INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO NORTE DE MINAS GERAIS

CAMPUS JANUÁRIA

CURSO SUPERIOR DE LICENCIATURA EM MATEMÁTICA

TRABALHO DE CONCLUSÃO DE CURSO

DESVENDANDO A CRIPTOGRAFIA DE CURVAS ELÍPTICAS: PROPRIEDADES, MÉTODOS E IMPLEMENTAÇÃO

ISAK PAULO DE ANDRADE RUAS

JANUÁRIA (MG)
2024

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO NORTE DE MINAS GERAIS

CAMPUS JANUÁRIA

CURSO SUPERIOR DE LICENCIATURA EM MATEMÁTICA

DESVENDANDO A CRIPTOGRAFIA DE CURVAS ELÍPTICAS: PROPRIEDADES, MÉTODOS E IMPLEMENTAÇÃO

ISAK PAULO DE ANDRADE RUAS

Sob a orientação do Professora
Me. Celimar Reijane Alves Damasceno Paiva

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia do Norte de Minas Gerais – Campus Januária como parte das exigências do Programa de Graduação em Matemática, para obtenção do título de Licenciado em Matemática.

JANUÁRIA, MG
Março de 2024

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO NORTE DE MINAS GERAIS

CAMPUS JANUÁRIA

DESVENDANDO A CRIPTOGRAFIA DE CURVAS ELÍPTICAS PROPRIEDADES, MÉTODOS E IMPLEMENTAÇÃO

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia do Norte de Minas Gerais – Campus Januária como parte das exigências do Programa de Graduação em Matemática, para obtenção do título de Licenciado em Matemática.

APROVADO: ____/____/_____

Me. Samuel Chaves Dias

Me. Fernando Marcos Souza Silva

Me. Celimar Reijane Alves Damasceno Paiva
(Orientadora)

LISTA DE FIGURAS

LISTA DE TABELAS

Ao ímpeto curioso que me guia na jornada de desvendar os segredos do universo.

AGRADECIMENTOS

Gostaria de expressar minha sincera gratidão a todos aqueles que tornaram este trabalho possível. Ao longo desta jornada, recebi um imenso apoio e incentivo que foram fundamentais para alcançar este marco.

Em primeiro lugar, devo expressar minha profunda gratidão às duas mulheres mais importantes da minha vida: minha mãe e minha noiva Sibele. A minha mãe sempre proporcionou amor incondicional e apoio constante, sendo uma inspiração para me esforçar constantemente em ser o melhor que posso ser. A Sibele, minha noiva, é a minha rocha emocional, tornando cada desafio mais fácil de enfrentar com seu constante incentivo. Agradeço a vocês duas pelo amor e compreensão.

Também sou grato à minha orientadora, a Professora Celimar, cuja dedicação e conhecimento foram inestimáveis para a realização deste trabalho. O seu constante apoio e conselhos orientadores foram de valor inestimável.

Estendo meus sinceros agradecimentos aos meus companheiros militares, Tenente Almeida, Tenente Erasmo, Tenente João Carlos, Sargento Virginia e Santos. A presença e apoio de vocês foram uma constante fonte de inspiração. Sou igualmente grato a Aline, assistente social, e Dallila, psicóloga, cujo apoio emocional me ajudou a equilibrar a vida pessoal e a jornada acadêmica.

Gostaria de agradecer aos professores Egídio, Everton, Josué e Gustavo, cujas aulas e orientações tiveram um impacto direto na qualidade deste trabalho. Agradeço também ao apoiadores, especialmente Breno Coe, cujo incentivo ao estudo me guiou ao longo de toda a minha jornada acadêmica.

Por fim, mas não menos importante, gostaria de expressar minha profunda gratidão ao Prof. Josué, que esteve comigo nos momentos mais difíceis. Sou eternamente grato pela sua amizade e apoio.

A todos que, de alguma maneira, contribuíram para a concretização deste projeto, estendo minha mais profunda gratidão. Mesmo aqueles que não foram mencionados aqui, tiveram uma contribuição essencial para esta conquista. Este trabalho é o resultado da colaboração e esforço de todos vocês. Obrigado!

"Matemática é a ferramenta que permite à criptografia transformar informações em segredos e segredos em informações."

Autor Desconhecido

RESUMO

RUAS, Isak Paulo de Andrade. **Desvendando a Criptografia de Curvas Elípticas:** Propriedades, Métodos e Implementação. 2024. Monografia (Graduação em Licenciatura em Matemática) Instituto Federal de Educação, Ciência e Tecnologia do Norte de Minas Gerais – Campus Januária, Januária, 2024.

Esta pesquisa foca em desvendar a criptografia de curvas elípticas, explorando suas propriedades fundamentais, os métodos para codificação e decodificação e suas possíveis implementações. Especial atenção é dada à aplicação dessas curvas na criptografia de ponta a ponta, com o intuito de aprofundar a compreensão dos mecanismos que garantem a segurança da comunicação digital.

Palavras-chave: criptografia de curvas elípticas. codificação e decodificação. implementação de criptografia.

Orientadora: Me. Celimar Reijane Alves Damasceno Paiva
Co-Orientador: Me. Fernando Marcos Souza Silva

SUMÁRIO

1 INTRODUÇÃO

Esta pesquisa foi motivada pela necessidade de entender a estrutura e funcionamento da criptografia de ponta a ponta no contexto digital. Questões como: *"Como funciona a criptografia de ponta a ponta?"*, e *"Como podemos assegurar a autenticidade do remetente e a integridade da mensagem?"* serviram como base para o desenvolvimento deste estudo.

A criptografia é uma ferramenta indispensável na garantia da privacidade e segurança das comunicações no mundo digital. No entanto, o desenvolvimento e implementação de sistemas de criptografia eficientes e seguros ainda consiste em um desafio para muitos desenvolvedores e organizações. Sistemas de criptografia, como os baseados em curvas elípticas, possuem uma complexidade matemática intrínseca que pode tornar seu entendimento e implementação um processo complexo.

O objetivo desta pesquisa é desmistificar o funcionamento da criptografia baseada em curvas elípticas e fornecer uma implementação passo a passo deste sistema. Este estudo é relevante pois pode servir de guia para desenvolvedores e pesquisadores que necessitam compreender a matemática por trás deste tipo de criptografia e saber como aplicá-la em situações práticas.

Além disso, considerando o papel primordial da criptografia nos dias atuais para a proteção de dados e garantia da privacidade no ambiente digital, este trabalho também tem como propósito contribuir para a literatura na área de segurança cibernética e criptografia. A pesquisa vai além de apenas apresentar fundamentos teóricos, propondo também uma implementação prática que possa ser facilmente reproduzida por outros profissionais da área.

A criptografia de curva elíptica tem se destacado como uma alternativa eficiente em relação a outras técnicas, devido ao seu desempenho consideravelmente superior com chaves de tamanho reduzido. Dessa forma, um estudo aprofundado sobre o uso de curvas elípticas na criptografia traz uma contribuição significativa tanto para a comunidade acadêmica quanto para a indústria.

Finalmente, esta investigação parte de um questionamento genuíno e uma busca por um entendimento técnico e prático dos conceitos que governam as aplicações de segurança digital. A busca por respostas a estas perguntas possibilita uma compreensão mais aprofundada dos sistemas criptográficos e permite uma análise mais crítica do cenário atual da segurança de dados.

2 PANORAMA HISTÓRICO DA CRIPTOGRAFIA

A criptografia é uma ferramenta indispensável para a proteção de informações. Derivada do grego "kriptos", que significa "secreto", e "graphia", que se refere à "escrita", é uma ferramenta indispensável para a proteção de informações (Pabón Cadavid, 2010, p. 59) ou (Filho e Azeredo, 2017, p. 23). Envolve a utilização de algoritmos para transformar a informação original em um formato indetectável e inacessível a indivíduos sem autorização. Ela desempenha um papel crucial na garantia da segurança da transmissão de informações, especialmente em redes ou ambientes considerados inseguros (Viana *et al*, 2022, p. 226).

Com sua origem na antiguidade, a criptografia já era utilizada em situações militares complexas, conforme relatados Heródoto; destacou-se ainda mais sobre figuras históricas como Júlio César, que utilizou técnicas esteganográficas e criptográficas em suas comunicações para garantir a confidencialidade das informações. A introdução dos sistemas elétricos de transmissão de informações impulsionou a criptografia para uma nova trajetória, tornando-se o precursora do que conhecemos hoje (Pabón Cadavid, 2010, p. 61-63).

2.1 Origem e desenvolvimento da criptografia

A era moderna, especialmente após a Segunda Guerra Mundial, trouxe avanços tecnológicos notáveis na criptografia. Durante esse período, houve uma expansão significativa na pesquisa e desenvolvimento em criptografia, o que despertou não apenas o interesse das forças armadas e dos governos, mas também do público em geral. Com o crescimento da pesquisa e do desenvolvimento no campo, houve a criação de sistemas computacionais avançados para fins de criptografia, destacando-se como ferramentas essenciais para a segurança da informação (Pabón Cadavid, 2010, p. 64).

Neste cenário, grandes descobertas, como a máquina de Charles Babbage e as bombas de Turing de Alan Turing, desempenharam um papel crucial. Essas invenções, originalmente voltadas para o desciframento de códigos, se tornaram fundamentais para o desenvolvimento da tecnologia informática atual. É importante ressaltar que sem esses desenvolvimentos pioneiros na criptografia, a revolução da tecnologia da informação poderia ter tomado um curso bastante diferente, evidenciando o papel crucial da criptografia na evolução da tecnologia (Pabón Cadavid, 2010, p. 64).

A criptografia moderna, embora mantenha a essência de sua prática histórica, se tornou amplamente complexa devido ao avanço tecnológico. Sua execução não se limita mais à simples reorganização do alfabeto, mas incorpora algoritmos matemáticos complicados para cifrar mensagens. Da mesma forma, a tarefa de "quebrar"códigos criptográficos, a criptoanálise, sai da capacidade humana de observar padrões de repetição de letras, tornando-se dependente da potência de supercomputadores. Dessa forma, se torna quase impossível criar

códigos seguros e descriptografá-los sem um sólido conhecimento em matemática, considerável poder de processamento de dados e um substancial investimento financeiro (Abreu, 2017, p. 28)

2.2 Adaptação da criptografia ao mundo moderno

No mundo contemporâneo, cada vez mais imerso em digitalização e conectividade, não podemos ignorar a preponderância e a importância vital da criptografia. Há uma presença profunda e abrangente da tecnologia em nossas vidas cotidianas, desde comunicação pessoal, transações financeiras até a infraestrutura crucial de qualquer nação. Tudo isso cria um cenário onde a segurança dos dados online se torna não apenas desejável, mas uma absoluta necessidade.

Nesse cenário, a criptografia emerge como um componente soberano na garantia da segurança e privacidade do usuário na internet. Este foco elevado na criptografia é compreensível considerando que ela proporciona um canal seguro para troca de informações delicadas nesse mar aberto e incerto que é a web (Viana *et al*, 2022, p. 231-236). A criptografia não apenas permeia o tecido de nossas interações online como indivíduos, mas também desempenha um papel crucial para entidades empresariais e governamentais. Seu alcance não se restringe somente a aplicativos e websites; na verdade, ela é de vital importância para sistemas mais complexos e intrincados.

É nesses sistemas que evidenciamos sua relevância no setor privado. As trocas de informações confidenciais são recorrentes nas interações empresariais, e a criptografia se torna um escudo protetor, guardião da integridade dessas informações. Além disso, situações de hostilidade, como períodos de guerra ou conflito, reforçam ainda mais a necessidade de mensagens codificadas para garantir a segurança e evitar deturpações (Andria; Gondim; Salomão, 2019, p. 31-36). Por outro lado, as transações financeiras se tornaram primariamente digitais devido ao conveniente avanço tecnológico. Essa transição para o meio digital das transações financeiras, junto com atividades comerciais e assuntos corporativos sensíveis, certamente exigem um nível de confidencialidade que somente a criptografia pode providenciar adequadamente.

Cada dia mais pessoas estão se deliciando com a comodidade que a internet proporciona, sem se dar conta de que cada transação, cada troca de dados seguros, é facilitada e resguardada por diversos sistemas de criptografia que trabalham incansavelmente em segundo plano. Por esse motivo, podemos afirmar sem hesitação que a criptografia não é apenas um luxo, mas uma necessidade crucial que proporciona a segurança e confiança tão necessárias na atual era digital (Andria; Gondim; Salomão, 2019, p. 31-36). Com tudo isso em mente, fica claro que sem a presença vital da criptografia, o ciberespaço atual seria um lugar muito mais perigoso e imprevisível de se navegar.

