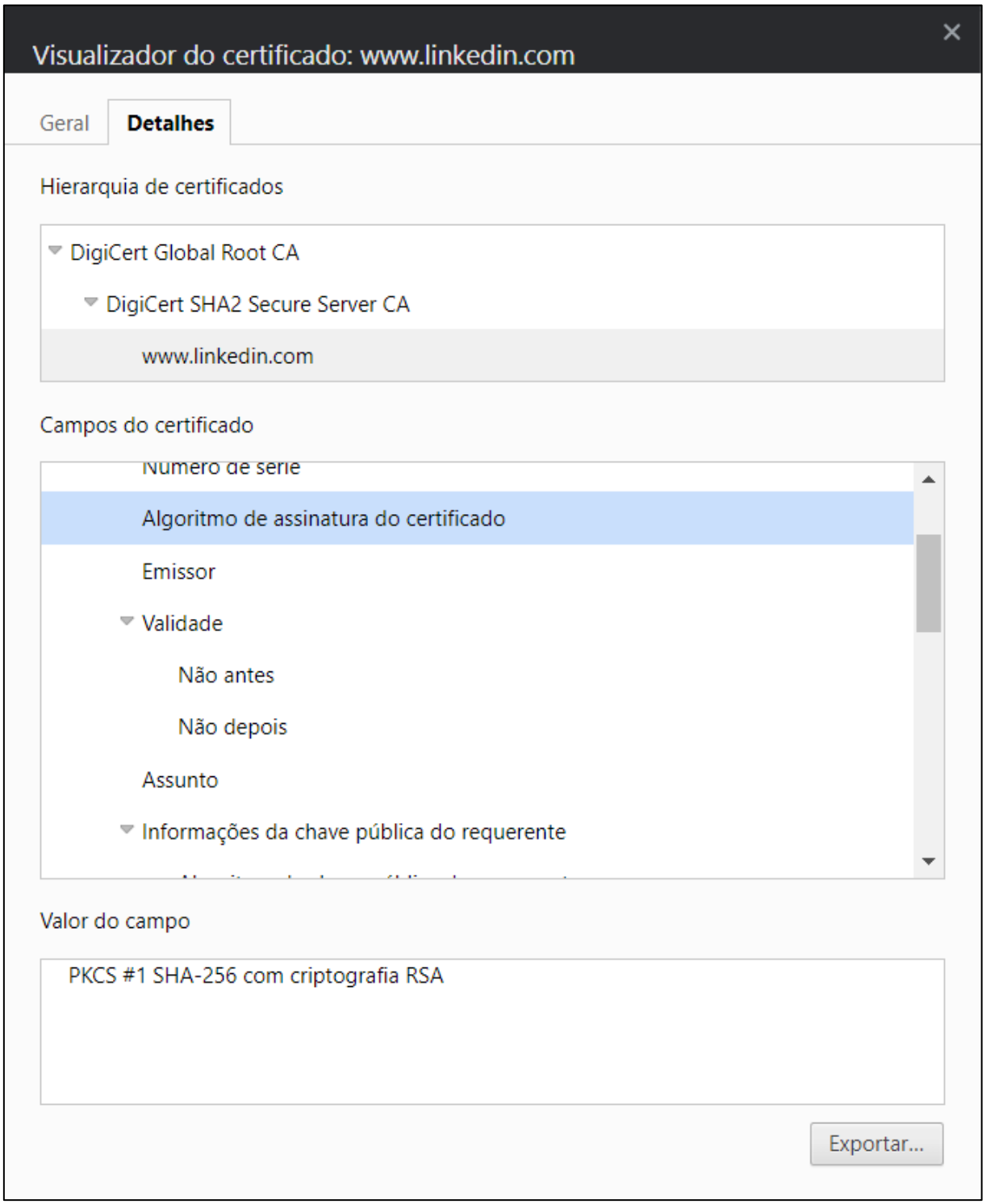
2.6 Certificado digital

Qualquer pessoa pode se passar por outra pessoa na *Internet*. Isso não é diferente quando o assunto está relacionado com assinatura digital. Devido a isso, o certificado digital surgiu para que uma autoridade certificadora confiável possa autentificar uma pessoa. O certificado digital vem com a Identificação (ID) do proprietário da chave pública. A chave pública e a sua assinatura são geradas por uma autoridade que na maioria das vezes é uma empresa do governo ou uma organização autorizada para tal fim. Ressalta-se que, o certificado digital tem data de validade. Logo, quem quer que precise da chave pública desse usuário pode obter o certificado e verificar se ele é válido por meio da assinatura confiável anexada. (STALLINGS, 2014).

Tanto o RSA quanto o ECC são aplicados no contexto da certificação digital para garantir a autenticidade e integridade dos certificados, bem como possibilitar a assinatura digital dos dados para verificação de autenticidade e não repúdio. Eles desempenham um papel crucial na segurança das transações eletrônicas, proteção de dados e estabelecimento de confiança em ambientes digitais (STALLINGS, 2014).

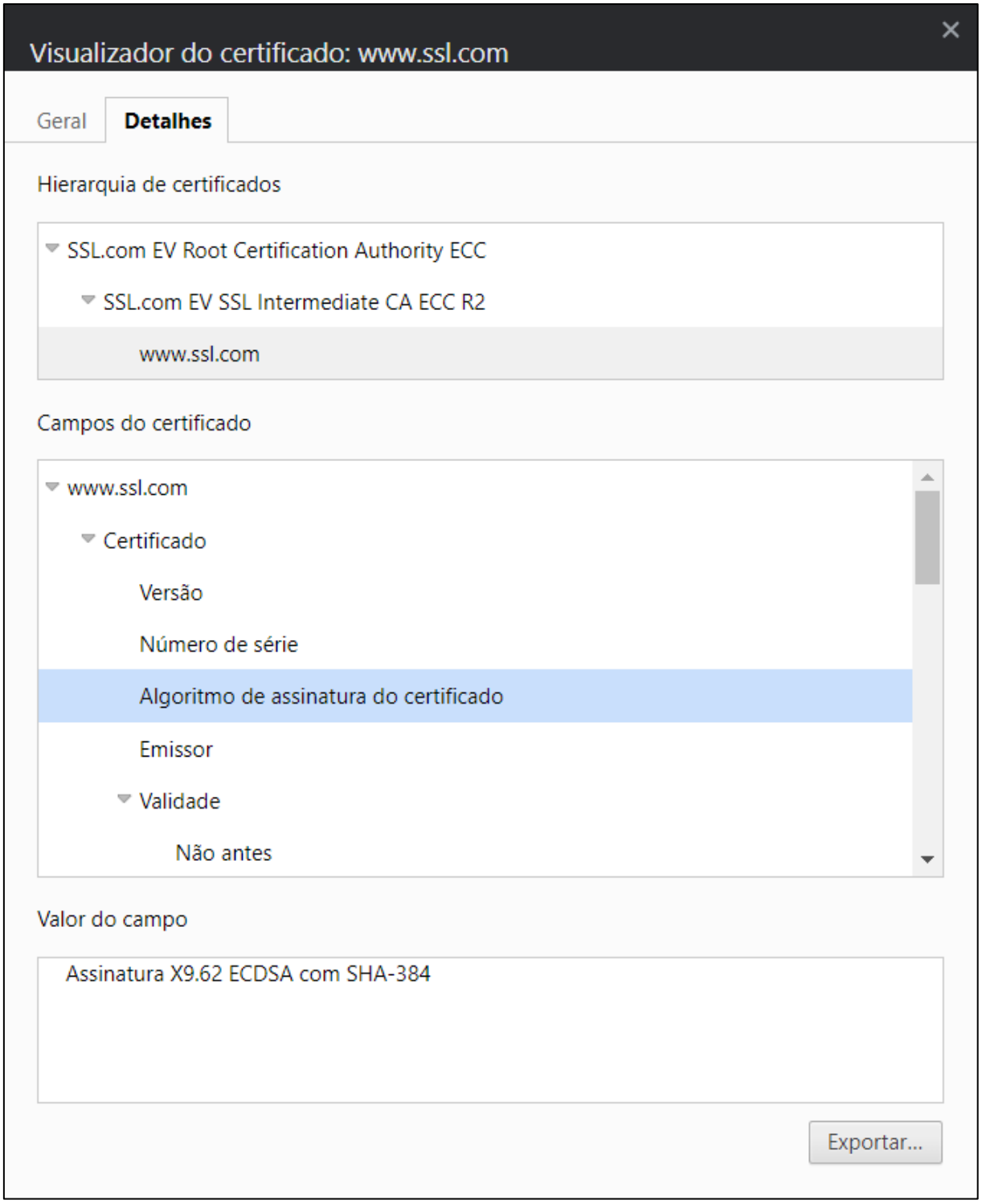
A Figura 10 mostra que o algoritmo de assinatura do certificado digital do *site* da empresa Linkedin utilizou o RSA. Já a Figura 11, mostra que o algoritmo de assinatura do certificado digital do *site* da empresa do SSL.com utilizou o ECDSA, no qual, é outra nomenclatura para o ECC.