2.3 Os impasses legais na utilização da criptografia

Na era da economia digital avançada e da transformação digital disseminada na vida cotidiana, a demanda por sistemas seguros e confiáveis está em progressivo crescimento. Ao atender essas demandas, a criptografia emerge como uma protagonista estratégica. Ela atua como uma poderosa ferramenta na salvaguarda de dados, assegurando que as informações transmitidas permaneçam inacessíveis a olhares indesejados (Viana *et al*, 2022, p. 231-232). No entanto, lançamos nosso olhar a um peculiar dilema quando o 'olhar indesejado' que tenta acessar essas informações se identifica como o próprio Estado.

A capacidade da criptografia em manter os dados impenetráveis e invioláveis frequentemente se torna o núcleo dos debates legais entre as grandes empresas tecnológicas e os governos. Estas discussões centram-se na dificuldade ou, em muitos casos, na impossibilidade de se violar a privacidade que esses sistemas criptográficos garantem. Podemos observar esse conflito através de casos emblemáticos, como o do aplicativo de mensagens instantâneas WhatsApp, bloqueado três vezes no Brasil por se recusar a fornecer o conteúdo de comunicações dos investigados em processos criminais (Abreu, 2017, p. 28).

Seguindo essa mesma linha, podemos revisitar o conflito de 2015 entre a Apple e o governo dos EUA. A face a face de discordâncias aconteceu por causa dos robustos mecanismos de criptografia utilizados nos iPhone, considerados invioláveis pelo governo. Este evento sublinha a tensão inerente na relação entre os Estados e a criptografia: é uma ferramenta que uso, mas rejeito seu uso por outros, especialmente quando esses "outros"são considerados potenciais adversários (Abreu, 2017, p. 28). Essa dualidade mostra a complexidade da relação entre segurança, privacidade e governança na era digital.

2.4 Criptografia: um olhar para o futuro

Portanto, a importância da criptografia não é uma questão só do passado ou do presente. Sua relevância se perpetua ao longo do tempo, evoluindo constante e adaptativamente para atender necessidades emergentes de segurança e para sustentar a confiabilidade na digitalização crescente (Viana *et al*, 2022, p. 234). A habilidade de cifrar mensagens, permitindo uma comunicação segura e privada, volta-se cada vez mais indispensável em uma sociedade que é, ao mesmo tempo, altamente digital e globalmente interconectada.

A criptografia, cujo papel tem sido de relevância para a humanidade desde os tempos antigos, certamente continuará a ser uma peça chave no nosso futuro digital. Por garantir a privacidade e a segurança das informações, a criptografia é uma ferramenta crucial para proteger dados em um mundo que se torna cada dia mais dependente da tecnologia e da digitalização. Sua relevância, longe de diminuir, só tem a crescer e se adaptar para atender às novas demandas e necessidades de segurança que irão surgir.

3 FUNDAMENTOS E CARACTERÍSTICAS DA CRIPTOGRAFIA

3.1 Entendendo a criptografia simétrica

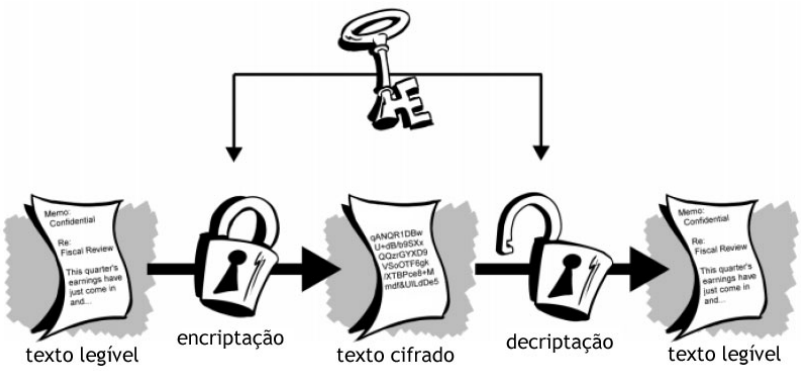

Figura 1 – Ilustração do processo de criptografia Simétrico
Fonte: Seragiotto, 2023, p. ?

A criptografia simétrica, de processo ilustrado na Figura 1, é um dos principais métodos de criptografia, onde é usada uma única chave tanto para criptografar quanto para descriptografar os dados. Este sistema é eficiente e rápido, onde o remetente utiliza a chave para transformar a mensagem em um formato ilegível, sendo então revertida no lado do destinatário usando a mesma chave (Viana *et al*, 2022, p. 235-236)

> Essencialmente, quando a origem (ALFA) cifra uma mensagem, ele utiliza um algoritmo de ciframento para transformar o conteúdo em claro da mensagem em texto cifrado. Quando o destino (BRAVO) decifra uma mensagem, ele utiliza o algoritmo de deciframento correspondente para converter o texto cifrado de novo em uma mensagem clara. Se um intruso (CHARLIE) conhecer o algoritmo de ciframento, ele poderia decifrar uma mensagem cifrada tão facilmente quanto o destino (BRAVO). A solução no uso da criptografia de chave privada propõe que quando a origem (ALFA) cifra uma mensagem, ele utilize um algoritmo de ciframento e uma chave secreta para transformar uma mensagem clara em um texto cifrado. O destino (BRAVO), por sua vez, ao decifrar a mensagem, utiliza o algoritmo de deciframento correspondente e a mesma chave para transformar o texto cifrado em uma mensagem em claro. O intruso (CHARLIE), por não possuir a chave secreta, mesmo conhecendo o algoritmo, não conseguirá decifrar a mensagem. A segurança do sistema passa a residir não mais no algoritmo e sim na chave empregada. É ela (chave privada) que agora, no lugar do algoritmo, deverá ser mantida em segredo pela origem (ALFA) e destino (BRAVO) (Oliveira, 2012, p. 3)

Entretanto, um desafio crítico ao utilizar esse sistema é o compartilhamento seguro da chave que deve ser efetuado entre o remetente e o destinatário. Este processo precisa ser seguro para evitar que indivíduos mal-intencionados interceptem a chave e tenham acesso à informação enviada (Viana *et al*, 2022, p. 236)

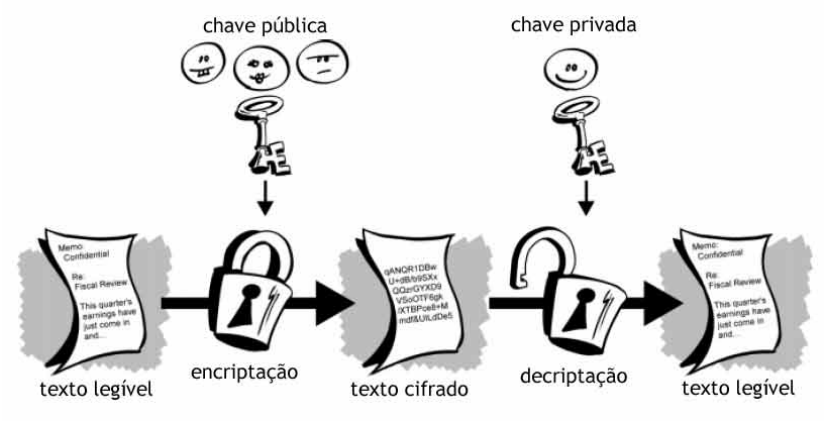

Figura 2 – Ilustração do processo de criptografia Assimétrico
Fonte: Seragiotto, 2023, p. ?

3.2 Entendendo a criptografia assimétrica

Contrastando com a criptografia simétrica, temos a criptografia assimétrica, de processo ilustrado na Figura 2, também conhecida como criptografia de chave pública. Aqui, um par de chaves é usado, sendo elas a chave pública, que pode ser compartilhada abertamente, e uma chave privada, que deve ser mantida em sigilo somente pelo destinatário da mensagem (Viana *et al*, 2022, p. 238)

> Essencialmente, o destino (BRAVO) e todos os que desejam comunicar-se de modo seguro geram uma chave de ciframento e sua correspondente chave de deciframento. Ele mantém secreta a chave de deciframento, esta é chamada de sua chave privada. Ele torna pública a chave de ciframento, esta é chamada de sua chave pública. A chave pública realmente condiz com seu nome. Qualquer pessoa pode obter uma cópia dela. O destino (BRAVO) inclusive encoraja isto, enviando-a para seus amigos ou publicando-a na internet. Assim, O intruso (CHARLIE) não tem nenhuma dificuldade em obtê-la. Quando a origem (ALFA) deseja enviar uma mensagem ao destino (BRAVO), precisa primeiro encontrar a chave pública dele. Feito isto, ela cifra sua mensagem utilizando a chave pública do destino (BRAVO), despachando-a em seguida. Quando o destino (BRAVO) recebe a mensagem, ele a decifra facilmente com sua chave privada. O intruso (CHARLIE), que interceptou a mensagem em trânsito, não conhece a chave privada do destino (BRAVO), embora conheça sua chave pública. Mas este conhecimento não o ajuda a decifrar a mensagem. Mesmo a origem (ALFA), que foi quem cifrou a mensagem com a chave pública do destino (BRAVO), não pode decifrá-la agora. (Oliveira, 2012, p. 4)

Para enviar uma mensagem utilizando criptografia assimétrica, a chave pública do destinatário é usada pelo remetente para criptografar a mensagem. Esta só pode ser decriptada e lida pelo destinatário que possui a chave privada correspondente. Isso adiciona uma camada extra de segurança à mensagem, pois a chave privada é exclusiva do destinatário (Viana *et al*, 2022, p. 240)

3.3 Contrapontos entre criptografia simétrica e assimétrica

A criptografia simétrica e a criptografia assimétrica possuem diferenças essenciais que ditam suas aplicações ideais. A criptografia simétrica é rápida e eficiente, tornando-a ideal para a codificação de conteúdos de mensagens. Isso garante a confidencialidade de dados no processo de transmissão. No entanto, as abordagens simétricas encontram suas principais desvantagens na complexidade do gerenciamento e distribuição de chaves (Oliveira, 2012, p. 8).

Gerenciar e distribuir chaves é um desafio na criptografia simétrica, pois exige um canal seguro para o compartilhamento das chaves entre emissor e receptor. Esse processo se torna complicado, especialmente quando há um grande número de participantes na transmissão de dados. Isso torna a gerência das chaves extremamente difícil e vulnerável a falhas (Oliveira, 2012, p. 8).

Em contrapartida, a criptografia assimétrica resolve os problemas de gerenciamento e distribuição de chaves que são inerentes à criptografia simétrica. Embora seja mais lenta devido ao seu alto grau de complexidade, ela fornece um nível maior de segurança. Isso ocorre através da separação das chaves em públicas e privadas (Oliveira, 2012, p. 8).

Na criptografia assimétrica, a chave pública pode ser distribuída livremente. Enquanto isso, a chave privada, utilizada para decodificar o documento codificado, é conhecida apenas pelo seu proprietário. Além disso, este tipo de criptografia é ideal para a distribuição de chaves e para assinatura digital. Isso garante não apenas a confidencialidade, mas também a autenticidade e a integridade das mensagens (Oliveira, 2012, p. 8).

Ambas os sistemas de criptografia possuem suas vantagens e desvantagens. Enquanto a criptografia simétrica é rápida e eficiente, a segurança do compartilhamento de chaves é um desafio. Por outro lado, a criptografia assimétrica fornece uma segurança extra com a separação de chaves em pública e privada, mas pode ser mais lenta devido ao seu maior nível de complexidade. A escolha do tipo de criptografia a ser usado depende das necessidades específicas de segurança de cada situação.

4 METODOLOGIA

A metodologia desta pesquisa foi estruturada em quatro etapas distintas:

Na primeira etapa, realizou-se uma revisão bibliográfica abrangente, na qual se analisaram e interpretaram trabalhos científicos, monografias, teses, dissertações e normas de instituições renomadas, como o National Institute of Standards and Technology (NIST) dos EUA. O foco principal dessa revisão foi a investigação da criptografia de ponta a ponta, curvas elípticas, corpos finitos e os fundamentos matemáticos dos algoritmos de criptografia. Essa revisão bibliográfica proporcionou uma sólida base teórica para o desenvolvimento das etapas subsequentes do projeto.

Na segunda etapa, exploraram-se as propriedades dos algoritmos de criptografia de ponta a ponta, com um enfoque particular naqueles baseados em curvas elípticas. A análise concentrou-se em questões como a codificação e decodificação de mensagens, o estabelecimento de chaves seguras, a troca de informações e a assinatura digital. Essa análise teórica permitiu uma compreensão mais aprofundada das características e dos desafios associados à criptografia.

A terceira etapa envolveu o desenvolvimento de um *software* simulador, utilizando a linguagem de programação Python como plataforma de implementação. A escolha do Python deveu-se à sua robustez e acessibilidade em temas de segurança da informação e criptografia. Essa etapa transformou a teoria em prática, permitindo a criação de um ambiente de simulação para os algoritmos de criptografia estudados.

A quarta etapa contemplou a análise dos resultados gerados pelo *software* implementado. Realizaram-se testes de eficácia, eficiência e confiabilidade dos algoritmos e protocolos utilizados. Durante essa fase, observações e limitações identificadas nos testes foram cuidadosamente documentadas e analisadas. Essa avaliação experimental forneceu insights valiosos sobre o desempenho dos algoritmos em um ambiente prático.

5 CRIPTOGRAFIA COM CURVAS ELÍPTICAS

5.1 Conceitos preliminares

Neste trabalho, adotaremos como definição equação de Weierstrass simplificada de uma curva elíptica sobre um corpo $\mathbb{K}$, resultado apresentado por Gonzaga e Pesco (2021, p. 37), $i.e$

Definição 5.1.1. Uma curva elíptica $E = \{(x, y) \in \mathbb{K} | (y^2 = x^3 + Ax + B)\}$ no qual $(car(\mathbb{K}) \notin \{2, 3\})$ e $(4A^3 + 27B^2 \neq 0)$.

$4A^3 + 27B^2 \neq 0$ é a condição necessária para que qualquer ponto $P \in E$ possua uma reta tangente (seja diferenciável) e $car(\mathbb{K}) \notin \{2, 3\}$ garante que haja inverso multiplicativo de 2 e 3 em $\mathbb{K}$. Para uma implementação computacional mais eficiente (Gonzaga e Pesco, 2021, p. 43), adotaremos $\mathbb{K}$ como o corpo dos inteiros módulo p, ou seja, adotaremos $\mathbb{K} = \mathbb{Z}_p$, desta forma, a definição anterior poderá ser escrita como $E(\mathbb{Z}_p) : y^2 \equiv x^3 + Ax + B \pmod p$, donde $4A^3 + 27B^2 \neq 0 \pmod p$ (Andria; Gondim; Salomão, 2019, p. 82) De maneira geral, é valida a proposição a seguir:

Proposição 5.1.1. Considere a curva elíptica E. O conjunto de pontos na curva sobre um corpo finito $\mathbb{F}_p$, $E(\mathbb{F}_p)$, é um grupo abeliano finito.

Demonstração. Pode ser vista em Brady, Davis e Tracy (2010, p. 6-7). □

Exemplo 5.1.1. Seja E uma curva elíptica sobre $(\mathbb{Z}_7, \oplus)$ de equação $y^2 = x^3 + x + 1$, determine todos os pontos desta curva.

Para encontrar a solução deste exemplo, podemos empregar a seguinte estratégia: criaremos uma tabela para calcular os valores de x e y separadamente e, em seguida, para cada valor de $x \in \mathbb{Z}_7$, verificaremos se o resultado da expressão $x^3 + x + 1$ é um quadrado perfeito. As tabelas a seguir apresentam os resultados dessas operações:

Tabela 1 – Lado esquerdo da equação $y^2 = x^3 + x + 1$: atribuindo valores à variável y

y	0	1	2	3	4	5	6
y^2	0	1	4	2	2	4	1

Fonte: —-

Deste modo, substituindo $x = 0$ no polinômio, obtemos 1, observando-se a coluna y^2, temos que $y = 1$ e $y = 6$ geram $y^2 = 1$, assim, temos que os pares ordenados formados são $(\overline{0}, \overline{1})$ e $(\overline{0}, \overline{6})$. Realizando este procedimento para todas os valores de x, encontramos: $E(\mathbb{Z}_7) = \{\mathcal{O}, (\overline{0}, \overline{1}), (\overline{0}, \overline{6}), (\overline{2}, \overline{2}), (\overline{2}, \overline{5})\}$. $\mathcal{O}$ é o elemento neutro de $(\mathbb{Z}_7, \oplus)$, este ponto também conhecido como *"ponto no infinito"*, sua existência é garantida pelo Teorema 5.1.1.

Tabela 2 – Lado direito da equação $y^2 = x^3 + x + 1$: atribuindo valores à variável x

x	0	1	2	3	4	5	6
$x^3 + x + 1$	1	3	4	3	6	5	6

Fonte: —-

Teorema 5.1.1. A curva elíptica E, com a estrutura de soma $\oplus$, é um grupo abeliano cujo elemento neutro é $\mathcal{O}$.

Demonstração. Pode ser vista em Silva, Salomão e Neves (2019, p. 85). □

Nesta sessão, apresentou-se os conceitos preliminares fundamentais para compreensão das curvas elípticas, no qual observou-se sua lei de formação e o método a ser empregado para se determinar os pontos da curva em $\mathbb{Z}_p$, na próxima seção apresentaremos o método a ser utilizado para construir a tabua de operação de $(\mathbb{Z}_p, \oplus)$.

5.2 Algoritmo para adição de pontos em curvas elípticas

Teorema 5.2.1. (Algoritmo Euclidiano Estendido - Lema de Bézout) Sejam a e b inteiros positivos. Então a equação nas variáveis X e Y:

$$aX + bY = mdc(a, b) \tag{1}$$

Possui soluções inteiras, digamos $X = u_0$ e $Y = v_0$. Além disso, todas as suas soluções são da forma:

$$X = u_0 + \frac{bk}{mdc(a, b)} \tag{2}$$

$$Y = v_0 - \frac{ak}{mdc(a, b)} \tag{3}$$

com $k \in \mathbb{Z}$.

Demonstração. Pode ser vista em Silva, Salomão e Neves (2019, p. 12-13). □

Definição 5.2.1. Dado uma curva elíptica E definida sobre $(\mathbb{Z}_p, \oplus)$ e os pontos $P = (x_1, y_1)$ e $Q = (x_2, y_2) \in E$, a soma $P \oplus Q$ resulta em $R = (x_3, y_3) \in E$ (consulte o Teorema 5.1.1). A operação $\oplus$ em E é definida da seguinte maneira:

Se $P = Q$, calculamos o valor de λ da seguinte forma:

$$\lambda = \frac{3x_1^2 + A}{2y_1} \pmod{p} \tag{4}$$

Se $P \neq Q$, o valor de λ é calculado como:

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \pmod{p} \tag{5}$$

Com base no valor de λ, determinamos as coordenadas do ponto R como segue:

$$x_3 = \lambda^2 - x_1 - x_2 \pmod{p} \tag{6}$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod{p} \tag{7}$$

Exemplo 5.2.1. Considere a curva elíptica E definida sobre $(\mathbb{Z}_7, \oplus)$ com a equação $y^2 = x^3 + x + 1$, calcule a soma de $P = (\overline{2}, \overline{5})$ com $Q = (\overline{2}, \overline{5})$

Devido a $P = Q$, aplicamos a equação (4). Assim, temos:

$$\begin{aligned} \lambda &= \frac{3 \cdot 2^2 + 1}{2 \cdot 5} \pmod{7} \\ &= \frac{13}{10} \pmod{7} \\ &= 13 \cdot 10^{-1} \pmod{7} \end{aligned} \tag{8}$$

Para calcular λ, precisamos resolver a equação diofantina (1):

$$10x + 7y = mdc(10, 7) = 1$$

Vamos resolver passo a passo:

$$10 = 7 \cdot 1 + 3 \tag{9}$$

$$7 = 3 \cdot 2 + 1 \tag{10}$$

Agora, podemos usar os resultados de (9) e (10) para encontrar x e y:

$$3 = 10 - 7 \cdot 1 \tag{11}$$

$$1 = 7 - 3 \cdot 2 \tag{12}$$

Substituindo (11) em (12), obtemos:

$$\begin{aligned}
1 &= 7 - (10 - 7 \cdot 1) \cdot 2 \\
1 &= 7 - 2 \cdot 10 - 2 \cdot 7 \cdot 1 \\
1 &= 10 \cdot (-2) + 7 \cdot (1 + 2) \\
1 &= 10 \cdot (-2) + 7 \cdot (3)
\end{aligned}$$

Portanto, pelo Teorema 5.2.1 equações (6) e (7), temos:

$$\begin{aligned}
x &= -2 + 7k \\
y &= 3 - 10k
\end{aligned}$$

Tomando $k = 1$, obtemos:

$$\begin{aligned}
x &= -2 + 7 = 5 \\
y &= 3 - 10 = -7
\end{aligned}$$

De fato, $10 \cdot 5 + 7 \cdot (-7) = 1$.

Agora, retornando a (8), podemos calcular λ:

$$\begin{aligned}
\lambda &= 13 \cdot 10^{-1} \pmod 7 \\
&= 13 \cdot 5 \pmod 7 \\
&= 65 \pmod 7 \\
&= 2
\end{aligned}$$

Agora, podemos usar (6) e (7) para calcular as coordenadas de R:

$$\begin{aligned}
x_3 &= 2^2 - 2 - 2 \pmod 7 \\
&= 0 \pmod 7 \\
&= 0
\end{aligned}$$

$$\begin{aligned}
y_3 &= 2 \cdot (2 - 0) - 5 \pmod 7 \\
&= -1 \pmod 7 \\
&= 7 - 1 \pmod 7 \\
&= 6 \pmod 7 \\
&= 6
\end{aligned}$$

Assim, encontramos que $P \oplus Q = R = (\overline{0}, \overline{6})$.

Exemplo 5.2.2. Considere a curva elíptica E definida sobre $(\mathbb{Z}_7, \oplus)$ com a equação $y^2 = x^3 + x + 1$, calcule a soma de $P = (\overline{2}, \overline{5})$ com $Q = (\overline{0}, \overline{6})$.

Devido a $P \neq Q$, aplicamos a equação (5). Assim, temos:

$$\begin{aligned}
\lambda &= \frac{6-5}{0-2} \pmod 7 \\
&= \frac{1}{-2} \pmod 7 \\
&= \frac{1 \pmod 7}{-2 \pmod 7} \\
&= \frac{1 \pmod 7}{7-2 \pmod 7} \\
&= \frac{1}{5} \pmod 7 \\
&= 5^{-1} \pmod 7
\end{aligned} \tag{13}$$

Para calcular λ, precisamos resolver a equação diofantina (1):

$$5x + 7y = mdc(5, 7) = 1$$

Vamos resolver passo a passo:

$$7 = 5 \cdot 1 + 2 \tag{14}$$

$$5 = 2 \cdot 2 + 1 \tag{15}$$

Agora, podemos usar os resultados de (14) e (15) para encontrar x e y:

$$2 = 7 - 5 \cdot 1 \tag{16}$$

$$1 = 5 - 2 \cdot 2 \tag{17}$$

Substituindo (16) em (17), obtemos:

$$\begin{aligned}
1 &= 5 - 2 \cdot 2 \\
1 &= 5 - 2 \cdot (7 - 5 \cdot 1) \\
1 &= 5 - 2 \cdot 7 + 2 \cdot 5 \\
1 &= 5 \cdot 3 + 7(-2
\end{aligned}$$

Portanto, pelo Teorema 5.2.1 equações (6) e (7), temos:

$$x = 3 + 7k$$
$$y = -2 - 5k$$

Tomando $k = 0$, obtemos:

$$x = 3 + 0 = 3$$
$$y = -2 - 0 = -2$$

De fato, $5 \cdot 3 + 7 \cdot (-2) = 1$.

Agora, retornando a (13), podemos calcular λ:

$$\begin{aligned} \lambda &= 5^{-1} \pmod 7 \\ &= 3 \pmod 7 \\ &= 3 \end{aligned}$$

Agora, podemos usar (6) e (7) para calcular as coordenadas de R:

$$\begin{aligned} x_3 &= 3^2 - 2 - 0 \pmod 7 \\ &= 7 - 7 \pmod 7 \\ &= 0 \end{aligned}$$

$$\begin{aligned} y_3 &= 3(2 - 0) - 5 \pmod 7 \\ &= 1 \pmod 7 \\ &= 1 \end{aligned}$$

Assim, encontramos que $P \oplus Q = R = (\overline{0}, \overline{1})$.