Figura 10 - Algorítimo de assinatura do certificado do site Linkedin



Fonte: Linkedin, 2023

Figura 11 - Algoritmo de assinatura do certificado do site SSL.com

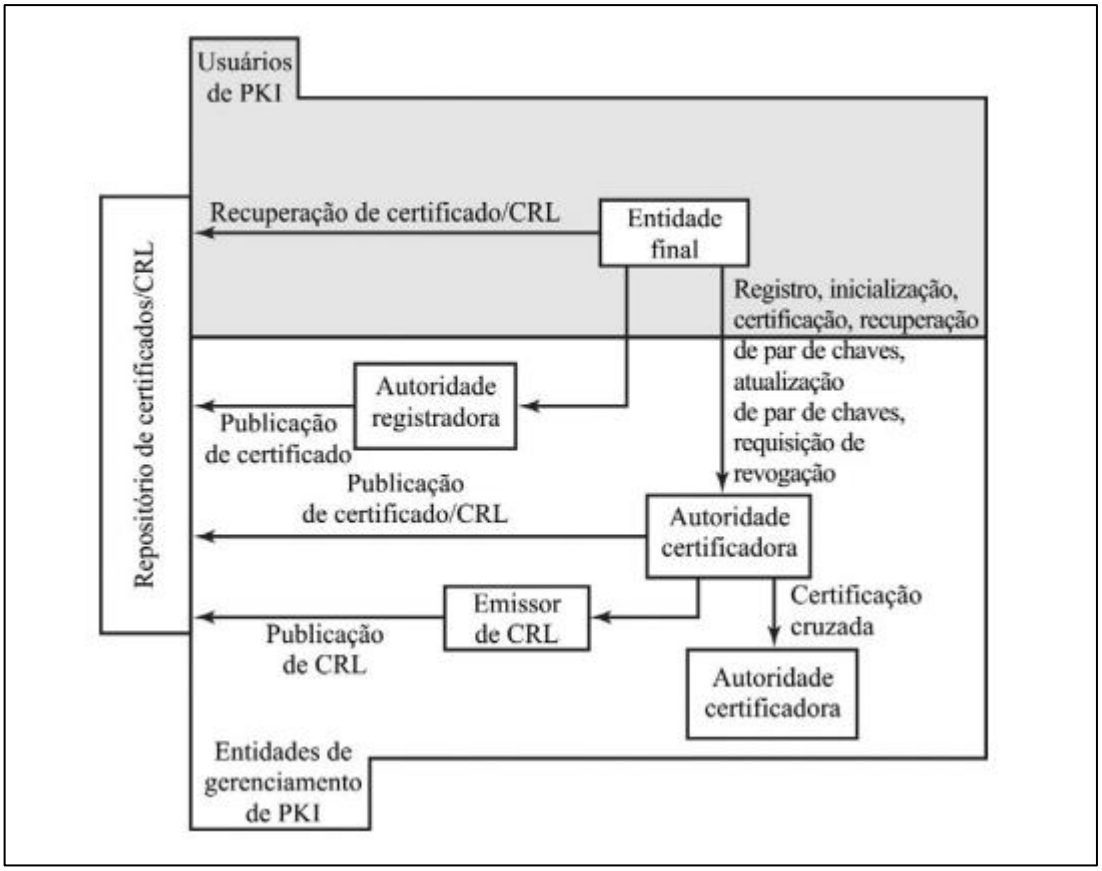


Fonte: SSL.com, 2023

2.6.1 Infraestrutura de chave pública

A *Public Key Infrastructure*, Infraestrutura de Chave Pública (PKI) é tudo aquilo que envolve os procedimentos utilizados para gerenciar um certificado digital.

Figura 12 - Arquitetura da PKI



Fonte: STALLINGS, 2014

Os elementos da PKI são exibidos na Figura 12 e são:

- Entidade final: é o usuário que adquiriu um certificado;
- Autoridade Certificadora (AC): é quem emite e renova os certificados;
- Autoridade Registradora (AR): é o local que, na maioria das vezes a entidade final procura para fazer o processo de registro;
- Emissor de *Certificate Revocation List*, Lista de Certificados Revogados (CRL): é aquele que pode divulgar a lista que contém os certificados revogados;
- Repositório: é utilizado para armazenar os certificados e os CRLs de forma correta, para que, se for preciso, é fácil o acesso e confiável.

A relação entre os elementos é: a entidade final procura uma AR para poder comprar o certificado e fazer todo o registro. A partir do registro, a AC emite o certificado para o usuário final. Destaca-se que o certificado tem data de validade. Logo, a AC envia o certificado para o CRL, que revoga o certificado. Precisando assim, a entidade final atualizar o par de chaves. (STALLINGS, 2014)

3 COMPARAÇÃO ENTRE RSA E ECC NA SEGURANÇA DE MENSAGENS

Nesse capítulo, o algoritmo RSA e o ECC são utilizados para análise comparativa, explorando-se as características e os benefícios de cada um dos algoritmos, a fim de compreender sua eficácia na segurança de mensagens.

3.1 Ambiente da implementação

O objetivo desse capítulo é comparar o desempenho do RSA e do ECC no contexto da certificação digital, abordando quatro aspectos: o tamanho das chaves, difusão de cada um em nível global e no Brasil e tempo de criptografia e decriptografia para vários tamanhos de chave e mensagens.

A avaliação da difusão de cada algoritmo foi feita por meio de pesquisa bibliográfica e a medição do tempo necessário para criptografar e decriptografar como RSA e o ECC foi implementado por meio de um *script* na linguagem Python.