Exemplo 5.2.3. Considere a curva elíptica E definida sobre $(\mathbb{Z}_7, \oplus)$ com a equação $y^2 = x^3 + x + 1$, calcule a soma de $P = (\overline{2}, \overline{5})$ com $Q = (\overline{2}, \overline{2})$.

Devido a $P \neq Q$, aplicamos a equação (5). Assim, temos:

$$\begin{aligned} \lambda &= \frac{2 - 5}{2 - 2} \pmod 7 \\ &= \frac{-4}{0} \pmod 7 \end{aligned} \tag{18}$$

Note que (18) é uma singularidade, ou seja, não possuirá solução, deste modo, concluímos que $P \oplus Q = R = \mathcal{O}$ (*ponto no infinito*)

Seguindo os passos apresentados nos Exemplos 5.2.1, 5.2.2 e 5.2.3, para todos os pontos da curva elíptica $E : y^2 = x^3+x+1$ sobre Z_7, $E(\mathbb{Z}_7) = \{\mathcal{O}, (\overline{0}, \overline{1}), (\overline{0}, \overline{6}), (\overline{2}, \overline{2}), (\overline{2}, \overline{5})\}$, podemos construir tábua de operação $\oplus$ de E sobre $\mathbb{Z}_7$. Vide Tabela 3.

Tabela 3 – Tábua de operação $\oplus$ de $E : y^2 = x^3 + x + 1$ sobre $\mathbb{Z}_7$.

$\oplus$	$\mathcal{O}$	$(\overline{0}, \overline{1})$	$(\overline{0}, \overline{6})$	$(\overline{2}, \overline{2})$	$(\overline{2}, \overline{5})$
$\mathcal{O}$	$\mathcal{O}$	$(\overline{0}, \overline{1})$	$(\overline{0}, \overline{6})$	$(\overline{2}, \overline{2})$	$(\overline{2}, \overline{5})$
$(\overline{0}, \overline{1})$	$(\overline{0}, \overline{1})$	$(\overline{2}, \overline{5})$	$\mathcal{O}$	$(\overline{0}, \overline{6})$	$(\overline{2}, \overline{2})$
$(\overline{0}, \overline{6})$	$(\overline{0}, \overline{6})$	$\mathcal{O}$	$(\overline{2}, \overline{2})$	$(\overline{2}, \overline{5})$	$(\overline{0}, \overline{1})$
$(\overline{2}, \overline{2})$	$(\overline{2}, \overline{2})$	$(\overline{0}, \overline{6})$	$(\overline{2}, \overline{5})$	$(\overline{0}, \overline{1})$	$\mathcal{O}$
$(\overline{2}, \overline{5})$	$(\overline{2}, \overline{5})$	$(\overline{2}, \overline{2})$	$(\overline{0}, \overline{1})$	$\mathcal{O}$	$(\overline{0}, \overline{6})$

Fonte: —-

Nesta sessão, exploramos o processo de calcular a soma de dois pontos na curva elíptica $E : y^2 = x^3 + x + 1$ sobre $\mathbb{Z}_7$. Utilizamos as equações estabelecidas em (6) e (7) para a adição de pontos, que envolve a determinação de λ e a aplicação do Teorema 5.2.1 para encontrar soluções de equações diofantinas. Além disso, fornecemos uma tabela de operações $(\mathbb{Z}_7, \oplus)$ para todos os pontos de E. Na sessão subsequente, abordaremos o método para a multiplicação de um ponto $P \in E$ por um escalar $k \in \mathbb{N}$.

5.3 Algoritmo para multiplicação de um escalar por um ponto em curvas elípticas

Definição 5.3.1. Seja E uma curva elíptica definida sobre o campo finito $\mathbb{Z}_p$ com a operação de grupo $\oplus$. Dado um número natural $k \in \mathbb{N}$ e um ponto P que pertence à curva E, a operação de multiplicação escalar $k \cdot P$ pode ser representada como a soma repetida de pontos $P_1 \oplus P_2 \oplus \ldots \oplus P_n$, onde n é um número inteiro positivo.

Exemplo 5.3.1. Considere a curva elíptica E definida sobre $\mathbb{Z}_7$ com a equação $y^2 = x^3 + x + 1$. Seja o ponto $P = (2, 5)$ pertencente a curva, calcule o resultado de $3 \cdot P$.

$$3 \cdot P = P \oplus P \oplus P$$

Pela tábua de operações na Tabela 3, observamos que:

$$P \oplus P = Q = (\overline{0}, \overline{6})$$

E que $Q \oplus P$

$$Q \oplus P = (\overline{0}, \overline{1})$$

Assim, concluímos que $3 \cdot P = (\overline{0}, \overline{1})$.

De fato, os resultados estão de acordo ao observado no Exemplo 5.2.2

A multiplicação de $k \in \mathbb{N}$ por $P \in E$, conforme definida em (5.3.1), embora funcional, revela-se pouco eficiente à luz da Ciência da Computação. Isso se deve ao tempo considerável necessário para executar as operações (Maimon, 2018, p. 3-4; Reyad, 2018, p. 3). No entanto, felizmente, existe uma abordagem alternativa que permite obter os mesmos resultados em um tempo computacional significativamente menor. Portanto, para multiplicar $k \in \mathbb{N}$ por $P \in E$, é recomendável adotar o método descrito a seguir:

Proposição 5.3.1. Considere um ponto Q pertencente à curva elíptica E sobre $(\mathbb{Z}_p, \oplus)$. A operação $k \cdot Q$, no qual $k \in \mathbb{N}$ é um escalar, é mais eficiente se realizada da seguinte forma (Maimon, 2018, p. 3-4): a) Primeiro, obtenha a representação binária de k, denotada como $(k)_2$ b) Começando pelo bit mais significativo em $(k)_2$, dobre o resultado atual em cada etapa, adicionando o ponto Q se o bit correspondente for 1 .

Exemplo 5.3.2. Considere a curva elíptica E definida sobre $\mathbb{Z}_{11}$ com a equação $y^2 = x^3 + x + 1$. Seja o ponto $P = (1, 5)$ pertencente a curva, calcule o resultado de $10 \cdot P$.

Temos que $k = 10$, deste modo, seguindo a Proposição 5.3.1 a), sabemos que $(k)_2 = 1010$. Assim, pelo item b) podemos construir a Tabela 4 a seguir:

Tabela 4 – Operações realizadas para $10 \cdot P$ em $E : y^2 = x^3 + x + 1$ sobre $\mathbb{Z}_{11}$ seguindo a Proposição 5.3.1

Bit	**Operação**	**Resultado**
1	Ignore	P
0	Dobre	$P \oplus P$
1	Dobre e Adicione	$4P \oplus P$
0	Dobre	$5P \oplus 5P$

Fonte: —-

Pela tábua de operações na Tabela 5, observamos que:

$$\begin{aligned} P \oplus P = 2P &= (\overline{3}, \overline{3}) \\ 4P = 2P \oplus 2P &= (\overline{6}, \overline{5}) \\ 4P \oplus P &= (\overline{4}, \overline{6}) \\ 5P \oplus 5P &= (\overline{6}, \overline{6}) \end{aligned}$$

Assim, concluímos que $10 \cdot P = (\overline{6}, \overline{6})$.

De fato, $2P = P \oplus P = (\overline{3}, \overline{3}) \to 3P = P \oplus 2P = (\overline{8}, \overline{2}) \to 4P = P \oplus 3P = (\overline{6}, \overline{5}) \to 5P = P \oplus 4P = (\overline{4}, \overline{6}) \to 6P = P \oplus 5P = (\overline{0}, \overline{10}) \to 7P = P \oplus 6P = (\overline{2}, \overline{0}) \to 8P = P \oplus 7P = (\overline{0}, \overline{1}) \to 9P = P \oplus 8P = (\overline{4}, \overline{5}) \to 10P = P \oplus 9P = (\overline{6}, \overline{6})$. Observe que chegamos ao mesmo resultado, porem, executando uma quantidade maior de operações.

Tabela 5 – Tábua de operação $\oplus$ de $E : y^2 = x^3 + x + 1$ sobre $\mathbb{Z}_{11}$.

$\oplus$	$\mathcal{O}$	$(\overline{0}, \overline{1})$	$(\overline{0}, \overline{10})$	$(\overline{1}, \overline{5})$	$(\overline{1}, \overline{6})$	$(\overline{2}, \overline{0})$	$(\overline{3}, \overline{3})$	$(\overline{3}, \overline{8})$	$(\overline{4}, \overline{5})$	$(\overline{4}, \overline{6})$	$(\overline{6}, \overline{5})$	$(\overline{6}, \overline{6})$	$(\overline{8}, \overline{2})$	$(\overline{8}, \overline{9})$
$\mathcal{O}$	$\mathcal{O}$	$(\overline{0}, \overline{1})$	$(\overline{0}, \overline{10})$	$(\overline{1}, \overline{5})$	$(\overline{1}, \overline{6})$	$(\overline{2}, \overline{0})$	$(\overline{3}, \overline{3})$	$(\overline{3}, \overline{8})$	$(\overline{4}, \overline{5})$	$(\overline{4}, \overline{6})$	$(\overline{6}, \overline{5})$	$(\overline{6}, \overline{6})$	$(\overline{8}, \overline{2})$	$(\overline{8}, \overline{9})$
$(\overline{0}, \overline{1})$	$(\overline{0}, \overline{1})$	$(\overline{3}, \overline{3})$	$\mathcal{O}$	$(\overline{4}, \overline{5})$	$(\overline{2}, \overline{0})$	$(\overline{1}, \overline{5})$	$(\overline{6}, \overline{6})$	$(\overline{0}, \overline{10})$	$(\overline{8}, \overline{2})$	$(\overline{1}, \overline{6})$	$(\overline{3}, \overline{8})$	$(\overline{6}, \overline{5})$	$(\overline{8}, \overline{9})$	$(\overline{4}, \overline{6})$
$(\overline{0}, \overline{10})$	$(\overline{0}, \overline{10})$	$\mathcal{O}$	$(\overline{3}, \overline{8})$	$(\overline{2}, \overline{0})$	$(\overline{4}, \overline{6})$	$(\overline{1}, \overline{6})$	$(\overline{0}, \overline{1})$	$(\overline{6}, \overline{5})$	$(\overline{1}, \overline{5})$	$(\overline{8}, \overline{9})$	$(\overline{6}, \overline{6})$	$(\overline{3}, \overline{3})$	$(\overline{4}, \overline{5})$	$(\overline{8}, \overline{2})$
$(\overline{1}, \overline{5})$	$(\overline{1}, \overline{5})$	$(\overline{4}, \overline{5})$	$(\overline{2}, \overline{0})$	$(\overline{3}, \overline{3})$	$\mathcal{O}$	$(\overline{0}, \overline{1})$	$(\overline{8}, \overline{2})$	$(\overline{1}, \overline{6})$	$(\overline{6}, \overline{6})$	$(\overline{0}, \overline{10})$	$(\overline{4}, \overline{6})$	$(\overline{8}, \overline{9})$	$(\overline{6}, \overline{5})$	$(\overline{3}, \overline{8})$
$(\overline{1}, \overline{6})$	$(\overline{1}, \overline{6})$	$(\overline{2}, \overline{0})$	$(\overline{4}, \overline{6})$	$\mathcal{O}$	$(\overline{3}, \overline{8})$	$(\overline{0}, \overline{10})$	$(\overline{1}, \overline{5})$	$(\overline{8}, \overline{9})$	$(\overline{0}, \overline{1})$	$(\overline{6}, \overline{5})$	$(\overline{8}, \overline{2})$	$(\overline{4}, \overline{5})$	$(\overline{3}, \overline{3})$	$(\overline{6}, \overline{6})$
$(\overline{2}, \overline{0})$	$(\overline{2}, \overline{0})$	$(\overline{1}, \overline{5})$	$(\overline{1}, \overline{6})$	$(\overline{0}, \overline{1})$	$(\overline{0}, \overline{10})$	$\mathcal{O}$	$(\overline{4}, \overline{5})$	$(\overline{4}, \overline{6})$	$(\overline{3}, \overline{3})$	$(\overline{3}, \overline{8})$	$(\overline{8}, \overline{9})$	$(\overline{8}, \overline{2})$	$(\overline{6}, \overline{6})$	$(\overline{6}, \overline{5})$
$(\overline{3}, \overline{3})$	$(\overline{3}, \overline{3})$	$(\overline{6}, \overline{6})$	$(\overline{0}, \overline{1})$	$(\overline{8}, \overline{2})$	$(\overline{1}, \overline{5})$	$(\overline{4}, \overline{5})$	$(\overline{6}, \overline{5})$	$\mathcal{O}$	$(\overline{8}, \overline{9})$	$(\overline{2}, \overline{0})$	$(\overline{0}, \overline{10})$	$(\overline{3}, \overline{8})$	$(\overline{4}, \overline{6})$	$(\overline{1}, \overline{6})$
$(\overline{3}, \overline{8})$	$(\overline{3}, \overline{8})$	$(\overline{0}, \overline{10})$	$(\overline{6}, \overline{5})$	$(\overline{1}, \overline{6})$	$(\overline{8}, \overline{9})$	$(\overline{4}, \overline{6})$	$\mathcal{O}$	$(\overline{6}, \overline{6})$	$(\overline{2}, \overline{0})$	$(\overline{8}, \overline{2})$	$(\overline{3}, \overline{3})$	$(\overline{0}, \overline{1})$	$(\overline{1}, \overline{5})$	$(\overline{4}, \overline{5})$
$(\overline{4}, \overline{5})$	$(\overline{4}, \overline{5})$	$(\overline{8}, \overline{2})$	$(\overline{1}, \overline{5})$	$(\overline{6}, \overline{6})$	$(\overline{0}, \overline{1})$	$(\overline{3}, \overline{3})$	$(\overline{8}, \overline{9})$	$(\overline{2}, \overline{0})$	$(\overline{6}, \overline{5})$	$\mathcal{O}$	$(\overline{1}, \overline{6})$	$(\overline{4}, \overline{6})$	$(\overline{3}, \overline{8})$	$(\overline{0}, \overline{10})$
$(\overline{4}, \overline{6})$	$(\overline{4}, \overline{6})$	$(\overline{1}, \overline{6})$	$(\overline{8}, \overline{9})$	$(\overline{0}, \overline{10})$	$(\overline{6}, \overline{5})$	$(\overline{3}, \overline{8})$	$(\overline{2}, \overline{0})$	$(\overline{8}, \overline{2})$	$\mathcal{O}$	$(\overline{6}, \overline{6})$	$(\overline{4}, \overline{5})$	$(\overline{1}, \overline{5})$	$(\overline{0}, \overline{1})$	$(\overline{3}, \overline{3})$
$(\overline{6}, \overline{5})$	$(\overline{6}, \overline{5})$	$(\overline{3}, \overline{8})$	$(\overline{6}, \overline{6})$	$(\overline{4}, \overline{6})$	$(\overline{8}, \overline{2})$	$(\overline{8}, \overline{9})$	$(\overline{0}, \overline{10})$	$(\overline{3}, \overline{3})$	$(\overline{1}, \overline{6})$	$(\overline{4}, \overline{5})$	$(\overline{0}, \overline{1})$	$\mathcal{O}$	$(\overline{2}, \overline{0})$	$(\overline{1}, \overline{5})$
$(\overline{6}, \overline{6})$	$(\overline{6}, \overline{6})$	$(\overline{6}, \overline{5})$	$(\overline{3}, \overline{3})$	$(\overline{8}, \overline{9})$	$(\overline{4}, \overline{5})$	$(\overline{8}, \overline{2})$	$(\overline{3}, \overline{8})$	$(\overline{0}, \overline{1})$	$(\overline{4}, \overline{6})$	$(\overline{1}, \overline{5})$	$\mathcal{O}$	$(\overline{0}, \overline{10})$	$(\overline{1}, \overline{6})$	$(\overline{2}, \overline{0})$
$(\overline{8}, \overline{2})$	$(\overline{8}, \overline{2})$	$(\overline{8}, \overline{9})$	$(\overline{4}, \overline{5})$	$(\overline{6}, \overline{5})$	$(\overline{3}, \overline{3})$	$(\overline{6}, \overline{6})$	$(\overline{4}, \overline{6})$	$(\overline{1}, \overline{5})$	$(\overline{3}, \overline{8})$	$(\overline{0}, \overline{1})$	$(\overline{2}, \overline{0})$	$(\overline{1}, \overline{6})$	$(\overline{0}, \overline{10})$	$\mathcal{O}$
$(\overline{8}, \overline{9})$	$(\overline{8}, \overline{9})$	$(\overline{4}, \overline{6})$	$(\overline{8}, \overline{2})$	$(\overline{3}, \overline{8})$	$(\overline{6}, \overline{6})$	$(\overline{6}, \overline{5})$	$(\overline{1}, \overline{6})$	$(\overline{4}, \overline{5})$	$(\overline{0}, \overline{10})$	$(\overline{3}, \overline{3})$	$(\overline{1}, \overline{5})$	$(\overline{2}, \overline{0})$	$\mathcal{O}$	$(\overline{0}, \overline{1})$

Fonte: —-

Exemplo 5.3.3. Considere a curva elíptica E definida sobre $\mathbb{Z}_{11}$ com a equação $y^2 = x^3 + x + 1$. Seja o ponto $P = (1, 6)$ pertencente a curva, calcule o resultado de $13 \cdot P$.

Temos que $k = 13$, deste modo, seguindo a Proposição 5.3.1 a), sabemos que $(k)_2 = 1101$. Assim, pelo item b) podemos construir a Tabela 6 a seguir:

Tabela 6 – Operações realizadas para $13 \cdot P$ em $E : y^2 = x^3 + x + 1$ sobre $\mathbb{Z}_{11}$ seguindo a Proposição 5.3.1

Bit	**Operação**	**Resultado**
1	Ignore	P
1	Dobre e Adicione	$2P \oplus P$
0	Dobre	$3P \oplus 3P$
1	Dobre e Adicione	$12P \oplus P$

Fonte: —-

Pela tábua de operações na Tabela 5, observamos que:

$$\begin{aligned} 2P &= (\overline{3}, \overline{8}) \\ 2P \oplus P = 3P &= (\overline{8}, \overline{9}) \\ 3P \oplus 3P = 6P &= (\overline{0}, \overline{1}) \\ 6P \oplus 6P = 12P &= (\overline{3}, \overline{3}) \\ 12P \oplus P = 13P &= (\overline{1}, \overline{5}) \end{aligned}$$

Assim, concluímos que $13 \cdot P = (\overline{1}, \overline{5})$.

De fato, $2P = P \oplus P = (\overline{3}, \overline{8}) \rightarrow 3P = P \oplus 2P = (\overline{8}, \overline{9}) \rightarrow 4P = P \oplus 3P = (\overline{6}, \overline{6}) \rightarrow 5P = P \oplus 4P = (\overline{4}, \overline{5}) \rightarrow 6P = P \oplus 5P = (\overline{0}, \overline{1}) \rightarrow 7P = P \oplus 6P = (\overline{2}, \overline{0}) \rightarrow 8P = P \oplus 7P = (\overline{0}, \overline{10}) \rightarrow 9P = P \oplus 8P = (\overline{4}, \overline{6}) \rightarrow 10P = P \oplus 9P = (\overline{6}, \overline{5}) \rightarrow 11P = P \oplus 10P = (\overline{8}, \overline{2}) \rightarrow 12P = P \oplus 11P = (\overline{3}, \overline{3}) \rightarrow 13P = P \oplus 12P = (\overline{1}, \overline{5})$. Observe que chegamos ao mesmo resultado, porem, executando uma quantidade maior de operações.

Nesta sessão, exploramos os métodos para multiplicação de um escalar $k \in \mathbb{N}$ por um ponto $P \in E$ de uma curva elíptica E sobre $(\mathbb{Z}_p, \oplus)$, exploramos a matemática base para entendimento dos métodos e protocolos a a serem abordado nos próximos capítulos.

5.4 O método de Koblitz para codificação e decodificação de mensagens em curvas elípticas

Para realizar a codificação e/ou decodificação de mensagens usando o método de Koblitz, podemos escolher entre duas opções para o ciframento inicial das mensagens, a Tabela Unicode ou a Tabela ASCII (Brady; Davis; Tracy, 2010, p. 7). Neste contexto, nos exemplos a serem apresentados, optaremos por utilizar a Tabela ASCII[1].

Definição 5.4.1. $\mathcal{M}$ é a representação de uma mensagem na forma de uma sequência de caracteres, denotada como $\{m_1, m_2, \ldots, m_n\}$, onde n é o tamanho da mensagem. $\mathcal{A}$ é uma sequência numérica, representada como $\{a_1, a_2, \ldots, a_n\}$, que corresponde ao decimal de cada caractere de $\mathcal{M}$ obtido nas tabelas Unicode ou ASCII. Considerando $b = 2^{16}$ para Unicode e $b = 2^8$ para ASCII, a mensagem cifrada m será obtida por:

$$m = \sum_{k=1}^{n} a_k \cdot b^{k-1} \tag{19}$$

e cada elemento $\{a_1, a_2, \ldots, a_n\}$ poderá ser obtido novamente pela relação a seguir:

$$a_n = m \div b^{n-1} \pmod{b} \tag{20}$$

Nota: $\div$ neste contexto representa divisão na qual o resultado é o quociente inteiro, sem considerar a parte fracionária.

Exemplo 5.4.1. Considerando a mensagem $\mathcal{M} = \{M, a, t, e, m, a, t, i, c, a\}$, obtenha a cifra m de $\mathcal{M}$ utilizando a tabela ASCII, e em seguida, reverta m para $\mathcal{M}$.

Inicialmente vamos encontrar o valor de $\mathcal{A}$, para isso, observando a tabela ASCII

[1] Vide Tabela 7

apresa entada na Tabela 7, obtemos que:

$$\mathcal{A} = \{77, 97, 116, 101, 109, 97, 116, 105, 99, 97\}$$

Deste modo, tomando $b = 2^8$ temos:

$$\begin{aligned} m &= 77b^0 + 97b^1 + 116b^2 + 101b^3 + 109b^4 + 97b^5 + 116b^6 + 105b^7 + 99b^8 + 97b^9 \\ &= 459903375307246593663309 \end{aligned}$$

Aqui, $m = 459903375307246593663309$ é o valor cifrado de $\mathcal{M}$ na tabela ASCII.

Agora, realizando o processo reverso temos que:

$$\begin{aligned} a_0 &= 459903375307246593663309 \div 256^0 \pmod{256} \\ &= 77 \end{aligned}$$

$$\begin{aligned} a_1 &= 459903375307246593663309 \div 256^1 \pmod{256} \\ &= 97 \end{aligned}$$

$$\begin{aligned} a_2 &= 459903375307246593663309 \div 256^2 \pmod{256} \\ &= 116 \end{aligned}$$

$$\begin{aligned} a_3 &= 459903375307246593663309 \div 256^3 \pmod{256} \\ &= 101 \end{aligned}$$

$$\begin{aligned} a_4 &= 459903375307246593663309 \div 256^4 \pmod{256} \\ &= 109 \end{aligned}$$

$$\begin{aligned} a_5 &= 459903375307246593663309 \div 256^5 \pmod{256} \\ &= 97 \end{aligned}$$

$$\begin{aligned} a_6 &= 459903375307246593663309 \div 256^6 \pmod{256} \\ &= 116 \end{aligned}$$

$$
\begin{aligned}
a_7 &= 459903375307246593663309 \div 256^7 \pmod{256} \\
&= 105
\end{aligned}
$$

$$
\begin{aligned}
a_8 &= 459903375307246593663309 \div 256^8 \pmod{256} \\
&= 99
\end{aligned}
$$

$$
\begin{aligned}
a_9 &= 459903375307246593663309 \div 256^9 \pmod{256} \\
&= 97
\end{aligned}
$$

Desta maneira, obtemos que $\mathcal{A} = \{77, 97, 116, 101, 109, 97, 116, 105, 99, 97\}$, o que condiz com o resultado inicialmente apresentado. Observando os valores correspondentes na tabela ASCII obtemos que $\mathcal{M} = \{M, a, t, e, m, a, t, i, c, a\}$.

Até então, apresentamos o método de codificação da mensagem como um número, conforme descrito por Brady, Davis e Tracy (2010, p. 7). O processo de codificação de Koblitz envolve mascarar o valor de m como um ponto em uma curva elíptica. Vale a observação de Silva, Salomão e Neves (2019, p. 102) sobre a escolha da curva E: deve-se selecionar uma curva na qual o comprimento do número primo escolhido seja maior que o comprimento de m.