Com base nos resultados obtidos, é possível analisar as vantagens e desvantagens de cada criptografia em diferentes cenários de uso, identificando as diferenças e limitações de cada um. O estudo visa contribuir para a compreensão dos aspectos técnicos e práticos envolvidos na escolha de uma criptografia assimétrica para a certificação digital e outros sistemas de segurança.

Os resultados foram obtidos a partir dos testes utilizando um *notebook* da marca Acer, com o processador Intel(R) Core (TM) i7-7700HQ CPU @ 2.80GHz, 2808 Mhz, 4 Núcleo(s), 8 Processador(es) Lógico(s), 16 GB de memória RAM DDR4 com a frequência de 2400Hz, a placa de vídeo GTX 1060 de 6GB de memória e um SSD adata xpg gammix s11 pro 256GB de memória m.2 nvme. O *software* utilizado para implementação dos códigos na linguagem Python foi o Visual Studio Code na versão: 1.76.0.

3.2 Comparação entre o RSA e o ECC em relação ao tamanho das chaves

O *site* ssl.com compara o ECC com o RSA com relação ao tamanho da chave de criptografia. São apresentados vários tamanhos de chaves de criptografia RSA e ECC que possuem níveis de segurança criptográfica equivalentes. Observa-se que, ECC requer chave criptográfica de tamanho menor que o RSA para um dado nível de segurança. A Tabela 1 apresenta os valores dessa comparação. (SSL, 2019)

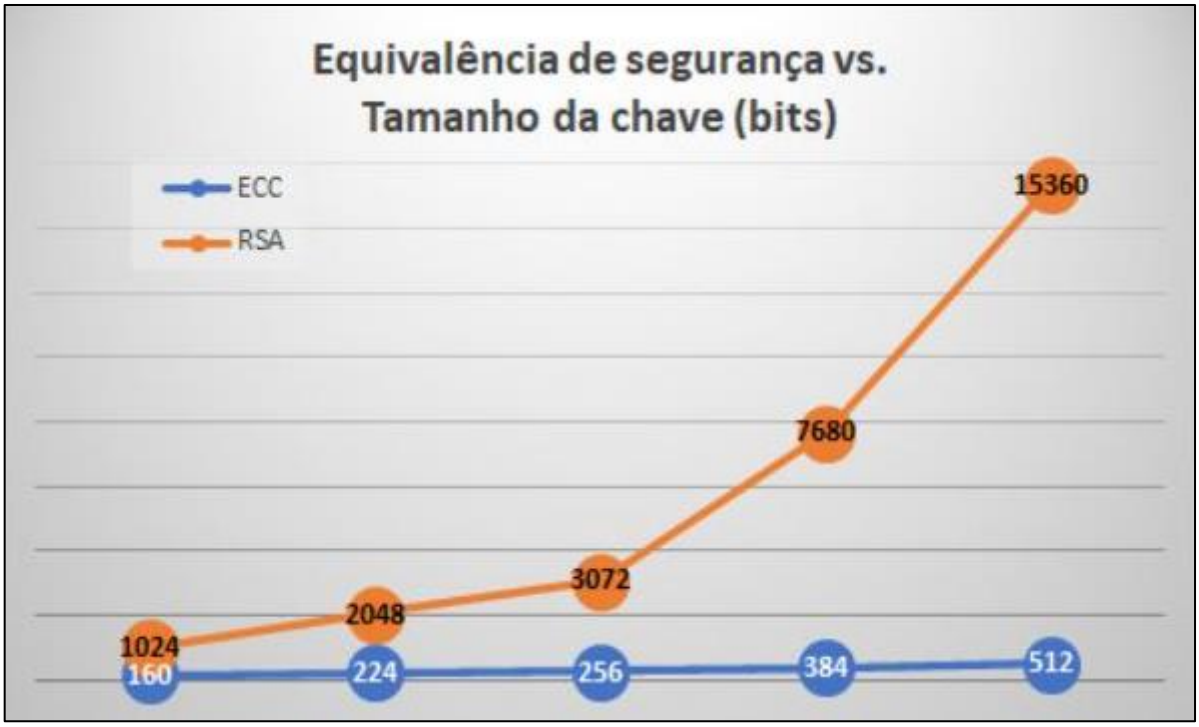
Tabela 1 - Tamanho de chave para segurança equivalente no ECC e no RSA (*bit*)

ECC	RSA
163	1024
224	2048
256	3072
384	7680
512	15360

Fonte: SSL, 2019

A partir dos dados da Tabela 1, pode-se afirmar que há discrepância no tamanho das chaves entre a técnica de ECC e o algoritmo RSA. A chave do RSA é muito maior quando comparada com a do ECC, para o mesmo nível segurança. Consequentemente, o tamanho reduzido das chaves ECC possibilita o uso de criptografia mais robusta com menor carga de processamento, o que diminui o tempo no processo de criptografia e decriptografia. (SSL, 2019).

Figura 13 - Equivalência de segurança vs. Tamanho da chave



Fonte: Elaborado pelo autor do trabalho a partir de SSL, 2019

Para melhor ilustrar a diferença de equivalência de segurança e o tamanho da chave, foi feito o gráfico mostrado na Figura 13, no qual observa-se que, para obter o

mesmo nível de segurança, o RSA requer um tamanho de chave muito superior quando comparado com o ECC. Além disso, na medida em que o nível de segurança aumenta, o tamanho da chave no RSA aumenta exponencialmente, o que não ocorre no ECC.

É válido destacar, ademais, que o algoritmo RSA utilizando chave de 48 *bits* já foi quebrado por meio do algoritmo *Shor* e, segundo o grupo que trabalhou nisso, no futuro será possível a quebra de chaves de 2048 *bits*, utilizando computadores quânticos (Kaminsky, 2023).

Segundo Sangalli (2012), a técnica ECC também já foi quebrada utilizando um sistema com chave de 109 *bits* pela técnica *birthday attack,* ataque aniversário*.*

3.3 Difusão do RSA e ECC pelo mundo

O algoritmo RSA, disponibilizado como técnica de criptografia pioneira para uso em certificação digital, ainda é o mais utilizado em todo o mundo. Contudo, o algoritmo ECC emerge como uma alternativa competitiva. A adoção generalizada da técnica de criptografia de curvas elípticas tem sido limitada por alguns fatores, que são descritos a seguir. (Sangalli, 2012)

A complexidade dos cálculos matemáticos envolvidos na implementação do ECC tem sido um desafio para a sua difusão. Embora seja demonstrado que o ECC é mais seguro e tem chave menor, alguns usuários ainda acreditam que, quanto maior a chave, maior a proteção. Além disso, sistemas que utilizam o algoritmo RSA há algum tempo podem não suportar a implementação do ECC, e requerem atualizações (Sangalli, 2012).

3.4 Difusão do RSA e ECC no Brasil

O principal órgão brasileiro que regulamenta as chaves públicas no Brasil é a Infraestrutura de Chaves Públicas Brasileira (ICP - Brasil). A instituição surgiu através da Medida Provisória 2.200-2 no ano de 2021, inicialmente para regulamentar as transações eletrônicas (Victor, 2021).

A pandemia do COVID-19 acelerou o processo da transformação digital do Brasil em 4 vezes, porém, o Brasil teve um prejuízo de mais de 104 bilhões de reais

em 2021 devido a ineficiência nos serviços de processos digitais e autenticação (Mozelli, 2023).