Nos exemplos a seguir, usaremos a curva $E : y^2 = x^3 + ax + b$ sobre $(\mathbb{Z}_p, \oplus)$, em que:

$$
\begin{aligned}
a &= 6277101735386680763835789423207666416083908700390324961276 \\
b &= 2455155546008943817740293915197451784769108058161191238065 \\
p &= 6277101735386680763835789423207666416083908700390324961279
\end{aligned}
$$

Esta curva também é conhecida como *secp192r1*[2].

Teorema 5.4.1. Dado um número primo $p \equiv 3 \pmod 4$ e um número inteiro s, a congruência $y^2 \equiv s \pmod p$ é verdadeira se e somente se ambas as seguintes condições são satisfeitas: $s^{(p+1)\div 2} \equiv s \pmod p$ e $y \equiv \pm s^{(p+1)\div 4} \pmod p$.

Demonstração. Pode ser vista em Brady, Davis e Tracy (2010, p. 9) □

Vamos aproveitar os resultados obtidos no Exemplo 5.4.1 , e pressupor que já esteja familiarizado com a Definição 5.4.1. Sendo assim, vamos codificar m em um ponto de

[2]SEC2, Standards for Efficient Cryptography Group. **SEC 2: Recommended Elliptic Curve Domain Parameters**, Version 2. [online]. Disponível em: https://www.secg.org/sec2-v2.pdf. Acesso em: 12 out. 2023.

$Q \in E$, sendo $m = 459903375307246593663309$. Observe que m tem comprimento menor que a curva escolhida, sendo assim, E é uma boa opção para o proposito que segue.

Vamos escolher um valor ω arbitrário, e encontrar $Q = (x, y) \in E$ tal que $x = m \cdot \omega + j$, com $1 \leq j < \omega - 1$. Tomemos $\omega = 100$ e $j = 1$, segue que:

$$\begin{aligned} x &= 459903375307246593663309 \cdot 100 + 1 \pmod{p} \\ &= 45990337530724659366330901 \pmod{p} \\ &= 45990337530724659366330901 \end{aligned} \tag{21}$$

Substituindo o valor de x no lado direito da equação da curva E, temos que:

$$\begin{aligned} y^2 &= 45990337530724659366330901^3 + a \cdot 45990337530724659366330901 + b \pmod{p} \\ &= 198669307762312977099423576270734450236501473690663303759 5 \end{aligned}$$

Vamos aplicar o Teorema 5.4.1 , e calcular $\lambda = s^{(p+1)\div 2} \pmod{p}$, vamos denotar este valor por λ.

$$\begin{aligned} \lambda &= 198669307762312977099423576270734450236501473690663303759 5^{(p+1)\div 2} \pmod{p} \\ &= 198669307762312977099423576270734450236501473690663303759 5 \end{aligned}$$

Como $y^2 = \lambda$, pelo Teorema 5.4.1 é possível encontrar ordenada y, da forma que segue:

$$\begin{aligned} y &= 198669307762312977099423576270734450236501473690663303759 5^{(p+1)\div 4} \pmod{p} \\ &= 267774206679971235671396456263369740725510447393555631577 4 \end{aligned} \tag{22}$$

Assim, obtemos os valores de x e y do ponto Q em E. Aqui, consideramos o valor de $j = 1$ e de imediato obtivemos um ponto da curva, entretanto, caso não fosse o caso, seguiríamos com outras escolas para j, até $j = \omega - 2$.

Agora, vamos realizar o processo contrario, dado Q, vamos obter novamente o valor de m. Das coordenadas do ponto, apenas o valor de x é significativo, e $m = (x - j) \div \omega$. Sendo assim, segue que:

$$\begin{aligned} m &= (45990337530724659366330901 - 1) \div 100 \\ &= 459903375307246593663309 \end{aligned}$$

De fato, o valor condiz com o considerado inicialmente; seguindo os passos do Exemplo 5.4.1 é possível obter a mensagem originalmente cifrada.

Tabela 7 – Tabela de códigos ASCII e caracteres correspondentes

Dec	Char	Dec	Char	Dec	Char	Dec	Char
0	NUL	32	Space	64	@	96	‘
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	’	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	ETB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	FS	60	<	92	\	124	|
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	_	127	DEL

Fonte: —-

Nesta sessão, estudamos os métodos para converter uma sequência de caracteres em um numero, e como converter este para um ponto de uma curva elíptica, assim como, o procedimento contrario para decodificação do ponto e numero dado. Estes métodos são fundamentais para o entendimento das próximas sessões, pois são seus resultados que possibilitarão a criptografia e descriptografia de dados em uma curva elíptica. Um exemplo de implementação deste método pode ser visto no Anexo 1, feito com a linguagem Bash Script com auxilio da biblioteca de criptografia *ecutils*.

5.5 O método de Diffie-Hellman para compartilhamento de chaves em um criptosistema baseado em curva elíptica

Em 1976, Martin Hellman e Whitfield Diffie revolucionaram a segurança da comunicação ao introduzir um sistema inovador de criptografia de chave assimétrica, onde a chave pública habilita a cifragem acessível ao público, enquanto a chave privada permanece confidencial, estabelecendo um marco fundamental para a autenticidade e confidencialidade nas comunicações e consolidando o conceito da assinatura digital (Pabón Cadavid, 2010, p. 67).

Definição 5.5.1. Considere uma curva elíptica E definida sobre um corpo finito $\mathbb{F}_p$ e os protagonistas Alice, Bob e Eva, em que Alice e Bob desejam criar um canal de comunicação seguro para proteger-se contra potenciais invasões de Eva. Ambos estão utilizando a mesma curva E. O protocolo Diffie-Hellman segue o fluxo apresentado na Tabela 8.

Tabela 8 – Fluxo de passos do protocolo Diffie-Hellman

Alice	**Canal Público**	**Bob**
Gera aleatoriamente $d_A \in \{1, \ldots, n\}$		Gera aleatoriamente $d_B \in \{1, \ldots, n\}$
Define $H_A = d_A \cdot G$	$H_A \Rightarrow$	Define $H_B = d_B \cdot G$
Calcula $S = d_A \cdot H_B$	$\Leftarrow H_B$	
		Calcula $S = d_B \cdot H_A$

Fonte: Maimon, 2018, p. 5

Em que n é a ordem do gerador da curva $i.e\ \forall P \in E : n \cdot P = \mathcal{O}$.

Exemplo 5.5.1. Seja a curva elíptica $E : y^2 = x^3 + x + 1$ sobre Z_7, tomemos o ponto gerador $G = (2, 5)$, seguindo os passos apresentados na Definição 5.5.1, encontre H_A, H_B e S.

Começando com o valor de $d_A = 3$, podemos calcular H_A como sendo igual a $3 \cdot G$. Sabemos que $3 \cdot G$ corresponde ao ponto $(0, 1)$. Portanto, temos $H_A = (0, 1)$. Da mesma forma, ao considerar $d_B = 2$, podemos calcular H_B como sendo igual a $2 \cdot G$. Sabemos que $2 \cdot G$ é igual a $(0, 6)$. Portanto, obtemos $H_B = (0, 6)$. Considere o personagem Alice, assim $S = d_A \cdot H_B \rightarrow S = 3 \cdot (0, 6) \rightarrow S = (2, 5)$. Considere o personagem Bob, assim $S = d_B \cdot H_A \rightarrow S = 2 \cdot (0, 1) \rightarrow S = (2, 5)$.

Observe que tanto Alice como Bob encontraram o mesmo ponto S, deste modo, toda a comunicação entre ambos pode ser feita por intermédio deste ponto. Observe também que no *Canal Público* só foi divulgado os valores H_A e H_B, chaves públicas, e que a Eva, a hipotética invasora, não conseguirá encontrar o valor de S apenas com H_A e H_B em mãos, também, que os valores d_A e d_B, chaves privadas, só estiveram em posse de seus criadores.

O que nos garante, dado um valor de H_A e H_B visíveis por Eva (no canal público), que não seja possível encontrar d_A e d_B, está na dificuldade de se resolver o problema, conhecido como: *problema do logaritmo discreto para curvas elípticas*. Seja E uma curva elíptica definida sobre Z_p, tomemos P e $Q \in E$; considere um numero natural n, de tal forma que $n \cdot P = Q$, dizemos que n é o logaritmo discreto de P na base Q; encontre n (Maimon, 2018, p. 4; Gonzaga; Pesco, 2021, p. 48).

Considerando os dados do Exemplo 5.5.1, poderíamos realizar tentativas sucessivas para encontrar o valor de n, o objetivo seria resolver $n \cdot G = H_A$ ou $n \cdot G = H_B$. Perceba que com poucas interações poderemos encontrar $n = 3$ e $n = 2$, que corresponde aos valores de d_A e d_B escolhidos. Isso se deve devido termos escolhido um grupo em que a ordem é baixa. Em aplicações reais do emprego deste protocolo, são escolhidas curvas em que a ordem é maior ou igual a 2^{160} (Andria; Gondim; Salomão, 2019, p. 99).

Nesta sessão, apresentamos o protocolo Diffie-Hellman, observamos os passos a serem seguidos para o compartilhamento de chaves entre dois personagens, também, observamos o problema matemático que guarente a segurança deste protocolo. Na próxima sessão, iremos aplicar todos os conceitos apresentados até então, e observar como eles interagem para efetivamente construirmos um criptosistema de curvas elípticas. Um exemplo de implementação deste protocolo pode ser visto no Anexo 2, feito com a linguagem Bash Script com auxilio da biblioteca de criptografia *ecutils*.

5.6 O método de Massey-Omura para um efetiva troca de informação em um criptosistema de curvas elípticas

Para compreensão do protocolo Massey-Omura, iremos seguir os passos apresentados por Brady, Davis e Tracy (2010, p. 14-15). Considere uma curva elíptica E sobre Z_p, e os personagens Alice e Bob. Alice enfrenta o desafio de garantir a segurança de uma mensagem que deseja compartilhar com Bob, sem revelar sua chave criptográfica. Ambos estão utilizando a mesma curva E. Com esse objetivo em mente, podemos seguir o fluxo do protocolo Massey-Omura, conforme apresentado na Tabela 9.
Em que n é a ordem do gerador da curva $i.e\ \forall P \in E : n \cdot P = \mathcal{O}$.

Exemplo 5.6.1. Consideremos a curva elíptica[3] $E : y^2 = x^3 + x + 1$ sobre Z_{11}, no qual $n = 14$. Alice deseja enviar a mensagem m para Bob, para isso, ela representa a mensagem m como sendo $M_A = (4, 5)$, um ponto de E. Alice gera aleatoriamente $d_A = 9$, deste modo, define $H_A = d_A \cdot M_A \rightarrow H_A = (8, 9)$. Bob gera aleatoriamente $d_B = 5$, e define $H_B = d_B \cdot H_A \rightarrow H_B = (1, 6)$. Segue que $d_A^{-1} = 9^{-1} \pmod{14} = 11 \rightarrow S_A = (8, 2)$, assim $d_B^{-1} = 5^{-1} \pmod{14} = 3 \rightarrow M_A = (4, 5)$

Observe que no Canal Público (ou canal inseguro), só ficaram visíveis os valores de

[3] A tábua de operação $\oplus$ desta curva sobre Z_{11} pode ser consultada na Tabela 5

Tabela 9 – Fluxo de passos do protocolo Massey-Omura

Alice	Canal Público	Bob
Representa m como um ponto $M_A \in E$		
Gera aleatoriamente $d_A \in \{1, \ldots, n\}$, $mdc(d_A, n) = 1$		Gera aleatoriamente $d_B \in \{1, \ldots, n\}$, $mdc(d_B, n) = 1$
Define $H_A = d_A \cdot M_A$	$H_A \Rightarrow$	Define $H_B = d_B \cdot H_A$
$S_A = (d_A^{-1} \pmod n) \cdot H_B$	$\Leftarrow H_B$	
	$S_A \Rightarrow$	$M_A = (d_B^{-1} \pmod n) \cdot S_A$

Fonte: —-

H_A, H_B e S_A, e que estes valores nada dizem sobre a mensagem M_A enviada, se analisados separadamente sem o conhecimentos das chaves privadas d_A e d_B de Alice e Bob. Perceba que, assim como no protocolo Diffie-Hellman, a segurança deste sistema é garantida pela dificuldade de encontrar d_A e d_B conhecendo-se apenas H_A, H_B e S_A.

Pode-se explorar o protocolo de Massey-Omura o combinando com o método de Koblitz. Na Sessão 5.4, codificamos a mensagem $\mathcal{M} = \{M, a, t, e, m, a, t, i, c, a\}$, utilizando a tabela ASCII e a curva E: *secp192r1* como um ponto $P \in E$ de cordeadas Px e Py apresentadas em (21), (22).

Podemos utilizar o método de Koblitz para converter uma mensagem para um ponto, e empregar o Massey-Omura para compartilha-la de maneira segura, sem que o conteúdo seja publicamente acessível. Um exemplo de implementação deste protocolo pode ser visto no Anexo 3, feito com a linguagem Bash Script com auxilio da biblioteca de criptografia *ecutils*.

5.7 O Algoritmo de Assinatura Digital de Curva Elíptica (ECDSA, sigla em inglês) para verificação de autenticidade e assinatura de mensagens

O ECDSA emergiu como uma técnica que firmemente se estabeleceu como o padrão de referência para autenticar e garantir a integridade de dados em um cenário abrangente de sistemas de comunicação e segurança cibernética. Sua concepção remonta ao ano de 1992, quando foi inicialmente proposto em resposta ao chamado do Instituto Nacional de Padrões e Tecnologia (NIST) (Johnson; Menezes; Vanstone, 2001. p. 2).

Desde então, o ECDSA conquistou uma notável adoção, obtendo reconhecimento como um padrão global inquestionável. Este algoritmo desempenha um papel crítico na garantia da autenticidade e na proteção contra manipulação indevida de informações vitais, representando um elemento fundamental em nosso mundo digital cada vez mais interconectado (Johnson; Menezes; Vanstone, 2001. p. 2).

Para a implementação do algoritmo ECDSA e a subsequente realização da assinatura de uma mensagem e verificação correspondente, podem ser adotados os procedimentos adotados por Johnson, Menezes e Vanstone (2001, p. 26), Gonzaga e Pesco (2021, p. 53-55), Ertaul e Lu (2005, p. 105) e Koblitz *et al* (2011, p. 786). Realizamos algumas adaptações na escrita dos procedimentos para facilitar o entendimento, e os apresentamos da definição a seguir:

Definição 5.7.1. Seja $E(\mathbb{Z}_p) : y^2 \equiv x^3 + Ax + B \pmod{p}$ curva elíptica, um ponto base G, um natural n, primo, tal que $\forall P \in E : n \cdot P = \mathcal{O}$, onde $\mathcal{O}$ é o ponto no infinito, m um numero natural representando uma mensagem e $d \in \{1, \ldots, n-1\}$ representando uma chave privada e $Q = d \cdot G$ representando uma chave publica.

Para Q assinar a mensagem m em E, siga os seguintes passos:

1. Escolha um $k \in \mathbb{N} \mid 1 \leq k \leq n-1$ e $mdc(k, n) = 1$.
2. Calcule $P = k \cdot G$.
3. Calcule $r = P_x \pmod{n}$. Se $r = 0$ volte ao passo 1.
4. Calcule $s = (m + r \cdot d) \cdot (k^{-1} \pmod{n}) \pmod{n}$. Se $s = 0$ volte ao passo 1.
5. A assinatura da mensagem m, por Q é o par r e s.

Para verificação da assinatura de m em E, dados r e s, siga os seguintes passos:

1. Verifique se r e s estão no intervalo $\{1, \ldots, n-1\}$
2. Calcule $w = s^{-1} \pmod{n}$
3. Calcule $u_1 = m \cdot w \pmod{n}$ e $u_2 = r \cdot w \pmod{n}$
4. Calcule $P = u_1 \cdot G \oplus u_2 \cdot Q$. Se $P = \mathcal{O}$ a assinatura é invalida.
5. Calcule $v = P_x \pmod{n}$. Se $v = r$ a assinatura é valida.

Observação 5.7.1. A Definição 5.7.1 apresenta apenas o métodos, em situação reais de emprego do ECDSA, é recomendável que se escolha parameros adequados para curva, uma forma de se obser os parametros adequados é seguir o algoritimo do *ANSI X9.62*[4] (Johnson; Menezes; Vanstone, 2001, p. 17).

Exemplo 5.7.1. Consideremos a curva elíptica[5] $E : y^2 = x^3 + 7517x + 971$ sobre Z_5, no qual $n = 7$ e $G = (0, 1)$. Alice deseja assinar uma mensagem $m = 15451$ em E, e

[4] **ANSI X9.62**. Disponível em: https://neuromancer.sk/std/methods/x962/. Acesso em: 17 out. 2023.
[5] A tábua de operação $\oplus$ desta curva sobre Z_5 pode ser consultada na Tabela 5

envia-la para Bob, que então, fará a verificação da assinatura. Considere que Alice escolheu $d = 3 \rightarrow Q = (3, 3)$ como sua chave privada. Desta forma, temos:

$$\begin{aligned} k &= 2 \\ P &= (1, 3) \\ r &= 1 \pmod 7 \\ &= 1 \\ s &= (15451 + 1 \cdot 3) \cdot (2^{-1} \pmod 7) \pmod 7 \\ &= 6 \end{aligned}$$

Assim, a assinatura da mensagem m por Alice é o par $r = 1$ e $s = 6$. Bob recebe de Alice os valores: $r = 1$, $s = 6$, $Q = (3, 3)$ e a mensagem $m = 15451$, assim, segue com a verificação se a mensagem foi, de fato, assinada por ela. Desta forma, segue: $1 \leq r \leq n - 1$ e $1 \leq s \leq n - 1$

$$\begin{aligned} w &= 6^{-1} \pmod 7 \\ &= 6 \\ u_1 &= 15451 \cdot 6 \pmod 7 \\ &= 5 \\ u_2 &= 1 \cdot 6 \pmod 7 \\ &= 6 \\ P &= 5 \cdot G \oplus 6 \cdot Q \\ &= (1, 3) \\ v &= 1 \pmod 7 \\ &= 1 \end{aligned}$$

Como $v = r$, a assinatura é valida.

Tabela 10 – Tábua de operação $\oplus$ de $E : y^2 = x^3 + 7517x + 971$ sobre $\mathbb{Z}_5$.

$\oplus$	$\mathcal{O}$	$(\overline{0}, \overline{1})$	$(\overline{1}, \overline{3})$	$(\overline{3}, \overline{3})$	$(\overline{3}, \overline{2})$	$(\overline{1}, \overline{2})$	$(\overline{0}, \overline{4})$
$\mathcal{O}$	$\mathcal{O}$	$(\overline{0}, \overline{1})$	$(\overline{1}, \overline{3})$	$(\overline{3}, \overline{3})$	$(\overline{3}, \overline{2})$	$(\overline{1}, \overline{2})$	$(\overline{0}, \overline{4})$
$(\overline{0}, \overline{1})$	$(\overline{0}, \overline{1})$	$(\overline{1}, \overline{3})$	$(\overline{3}, \overline{3})$	$(\overline{3}, \overline{2})$	$(\overline{1}, \overline{2})$	$(\overline{0}, \overline{4})$	$\mathcal{O}$
$(\overline{1}, \overline{3})$	$(\overline{1}, \overline{3})$	$(\overline{3}, \overline{3})$	$(\overline{3}, \overline{2})$	$(\overline{1}, \overline{2})$	$(\overline{0}, \overline{4})$	$\mathcal{O}$	$(\overline{0}, \overline{1})$
$(\overline{3}, \overline{3})$	$(\overline{3}, \overline{3})$	$(\overline{3}, \overline{2})$	$(\overline{1}, \overline{2})$	$(\overline{0}, \overline{4})$	$\mathcal{O}$	$(\overline{0}, \overline{1})$	$(\overline{1}, \overline{3})$
$(\overline{3}, \overline{2})$	$(\overline{3}, \overline{2})$	$(\overline{1}, \overline{2})$	$(\overline{0}, \overline{4})$	$\mathcal{O}$	$(\overline{0}, \overline{1})$	$(\overline{1}, \overline{3})$	$(\overline{3}, \overline{3})$
$(\overline{1}, \overline{2})$	$(\overline{1}, \overline{2})$	$(\overline{0}, \overline{4})$	$\mathcal{O}$	$(\overline{0}, \overline{1})$	$(\overline{1}, \overline{3})$	$(\overline{3}, \overline{3})$	$(\overline{3}, \overline{2})$
$(\overline{0}, \overline{4})$	$(\overline{0}, \overline{4})$	$\mathcal{O}$	$(\overline{0}, \overline{1})$	$(\overline{1}, \overline{3})$	$(\overline{3}, \overline{3})$	$(\overline{3}, \overline{2})$	$(\overline{1}, \overline{2})$

Fonte: —-

Nesta sessão, exploramos os passos para criação e verificação de uma assinatura digital em uma curva eliptica, criptomoedas como o Bitcoin utilizam este metodo para garantir a integridade das transações de criptmoedas realizadas em sua rede (Gonzaga; Pesco, 2021). Um exemplo de implementação deste algoritimo pode ser visto no Anexo 4, feito com a linguagem Bash Script com auxilio da biblioteca de criptografia *ecutils*.

5.8 Implementação prática: criptografia de curvas elípticas em um sistema de chat seguro

Até o momento, examinamos os princípios teóricos relacionados à criptografia de curvas elípticas, revisando conceitos essenciais para esclarecer a base teórica que sustentará o que será abordado nesta seção. Agora, vamos explorar como aplicar os protocolos estudados na criação de um sistema de criptografia baseado em curvas elípticas. Apresentaremos dois exemplos de implementação usando a linguagem de programação Python, com o objetivo de ilustrar como um sistema de chat com criptografia ponta a ponta pode ser desenvolvido.

No primeiro exemplo, empregaremos o método de Koblitz em conjunto com o protocolo Diffie-Hellman e ECDSA. No segundo exemplo, utilizaremos o método de Koblitz em conjunto com o protocolo Massey-Omura e ECDSA. Em ambos os casos, a troca de chaves entre os participantes do chat será assimétrica, e uma vez concluída, toda a comunicação ocorrerá de forma simétrica.

5.8.1 Combinado o método de Koblitz, protocolo Diffie-Hellman e algorítimo ECDSA

```
```python
# Parâmetros da Curva Elíptica (secp192r1)
# Estabelecemos parâmetros para uma curva elíptica, que
# são comuns a ambas as partes.
# Esses parâmetros definem a forma da curva e são usados
# para operações criptográficas.
p = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFFFFFFFFFFFF
a = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFFFFFFFFFFFC
b = 0x64210519E59C80E70FA7E9AB72243049FEB8DEECC146B9B1

# G é um ponto base na curva elíptica. É um ponto público
# amplamente conhecido.
G = (
0x188DA80EB03090F67CBF20EB43A18800F4FF0AFD82FF1012,
0x07192B95FFC8DA78631011ED6B24CDD573F977A11E794811,
)
```
```

```
# n é a ordem da curva, que representa o número de pontos
# na curva.
n = 0xFFFFFFFFFFFFFFFFFFFFFFFF99DEF836146BC9B1B4D22831

# h é o cofator, que é igual a 1 no exemplo dado.
h = 0x1
```

```python
# Protocolo Diffie-Hellman
# Alice gera sua chave privada d_A e calcula seu ponto
# público H_A na curva elíptica.
d_A = 0xFACBE51050A25622A62131383E70EBF7C6DA8BB5AE6D2215
H_A = elliptic_curve_multiplication(d_A, G)
H_A
```

(399048536151859423482224547469805972647660025511046866464 5,
265670790606249146776468929070352755532824669303111683109)

```python
# Bob gera sua chave privada d_B e calcula seu ponto público
# H_B na curva elíptica.
d_B = 0x5FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFA000000000000002
H_B = elliptic_curve_multiplication(d_B, G)
H_B
```

(620386484204472875064802954563748835815025153726839200880 3,
588012167218420083788043633663741193727418191092162951 57)

```python
# Alice e Bob calculam um segredo compartilhado S_A e S_B.
# Isso é feito multiplicando a chave privada de um com o
# ponto público do outro.
S_A = elliptic_curve_multiplication(d_A, H_B)
S_B = elliptic_curve_multiplication(d_B, H_A)
```

```python
# Verificam se o segredo compartilhado é o mesmo, o que
# garante que ambos possuem o mesmo segredo.
segredo_compartilhado = S_A == S_B  # Deve ser True
segredo_compartilhado
```

True

```python
# Algoritmo de Koblitz
# Alice codifica uma mensagem (por exemplo, "Matematica")
# como um ponto na curva elíptica.
# Isso envolve um processo de codificação específico para
# curvas elípticas Koblitz.
P, j = koblitz_encode("Matematica")
P
```