Uma das tentativas para amenizar essa perda, é visto na Medida Provisória 983/2020 que tem como intuito agilizar os atendimentos, facilitar para a população em relação a demora nas filas de espera e desburocratizar os processos, especialmente durante a pandemia do COVID-19 (Andrade, 2023). A Figura 14 faz parte da propaganda do governo para disseminar a assinatura digital e aumentar a adesão da população brasileira.

Figura 14 - Assinatura digital

Fonte: ANFIP-SP, 2022

A MP 983/2020 permite que qualquer pessoa faça sua assinatura eletrônica através do *site* do governo, utilizando os cadastros dos bancos que atuam no Brasil como forma de facilitar a autenticação. Isso torna o processo simples e rápido para aqueles que possuem conta digital em bancos como Caixa e Banco do Brasil, dentre outros. É importante ressaltar que a assinatura eletrônica tem implicações legais e

está sendo regulamentada no Brasil pela Medida Provisória MP 2.200-2/2001 e pela Lei nº 14.063/2020, que garantem sua validade jurídica.

Segue o passo a passo extraído do gov.br (2021), para que qualquer cidadão possa fazer sua assinatura digital:

1) Acessar o Portal de Assinatura Eletrônica utilizando a sua conta gov.br. Caso não tenha uma conta, é necessário fazer uma e validar de acordo as normas exigidas para poder usufruir da assinatura eletrônica;
2) Fazer o login na conta usando o CPF e a senha;
3) Adicionar o arquivo que será assinado;
4) Escolher o local da assinatura no documento;
5) Assinar o documento;
6) Baixar o documento assinado;
7) Consultar assinatura do documento.

No desenvolvimento desse trabalho foi enviado uma manifestação de dúvida no *site* do Instituto Nacional de Tecnologia da Informação, o qual rege o ICP-Brasil, perguntando acerca da utilização do RSA e do ECC por ele no Brasil. Na resposta foi informado que, ele não utiliza a criptografia ECC, somente a RSA. A Figura 15 e a Figura 16 apresentam a cópia dessa consulta e a respectiva resposta.

Figura 15 - Manifestação enviada para o ICP-Brasil

Teor da Manifestação	
Resumo:	Informações a respeito da certificação digital.
Extrato:	Boa tarde, estou fazendo meu TCC na área da computação e gostaria de saber como faço para conseguir dados a respeito das criptografias que protegem os certificados através de um script que use uma API e faça um requisição. Eu preciso saber dos 11.763.999 de certificados ativos, quais utilizam a criptografia RSA e ECC afim de compara-los. Obrigado

Fonte: Instituto Nacional de Tecnologia da Informação

Figura 16 - Resposta a manifestação enviado do ICP-Brasil

Dados das Respostas

Tipo de Resposta	Data/Hora	Teor da Resposta	Decisão
Resposta Conclusiva	27/03/2023 18:36	Prezado(a), As informações a respeito de criptografia podem ser encontradas no seguinte endereço: https://www.gov.br/iti/pt-br/assuntos/legislacao/documentos-principais/IN2022_22_DOC_ICP_01.01_assinado.pdf Informamos também que a ICP-Brasil não utiliza o padrão ECC, apenas RSA. Atenciosamente,	Acesso Concedido

Fonte: Instituto Nacional de Tecnologia da Informação

3.5 Tempo de criptografia e decriptografia para vários tamanhos de chave

Um aspecto fundamental na comparação entre o RSA e o ECC é o desempenho em relação ao tempo de criptografia e decriptografia. Neste contexto, foram criados *scripts* na linguagem Python e realizados testes utilizando os algoritmos RSA e ECC para medir os tempos necessários para realizar essas operações em diferentes tamanhos de chave e com mensagem de tamanho fixa.

Para a criptografia RSA, foram implementados testes com chaves de 1024 *bits*, 2048 *bits*, 3072 *bits* e 7680 *bits,* utilizando a equivalência de segurança conforme está dito no item 3.2 deste trabalho. A criptografia e decriptografia foram realizadas utilizando as chaves pública e privada geradas para cada tamanho de chave através da biblioteca *rsa* em Python. Os tempos de criptografia e decriptografia foram medidos utilizando a biblioteca *time* em Python. O *script* completo encontra-se no Apêndice 1.

Figura 17 - Trecho do código RSA variado o tamanho da chave e mensagem

```
1  import rsa
2  import time
3
4  # Gerar as chaves pública e privada
5  #1024, 2048, 3072, 7680
6  (chave_publica, chave_privada) = rsa.newkeys(2048)
7
8  # Criptografar uma mensagem
9  mensagem = "Ola, mundo.".encode()
10 tamanho_mensagem_bits = len(mensagem) * 8
11
12 t_inicio_criptografia = time.time() # Início do tempo de criptografia
13 mensagem_criptografada = rsa.encrypt(mensagem, chave_publica)
14 t_fim_criptografia = time.time() # Fim do tempo de criptografia
15 tempo_criptografia = t_fim_criptografia - t_inicio_criptografia
16
17 # Decriptografar a mensagem criptografada
18 t_inicio_descriptografia = time.time() # Início do tempo de descriptografia
19 mensagem_descriptografada = rsa.decrypt(mensagem_criptografada, chave_privada)
20 t_fim_descriptografia = time.time() # Fim do tempo de descriptografia
21 tempo_descriptografia = t_fim_descriptografia - t_inicio_descriptografia
```

Fonte: Autoria própria

Utilizando a biblioteca *rsa*, foram geradas as chaves pública e privada com um tamanho de acordo ao desejado, observa-se que a Figura 17 apresenta o tamanho de 1024 *bits* na linha 6, mas poderia ser o de 2048 *bits* ou outro tamanho.

Antes de criptografar a mensagem, foi necessário converter o texto em uma sequência de *bytes* utilizando o método *encode*(). Dessa forma, a mensagem pode ser processada corretamente pelos algoritmos de criptografia.

A criptografia da mensagem foi realizada utilizando a função *rsa.encrypt*(). Nesse processo, a mensagem original é criptografada usando a chave pública fornecida. O resultado é a mensagem criptografada, que não pode ser lida ou compreendida sem a chave privada correspondente.

Para determinar o tempo necessário para criptografar a mensagem, foram registrados o tempo de início e o tempo de fim utilizando a função *time.time*(). Ao calcular a diferença entre esses tempos, obtém-se o tempo de criptografia, que indica quanto tempo foi necessário para criptografar a mensagem.

A decriptografia foi realizada utilizando a função *rsa.decrypt*(). Essa função utiliza a chave privada correspondente à chave pública utilizada na criptografia para desfazer o processo de criptografia e obter a mensagem original.

Da mesma forma que na criptografia, foram registrados o tempo de início e de fim para medir o tempo de decriptografia, calculando-se a diferença entre esses tempos.

Os resultados obtidos para o algoritmo RSA revelaram uma relação entre o tamanho da chave e o tempo necessário para realizar as operações de criptografia e decriptografia. Em chaves de menor tamanho, como 1024 *bits*, observou-se um tempo insignificante para criptografar a mensagem, enquanto a decriptografia levou aproximadamente 5,00273704528808 ms. Conforme o tamanho da chave aumentou para 2048 *bits*, os tempos de criptografia e decriptografia aumentaram para aproximadamente 0,995874404907226 ms e 11,0101699829101 ms, respectivamente. Nos casos das chaves de 3072 *bits* e 7680 *bits*, o tempo de criptografia foi próximo de zero, enquanto o tempo de decriptografia aumentou significativamente para aproximadamente 33,9095592498779 ms e 363,069772720336 ms, respectivamente. Os resultados são exibidos na Tabela 2.

Tabela 2 - Tempo de criptografia e decriptografia RSA (ms)

Tamanho da chave (b*its*)	**Criptografia**	**Decriptografia**
1024	0	5,00273704528808
2048	0,995874404907226	11,0101699829101
3078	0	33,9095592498779
7680	2,98810005187988	363,069772720336

Fonte: Autoria própria

Para o a criptografia ECC, foram implementados testes utilizando chaves equivalentes aos tamanhos do RSA, os quais são chaves de 163 *bits*, 224 *bits*, 256 *bits* e 384 *bits*, utilizando a equivalência de segurança conforme está dito no item 3.2 deste trabalho. A criptografia e a decriptografia foi realizada utilizando as chaves pública e privada geradas para cada tamanho de chave através da biblioteca *cryptography* em Python. Os tempos de criptografia e decriptografia foram medidos utilizando a biblioteca *time* em Python.

Figura 18 - Trecho do código ECC variado o tamanho da chave e mensagem

```
1  from cryptography.hazmat.primitives.asymmetric import ec
2  from cryptography.hazmat.primitives.kdf.hkdf import HKDF
3  from cryptography.hazmat.primitives import hashes
4  from cryptography.hazmat.primitives.ciphers.aead import AESGCM
5  import os
6
7  # Gerar chave privada
8  #SECT163R2; SECP224R1; BrainpoolP256R1; BrainpoolP384R1; BrainpoolP512R1
9  private_key = ec.generate_private_key(ec.SECP224R1())
10
11 # Gerar chave pública
12 public_key = private_key.public_key()
13
14 # Mensagem a ser criptografada
15 mensagem = "Ola, mundo.".encode()
16 tamanho_mensagem_bits = len(mensagem) * 8
17
18 # Gerar chave simétrica
19 shared_key = private_key.exchange(ec.ECDH(), public_key)
20 hkdf = HKDF(algorithm=hashes.SHA256(), length=32, salt=None, info=b'handshake data',)
21 aes_key = hkdf.derive(shared_key)
```

Fonte: Autoria própria

Utilizando a biblioteca *cryptography*, foi gerada uma chave privada para o algoritmo ECC. Nesse caso, está sendo utilizado o algoritmo SECP224R1, no qual representa a chave de 224 *bits*, apresentado na linha 9 da Figura 18.

A partir da chave privada, foi obtida a chave pública correspondente. A chave pública é utilizada para criptografar a mensagem.

Antes de criptografar a mensagem, ela foi convertida para uma sequência de *bytes* utilizando o método *encode()*. Isso é necessário para que a mensagem possa ser processada corretamente pelos algoritmos de criptografia.

Utilizando o algoritmo de acordo de *chaves Elliptic curve Diffie Hellman*, Curva elíptica Diffie Hellman (ECDH) baseado no ECC, a chave privada é combinada com a chave pública para gerar uma chave compartilhada. Essa chave compartilhada é utilizada para derivar uma chave através do algoritmo *Hierarchical Key Derivation Function,* Função de Derivação de Chave Hierárquica (HKDF). A chave é necessária para criptografar e decriptografar a mensagem (MENEZES; VAN OORSCHOT; VANSTONE, 1996)

A criptografia da mensagem foi realizada utilizando o algoritmo *Advanced Encryption Standard - Galois/Counter Mode,* Padrão de Criptografia Avançada - Modo Galois/Contador (AES-GCM). Nesse processo, a chave derivada é utilizada para

criptografar a mensagem. Um valor aleatório chamado de *nonce* também é gerado para garantir a não repetição do criptograma (MENEZES et al., 1996).

A decriptografia da mensagem é realizada utilizando o mesmo algoritmo AES-GCM. A chave e o *nonce* são utilizados para decriptografar a mensagem e obter o texto original (MENEZES et al., 1996).

O algoritmo ECC apresentou eficiência notável em termos de tempo de criptografia e decriptografia, quando comparado com RSA. Utilizando chaves equivalentes em segurança aos tamanhos do RSA, os tempos foram significativamente mais curtos. Para chaves de 163 *bits*, 224 *bits*, 256 *bits* e 384 *bits*, o tempo de criptografia foi próximo de zero, e o tempo de decriptografia foi igual a zero, como observa-se na Tabela 3.

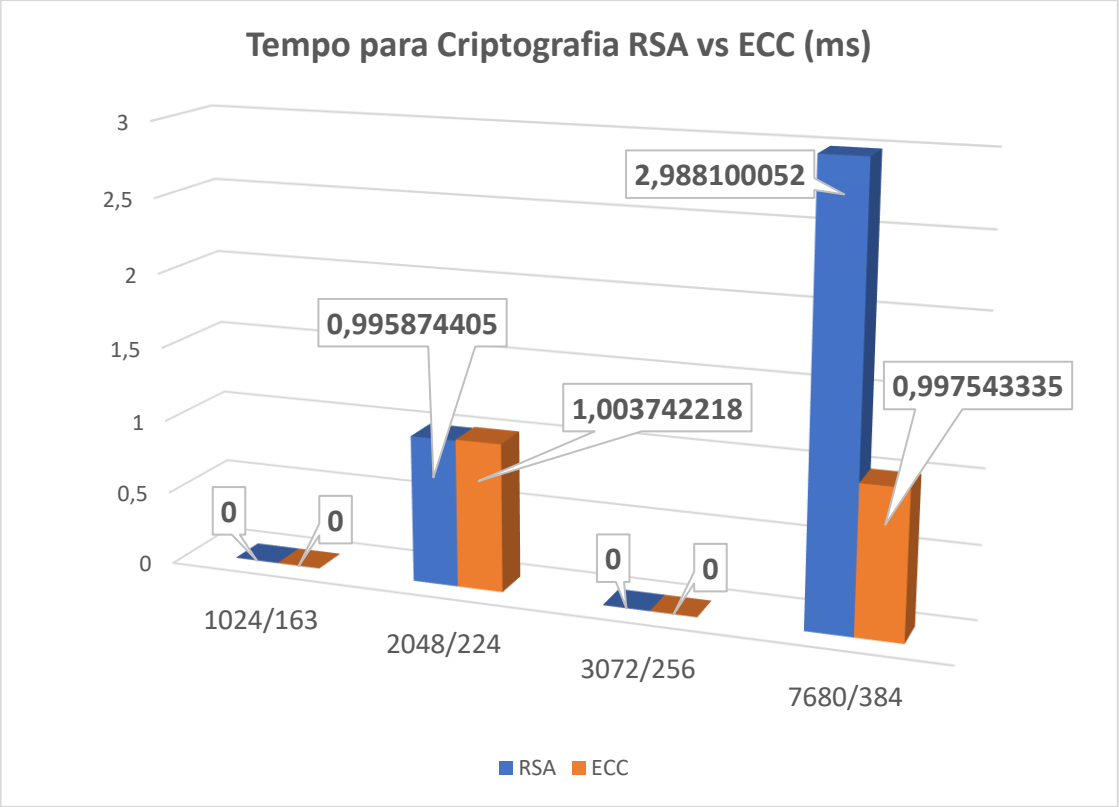
Tabela 3 - Tempo de criptografia e decriptografia ECC (ms)

Tamanho da chave (*bits*)	**Criptografia**	**Decriptografia**
163	0	0
224	1,00374221801757	0
256	0	0
384	0,997543334960937	0

Fonte: Autoria própria

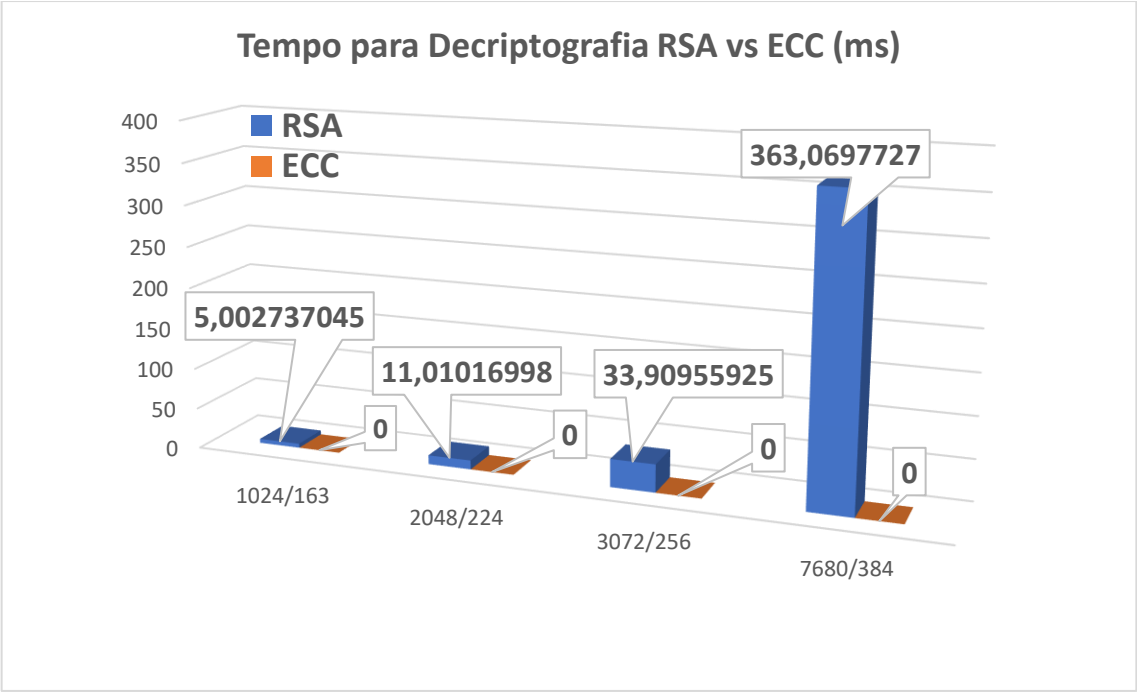
Esses resultados indicam claramente a vantagem do algoritmo ECC em termos de desempenho relacionado ao tempo, especialmente em cenários com chaves maiores. Enquanto o algoritmo RSA apresenta tempos mais elevados para a decriptografia, o ECC se destaca por sua eficiência e velocidade, mesmo com chaves de tamanhos equivalentes em segurança, como são vistos nos gráficos das Figuras 19 e 20.

Figura 19 - Tempo para criptografia RSA x ECC (ms)



Fonte: Autoria própria

Figura 20 - Tempo para decriptografia RSA x ECC (ms)



Fonte: Autoria própria

3.6 Tempo de criptografia e decriptografia para vários tamanhos de mensagem

O tempo de criptografia e decriptografia é um fator crucial a ser considerado ao avaliar a eficiência e desempenho dos algoritmos criptográficos. Em diversos contextos, o tempo de processamento necessário para criptografar e decriptografar mensagens pode ter um impacto direto na velocidade e eficiência de sistemas de segurança.

Nesta sessão, é abordada a influência do tamanho da mensagem no tempo de criptografia e decriptografia em relação ao algoritmo RSA e ECC com chaves de tamanho fixo de 2048 *bits* para o RSA e o seu equivalente que é 224 *bits* para o ECC, utilizando a equivalência de segurança conforme está dito no item 3.2 deste trabalho. Uma série de testes foram realizados utilizando mensagens com o tamanho de 512 *bits*, 1024 *bits*, 1536 *bits*, 1960 *bits* e 2048 *bits*, visando analisar como o aumento do tamanho da mensagem afeta o desempenho do algoritmo. A Tabela 4 contêm o tempo da criptografia e decriptografia do RSA.

Tabela 4 - Tempo de criptografia e decriptografia RSA (ms)

Tamanho da mensagem (*bits*)	**Criptografia**	**Decriptografia**
512	0	11,9679
1024	0,996351	12,0099
1536	0	13,0022
1960	0,99802	11,9827
2048	S/R	S/R

Fonte: Autoria própria

Como mostra na Tabela 4, esses resultados indicam que o tempo de criptografia e decriptografia aumentam à medida que o tamanho da mensagem cresce. No entanto, é interessante observar que nem sempre há uma correlação direta entre o tamanho da mensagem e o tempo de processamento. Em alguns casos, mensagens com tamanhos diferentes podem ter tempos semelhantes de criptografia e decriptografia. Além disso, na mensagem com tamanho de 2048 *bits*, quando executa o *script*, aparece um erro, houve um estouro na memória, uma vez que essa

chave só suporta mensagens de até 245 *bytes* ou então 1960 *bits*. Por isso que na Tabela 4, que deveria apresentar o tempo de criptografia e decriptografia para mensagens de 2048 *bits,* contêm um S/R, que significa sem resultado.

Com base nos dados coletados, foram analisados os tempos de criptografia e decriptografia para diferentes tamanhos de mensagem utilizando o algoritmo ECC. Os resultados obtidos estão na Tabela 5.

Tabela 5 - Tempo de criptografia e decriptografia ECC (ms)

Tamanho da mensagem (*bits*)	**Criptografia**	**Decriptografia**
512	0,997066	0
1024	0	0,999451
1536	0	1,008
1960	0	0
2048	0	0

Fonte: Autoria própria

Observa-se que, para todos os tamanhos de mensagem testados, o tempo de criptografia foi pequeno, geralmente próximo a 0 ms. Isso indica a eficiência do algoritmo ECC na criptografia de mensagens de diferentes tamanhos.

Além disso, o tempo de decriptografia também foi pequeno, com valores próximos a 0 ms. Isso significa que a decriptografia utilizando o algoritmo ECC é igualmente eficiente, independentemente do tamanho da mensagem. Por fim, diferente do RSA, o ECC não teve dificuldade para criptografar e decriptografar mensagens a partir 1960 *bits* ou 245 *bytes*.

3.7 Discussão dos resultados

A simulação foi realizada em um computador pessoal do autor deste trabalho, citado na item 3.1, utilizando o Visual Studio Code e a linguagem Python como ambiente de desenvolvimento. É importante ressaltar que os valores obtidos nessa simulação podem não refletir exatamente a realidade em diferentes cenários. Caso os testes sejam refeitos utilizando hardware ou software diferentes, é possível que os resultados variem.

No entanto, mesmo considerando essa ressalva, a comparação de desempenho entre o ECC e o RSA, levando em consideração o tamanho da chave e da mensagem, resultou em observações significativas. Notavelmente, o ECC demonstrou um desempenho superior em relação ao tempo necessário para operações de criptografia e decriptografia. Essa diferença de desempenho pode ser atribuída ao fato de que o ECC utiliza chaves criptográficas menores em comparação ao RSA para atingir o mesmo nível de segurança.

Portanto, mesmo levando em conta as particularidades da simulação realizada, os resultados obtidos sugerem que o ECC apresenta uma vantagem em termos de eficiência e desempenho em relação ao RSA, o que pode ser considerado um aspecto relevante ao avaliar a escolha do algoritmo de criptografia para utilizar na certificação digital.

4 CONSIDERAÇÕES FINAIS

O estudo comparativo entre os algoritmos RSA e ECC no contexto da criptografia de certificado digital apresentou importantes resultados. A análise considerou o tamanho das chaves, a difusão dos algoritmos no Brasil e globalmente, e o tempo de criptografia e decriptografia para diferentes tamanhos de chave e mensagem.

Em relação ao tamanho das chaves, verificou-se que o ECC requer chaves de tamanho menor em comparação com o RSA para obter o mesmo nível de segurança criptográfica. Isso permite o uso de uma criptografia mais robusta com menor carga de processamento, resultando em tempos mais rápidos de criptografia e decriptografia. Além disso, foi destacado que o RSA com chaves de 48 *bits* já foi quebrado por meio do algoritmo *Shor*, e no futuro a quebra de chaves de 2048 *bits* usando computadores quânticos é esperada. Por outro lado, o ECC também teve sua segurança comprometida devido ao *birthday attack*.

Em relação à difusão dos algoritmos, o RSA é o mais utilizado em todo o mundo, enquanto o ECC emerge como uma alternativa competitiva. No entanto, a complexidade dos cálculos matemáticos envolvidos na implementação do ECC tem sido um desafio para sua adoção generalizada. Além disso, sistemas que já utilizam o RSA podem não suportar a implementação do ECC, o que requer atualizações.

No contexto brasileiro, o órgão regulamentador das chaves públicas, ICP-Brasil, utiliza apenas o RSA, conforme informado em resposta a uma consulta realizada. O Brasil tem investido em transformação digital, mas a ineficiência nos serviços de processos digitais e autenticação tem gerado gastos significativos. Medidas como a MP 983/2020 foram implementadas para facilitar a autenticação e agilizar os atendimentos, especialmente durante a pandemia do COVID-19.

No que diz respeito ao tempo de criptografia e decriptografia, foi demonstrado a importância de considerar o desempenho como um fator crucial na escolha de um algoritmo criptográfico para aplicações de certificação digital. Os testes realizados demonstraram que o ECC apresenta tempos significativamente menores em comparação com o RSA. Mesmo com tamanhos de chave equivalentes, o ECC mostrou maior eficiência e velocidade, especialmente em cenários com chaves maiores. O aumento do tamanho da mensagem afetou o desempenho de ambos os

algoritmos, mas o ECC demonstrou uma eficiência notável em termos de tempo de processamento, mantendo tempos de criptografia e decriptografia próximos a zero.

No contexto da certificação digital, a consideração do tempo de criptografia e decriptografia é essencial para garantir a eficiência e a segurança das transações eletrônicas. A escolha do algoritmo criptográfico deve ser baseada em uma análise criteriosa dos requisitos específicos de cada aplicação, levando em consideração não apenas o desempenho, mas também outros fatores, como segurança, interoperabilidade e suporte a padrões.

Dessa forma, a comparação entre os tempos de criptografia e decriptografia do RSA e ECC destaca a relevância de selecionar o algoritmo mais adequado para cada contexto, considerando tanto os aspectos de segurança quanto as demandas de desempenho das aplicações de certificação digital.

Em resumo, a escolha entre o RSA e o ECC para a criptografia de certificado digital envolve considerações sobre o tamanho das chaves, a difusão dos algoritmos, a compatibilidade com sistemas existentes e o desempenho em relação ao tempo. O ECC oferece a vantagem de chaves menores e tempos de processamento mais rápidos, mas a adoção ainda é limitada devido a desafios técnicos e a preferência por chaves maiores por indivíduos que não tem domínio sobre o tema.

4.1 Sugestões de trabalhos futuros

- Avaliar outros algoritmos de criptografia, sejam eles simétricos ou assimétricos;
- Estudo sobre a segurança dos algoritmos RSA e ECC em relação a ataques específicos, como ataques de força bruta ou ataques de canal lateral;
- Investigação da adoção do ECC em setores específicos, como serviços financeiros, governo ou saúde.

REFERÊNCIAS

ANDRADE, M. **FGV: Brasil desperdiça R$ 104 bilhões com processos de identificação**. Metrópoles. Disponível em: <https://www.metropoles.com/brasil/fgv-brasil-desperdica-r-104-bilhoes-com-processos-de-identificacao>. Acesso em 03 maio 2023.

ANFIP**. Aplicativo Gov.br disponibiliza assinatura digital de documentos do poder público**. ANFIP. Disponível em: < https://www.anfip-sp.org.br/2022/05/aplicativo-gov-br-disponibiliza-assinatura-digital-de-documentos-do-poder-publico/>. Acesso em 03 maio 2023.

CARVALHO, Daniel. **Seguranca de Dados Com Criptografia Métodos e Algoritmos**. 2ª ed. São Paulo: Book Express, 2001.

CAVALCANTI, A. M. S. **Certificados digitais e programação - CRL e OCSP**. Linkedin. Disponível em: < https://pt.linkedin.com/pulse/certificados-digitais-e-programa%C3%A7%C3%A3o-crl-ocsp-s-cavalcanti>. Acesso em 27 de maio de 2023.

ENTRUST. **CRIPTOGRAFIA DE CURVA ELÍPTICA (ECC).** Entrust. Disponível em: < https://www.entrust.com/pt/resources/certificate-solutions/learn/elliptic-curve-cryptography-ssl>. Acesso em 29 nov. 2022.

FOROUZAN, Behrouz A. **Comunicação de dados e redes de computadores.** 4ª ed. Porto Alegre: AMGH, 2010.

GIMENES, D. **CERTIFICADO DIGITAL – O QUE É, QUAIS OS TIPOS E PARA QUE SERVE?**. DNA FiNANCEIRO. Disponível em: < https://dnafinanceiro.com/blog/certificado-digital-o-que-e-para-que-serve>. Acesso em 29 nov. 2022.

GOV.BR. **Assinatura Eletrônica do GOV.BR**. GOV.BR. Disponível em: < https://www.gov.br/governodigital/pt-br/assinatura-eletronica>. Acesso em 17 de maio 2023.

GOV.BR. **VISÃO GERAL SOBRE ASSINATURAS DIGITAIS NA ICP-BRASIL DOC-ICP-15 Versão 1.0**. GOV.BR. Disponível em: <https://www.gov.br/iti/pt-br/central-de-conteudo/doc-icp-15-v-1-0-pdf>. Acesso em 27 de maio 2023.

ICP BRASIL. **VISÃO GERAL SOBRE ASSINATURAS DIGITAIS NA ICP-BRASIL**. ICP BRASIL. Disponível em: <https://www.gov.br/iti/pt-br/central-de-conteudo/doc-icp-15-v-2-0-pdf>. Acesso em 30 nov. 2022.

ITI. **HISTÓRICO ANUAL DE EMISSÃO DE CERTIFICADOS.** ITI em números. Disponível em: <https://numeros.iti.gov.br/#/cert-visao-geral>. Acesso em 24 nov. 2022.

KAMINSKY, S. **Computadores quânticos quebrarão a criptografia RSA em 2023?.** Kaspersky. Disponível em: < https://www.kaspersky.com.br/blog/quantum-computers-and-rsa-2023/20551/>. Acesso em 08 maio 2023.

LSEC. **Criptus | Aula 5: Criptografia de Curvas Elípticas (ECC).** Youtube, 24 abr. 2016. Disponível em: <https://www.youtube.com/watch?v=AEVrpvaObdI>. Acesso em 30 nov. 2022.

MATOS, Luís. **SEGURANÇA.** Disponível em: <https://www.dicionariofmp-ifilnova.pt/wp-content/uploads/2019/07/Seguranca.pdf>. Acesso em: 01 nov. 2022.

MENEZES, A.; VAN OORSCHOT, P.; VANSTONE, S. M**anual de Criptografia Aplicada**. 1ª ed. Boca Raton: CRC Press, 1996.

MOZELLI, Rodrigo. **Brasil gasta bilhões para ter certeza que você é você; entenda.** Olhar digital. Disponível em: < https://olhardigital.com.br/2023/04/12/pro/governo-gasta-bilhoes-para-ter-certeza-que-voce-e-voce-entenda/>. Acesso em 17 maio 2023.

SANGALLI, L. A. **Criptossistemas baseados em curvas elípticas e seus desafios**. Disponível em: <https://www.dca.fee.unicamp.br/portugues/pesquisa/seminarios/2012/artigos/217.pdf>. Acesso em 03 maio 2023.

STALLINGS, W. **Segurança de Computadores.** 2ª ed. Rio de Janeiro: Elsevier, 2014.

SSL. **Comparando ECDSA x RSA. SSL**. Disponível em: < https://www.ssl.com/article/comparing-ecdsa-vs-rsa/>. Acesso em 03 maio 2023.

SSL. **O que é criptografia de curva elíptica (ECC)?**. SSL. Disponível em: < https://www.ssl.com/faqs/what-is-elliptic-curve-cryptography-ecc/>. Acesso em 03 maio 2023.

VERDÉLIO, A. **Governo normatiza assinatura eletrônica de documentos públicos**. Agência Brasil. Disponível em: <https://agenciabrasil.ebc.com.br/politica/noticia/2020-06/governo-normatiza-assinatura-eletronica-de-documentos-publicos> . Acesso em 03 maio 2023.

VICTOR. **O que é ICP-Brasil?** Infoco Certificação Digital. Disponível em: < https://www.infocodigital.com.br/icp-brasil/#:~:text=Ela%20foi%20criada%20por%20meio,empresas%20e%20pessoas%20f%C3%ADsicas%20envolvidas>. Acesso em 16 maio 2023.

WAZLAWICK, Raul. **Metodologia de Pesquisa para Ciência da Computação**. 2ª ed. Editora: ELSEVIER, Rio de Janeiro, 2014.

APÊNDICE A - CODIGO DO RSA EM PYTHON

```python
import rsa
import time

# Gerar as chaves pública e privada
#1024, 2048, 3072, 7680
(chave_publica, chave_privada) = rsa.newkeys(2048)

# Imprimir as chaves geradas
print("Chave Pública: ", chave_publica)
print("Chave Privada: ", chave_privada)

# Criptografar uma mensagem
# Esse "b" é um prefixo para tratar a string em bytes
mensagem = "Ola, mundo.".encode()
tamanho_mensagem_bits = len(mensagem) * 8

t_inicio_criptografia = time.time() # Início do tempo de criptografia
mensagem_criptografada = rsa.encrypt(mensagem, chave_publica)
t_fim_criptografia = time.time() # Fim do tempo de criptografia
tempo_criptografia = t_fim_criptografia - t_inicio_criptografia

# Imprimir a mensagem criptografada
#print("Mensagem Criptografada: ", mensagem_criptografada)
print("Tamanho da mensagem em bits:", tamanho_mensagem_bits)

# Decriptografar a mensagem criptografada
t_inicio_descriptografia = time.time() # Início do tempo de descriptografia
mensagem_descriptografada = rsa.decrypt(mensagem_criptografada, chave_privada)
t_fim_descriptografia = time.time() # Fim do tempo de descriptografia
tempo_descriptografia = t_fim_descriptografia - t_inicio_descriptografia

# Imprimir a mensagem decriptografada
print("Mensagem Descriptografada: ", mensagem_descriptografada)

# Imprimir o tempo de criptografia e decriptografia
print("Tempo de Criptografia: ", tempo_criptografia, "segundos")
print("Tempo de Decriptografia: ", tempo_descriptografia, "segundos")
```

APÊNDICE B - CODIGO DO ECC EM PYTHON

```python
import time
from cryptography.hazmat.primitives.asymmetric import ec
from cryptography.hazmat.primitives.kdf.hkdf import HKDF
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.ciphers.aead import AESGCM
import os

# Gerar chave privada
#SECT163R2; SECP224R1; BrainpoolP256R1; BrainpoolP384R1; BrainpoolP512R1
private_key = ec.generate_private_key(ec.SECP224R1())

# Gerar chave pública
public_key = private_key.public_key()

# Mensagem a ser criptografada
mensagem = "Ola, mundo.".encode()
tamanho_mensagem_bits = len(mensagem) * 8

# Gerar chave simétrica
shared_key = private_key.exchange(ec.ECDH(), public_key)
hkdf = HKDF(algorithm=hashes.SHA256(), length=32, salt=None, info=b'handshake data',)
aes_key = hkdf.derive(shared_key)

# Criptografando a mensagem
nonce = os.urandom(12)
cipher = AESGCM(aes_key)

# Medindo o tempo de criptografia
start_time = time.time()
ciphertext = cipher.encrypt(nonce, mensagem, None)
end_time = time.time()

# Tempo de criptografia
elapsed_time = end_time - start_time
print("Tempo de criptografia: ", elapsed_time)

# Decriptografando a mensagem
cipher = AESGCM(aes_key)

# Medindo o tempo de decriptografia
start_time = time.time()
decrypted_message = cipher.decrypt(nonce, ciphertext, None)
end_time = time.time()

# Tempo de decriptografia
elapsed_time = end_time - start_time
print("Tempo de decriptografia: ", elapsed_time)

# Imprimindo resultados
print("Chave privada:", private_key)
print("Chave pública:", public_key)
print("Mensagem original:", mensagem)
print("Mensagem criptografada:", ciphertext)
print("Mensagem decriptografada:", decrypted_message)
print("Tamanho da mensagem em bits:", tamanho_mensagem_bits)
```

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR

Av. Universitária, 1069 ● Setor Universitário
Caixa Postal 86 ● CEP 74605-010
Goiânia ● Goiás ● Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br ● reitoria@pucgoias.edu.br

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O(A) estudante João Alan Pimenta de Mello Ernesto do Curso de Engenharia de Computação, matrícula 20182003300811, telefone: (73) 999000506 e-mail joao_alan8@hotmail.com, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado COMPARAÇÃO ENTRE RSA E ECC NO ÂMBITO DA CERTIFICAÇÃO DIGITAL, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto (PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 05 de Abril de 2023

Assinatura do(s) autor(es): gov.br Documento assinado digitalmente
JOAO ALAN PIMENTA DE MELLO ERNESTO
Data: 05/04/2023 11:06:07-0300
Verifique em https://validar.iti.gov.br

Nome completo do autor: João Alan Pimenta de Mello Ernesto

Assinatura do professor-orientador:

Nome completo do professor-orientador: Angelica da Silva Nunes