(459903375307246593663309 01,
267774206679971235671396456263369740725510447393555631577 4)

```python
# Alice usa o segredo compartilhado para criptografar a
# mensagem para Bob.
# Ela multiplica o ponto da mensagem pelo componente x do
# segredo compartilhado S_A.
M = elliptic_curve_multiplication(S_A[0], P)
M
```

(164417786697967903626070938747924548842622564715062582164 0,
330414942292487068934441777500463139554719766620414548253 3)

```python
# Alice assina digitalmente a mensagem M.
# Isso envolve a geração de uma assinatura digital usando
# sua chave privada d_A.
r, s = ecdsa_generate_signature(d_A, M[0])
```

```
r, s
```

```
(426679100812330957139738757186245877801648629157470407006 2,
501595235600740755284717822257650884179635393144388464780 0)
```

```python
# Bob recebe a mensagem M, o valor j (que é usado na
# codificação), e a assinatura digital (r, s) de Alice.
# Bob verifica a assinatura digital de Alice usando o
# ponto público H_A.
assinatura_valida = ecdsa_verify_signature(
H_A, M[0], r, s
)  # Deve ser True
assinatura_valida
```

```
True
```

```python
# Bob obtém o ponto original da mensagem M, usando o
# inverso multiplicativo do seu componente x.
Q = elliptic_curve_multiplication(pow(S_B[0], -1, n), M)
Q
```

```
(459903375307246593663309 01,
267774206679971235671396456263369740725510447393555631 5774)
```

```python
# Bob decodifica a mensagem recebida usando o valor j e
# obtém a mensagem original.
mensagem_decodificada = koblitz_decode(
Q, j
)  # Deve ser 'Matematica'
mensagem_decodificada
```

```
'Matematica'
```

A implementação deste exemplo envolve a utilização dos códigos apresentados nos Anexos 5, 6 e 7. No cenário em questão, Alice empregou o protocolo Diffie-Hellman para

estabelecer um canal de comunicação seguro entre ela e Bob. Em seguida, ela utilizou o método de Koblitz para codificar a mensagem, adicionou uma camada de criptografia e acrescentou uma assinatura digital antes de encaminhar a mensagem a Bob. Posteriormente, Bob validou a assinatura digital de Alice e, por meio de sua implementação, decodificou a mensagem recebida. Este é um exemplo simples que demonstra o funcionamento de um sistema de mensagens com criptografia de ponta a ponta, fornecendo segurança e autenticidade na comunicação entre as partes.

5.8.2 Combinado o método de Koblitz, protocolo Massey-Omura e algorítimo ECDSA

```
'''python
# Parâmetros da Curva Elíptica (secp192r1)
# Estabelecemos parâmetros para uma curva elíptica, que
# são comuns a ambas as partes.
# Esses parâmetros definem a forma da curva e são usados
# para operações criptográficas.
p = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFFFFFFFFFFFF
a = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFFFFFFFFFFFC
b = 0x64210519E59C80E70FA7E9AB72243049FEB8DEECC146B9B1

# G é um ponto base na curva elíptica. É um ponto público
# amplamente conhecido.
G = (
0x188DA80EB03090F67CBF20EB43A18800F4FF0AFD82FF1012,
0x07192B95FFC8DA78631011ED6B24CDD573F977A11E794811,
)

# n é a ordem da curva, que representa o número de pontos
# na curva.
n = 0xFFFFFFFFFFFFFFFFFFFFFFFF99DEF836146BC9B1B4D22831

# h é o cofator, que é igual a 1 no exemplo dado.
h = 0x1
'''
```

```
'''python
# Protocolo Massey-Omura
# Alice codifica uma mensagem como um ponto da curva
# elíptica
M_A, j = koblitz_encode("Matematica")
```

```
M_A, j
```

```
((459903375307246593663309 01,
267774206679971235671396456263369740725510447393555631577 4),
1)
```

```python
# Alice gera sua chave privada d_A e calcula seu ponto
# público P_A na curva elíptica.
d_A = 0xFACBE51050A25622A62131383E70EBF7C6DA8BB5AE6D2215
P_A = elliptic_curve_multiplication(d_A, G)
d_A, P_A
```

```
(614951140355464396830601702155829637764443878304437168590 9,
(399048536151859423482224547469805972647660025511046866464 5,
265670790606249146776468929070352755532824669303111683109))
```

```python
# Alice assina digitalmente a mensagem M_A com sua
# chave privada d_A
r_A, s_A = ecdsa_generate_signature(d_A, M_A[0])
r_A, s_A
```

```
(488230953187712985186125360253240197454949386236107320544 4,
197523509489026339389970653537695303607877155534553631277 1)
```

```python
# Alice calcula sua chave pública (de M_A) H_A
H_A = elliptic_curve_multiplication(d_A, M_A)
H_A
```

```
(235269103036049519424087788030935133408745774947261609559 7,
842885506933006410592700425384460770193960857914983449523)
```

```python
# Bob gera sua chave privada d_B e calcula seu ponto público
# P_B na curva elíptica.
d_B = 0x5FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFA000000000000002
P_B = elliptic_curve_multiplication(d_B, G)
```

```
d_B, P_B
```

```
(23539131507700052864384210337028749060314657626463718604 82,
(62038648420447287506480295456374883581502515372683920088 03,
58801216721842008378804363366374119372741819109216295157))
```

```python
# Bob recebe o valor H_A e P_A de Alice e calcula sua
# chave pública (de H_A) H_B
H_B = elliptic_curve_multiplication(d_B, H_A)
H_B
```

```
(27608177693583794036751704984550256271615621748986255707 67,
33404911596380148844038201532016606355175082525240869254 41)
```

```python
# Bob assina digitalmente a mensagem H_B com sua chave
# privada d_B
r_B, s_B = ecdsa_generate_signature(d_B, H_B[0])
r_B, s_B
```

```
(39220376507250480899055014305589463146123512292131974179 85,
51226745654744437565689756105311890504183682004935278741 03)
```

```python
# Alice recebe o valor P_B, H_B, r_B e s_B de Bob
# e verifica se H_B tem assinatura digital válida
assinatura_valida = ecdsa_verify_signature(
P_B, H_B[0], r_B, s_B
)
assinatura_valida
```

```
True
```

```python
# Alice calcula o segredo compartilhado S_A
S_A = elliptic_curve_multiplication(pow(d_A, -1, n), H_B)
S_A
```

```
(256818788376329300389786911472193955601505766377517525290 1,
35628946481154651271201370442821745785673244944273166465 55)
```python
# Bob recebe o valor S_A, j, r_A e s_a de Alice e calcula
# o valor de M_B
M_B = elliptic_curve_multiplication(pow(d_B, -1, n), S_A)
M_B
```

```
(45990337530724659366330901,
26777420667997123567139645626336974072551044739355563157 74)
```python
# Bob verifica se a assinatura digital da mensagem M_B
# é válida
assinatura_valida = ecdsa_verify_signature(
P_A, M_B[0], r_A, s_A
)
assinatura_valida
```

```
True
```python
# Bob decodifica a mensagem recebida usando o valor j e
# obtém a mensagem original
mensagem_decodificada = koblitz_decode(M_B, j)
mensagem_decodificada
```

```
'Matematica'
```

Este exemplo demonstra como a criptografia de chave pública e a assinatura digital, implementadas por meio de parâmetros de curva elíptica, podem ser utilizadas por Alice e Bob para garantir a autenticidade e a confidencialidade das mensagens em um cenário de comunicação segura. Ambos geram chaves privadas e públicas, assinam e verificam mensagens, calculam segredos compartilhados e decodificam mensagens codificadas, proporcionando segurança nas trocas de informações, essenciais para manter a integridade e privacidade da comunicação. A implementação deste exemplo envolve a utilização dos códigos apresentados nos Anexos 5, 6 e 7.
```

6 CONSIDERAÇÕES FINAIS

Ao longo deste trabalho, buscou-se desvendar a abordagem da criptografia, lançando uma luz sobre a complexidade subjacente aos processos rotineiros de comunicação segura. A criptografia, em suas variadas formas, permeia nosso cotidiano de tal maneira que sua influência passa quase completamente despercebida, apesar de sua importância crítica.

Os exemplos apresentados nas seções dedicadas à implementação prática da criptografia de Curvas Elípticas em um sistema de Chat seguro têm caráter ilustrativo. Eles procuraram demonstrar a aplicação conjunta dos protocolos de criptografia estudados, embora otimizações adicionais e a análise de vulnerabilidades de segurança não tenham sido nosso principal escopo neste trabalho.

Além disso, neste estudo, levou-se a cabo a elaboração de uma biblioteca em Python, chamada *Elliptic Curve Utils (ecutils)*. Esta biblioteca implementa os algoritmos estudados e permite a execução das operações matemáticas básicas nas curvas elípticas. É um instrumento valioso para quem está ingressando no estudo da criptografia e pode ser facilmente acessada e instalada através do repositório GitHub[6] ou do *Python Package Index*[7] (PyPI).

É fundamental ressaltar que este estudo não esgotou todos os aspectos teóricos acerca da criptografia e suas aplicações. Nosso objetivo principal foi fornecer uma introdução abrangente ao tema e um trampolim para mais pesquisas em profundidade. As referências listadas oferecem excelentes pontos de partida para aqueles que desejam se aprofundar ainda mais nas camadas de complexidade.

A importância da criptografia em nossas vidas não pode ser subestimada. Ela é a força que nos permite interagir com sistemas informatizados com um nível razoável de segurança e confiança. Diante disso, este estudo pode ser considerado um primeiro passo em direção a uma maior compreensão e apreciação desta disciplina crucial. Ainda há muito terreno a ser coberto, principalmente na fronteira emergente da criptografia e da computação quântica.

Finalmente, respaldado pelos tópicos abordados neste trabalho, é empolgante imaginar as possibilidades futuras para a criptografia. Diante do contínuo e vertiginoso avanço das tecnologias digitais e computacionais, acreditamos que a criptografia apenas crescerá em importância e relevância. Ao continuar pesquisando e construindo sobre os fundamentos estabelecidos aqui, podemos contribuir para a evolução de um mundo digital mais seguro e confiável.

[6] https://github.com/isakruas/ecutils
[7] https://pypi.org/project/ecutils

REFERÊNCIAS

ABREU, Jacqueline de Souza. **Passado, Presente e Futuro da Criptografia Forte**: desenvolvimento tecnológico e regulação. *Revista Brasileira de Políticas Públicas*, Brasília, ano 2017, v. 7, n. 3, p. 24-42, 9 dez. 2017. DOI http://doi.org/10.5102/rbpp.v7i3.4869. Disponível em: https://www.publicacoes.uniceub.br/RBPP/article/view/4869. Acesso em: 15 out. 2023.

ANDRIA, Sally; GONDIM, Rodrigo; SALOMÃO, Rodrigo. **Introdução à Criptografia com Curvas Elípticas**. Rio de Janeiro: IMPA, 2019. 139 p. Disponível em: https://impa.br/wp-content/uploads/2022/03/32CBM09_eBook.pdf. Acesso em: 12 out. 2023.

BRADY, Renee; DAVIS, Naleceia; TRACY, Anna. **Encrypting with Elliptic Curve Cryptography**. *Florida AM University. Naleceia Davis. Spelman College. Anna Tracy. University of the South*, 2010. Disponível em: https://www.msri.org/msri_ups/531/schedules/4117. Acesso em: 21 jan. 2022.

ERTAUL, Levent; LU, Weimin. ECC Based Threshold Cryptography for Secure Data Forwarding and Secure Key Exchange in MANET (I). In: **Networking 2005. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. p. 102-113. ISBN: 978-3-540-32017-3. DOI http://doi.org/10.1007/11422778_9. Disponível em: https://link.springer.com/chapter/10.1007/11422778_9. Acesso em: 15 out. 2023.

FILHO, José Eustáquio Ribeiro Vieira; AZEREDO, Paula Prestes. Tecnologia, criptografia e matemática: da troca de mensagens ao suporte em transações econômicas. **Desenvolvimento Socioeconômico em Debate**, [S. l.], v. 2, n. 2, p. 22–31, 2017. DOI: 10.18616/rdsd.v2i2.3225. Disponível em: https://periodicos.unesc.net/ojs/index.php/RDSD/article/view/3225. Acesso em: 12 out. 2023.

GONZAGA, Raoni do Nascimento; PESCO, Sinésio. **Bitcoin: uma introdução à matemática das transações**. Rio de Janeiro: [s.n.], 2021. 65 p. Dissertação (Mestrado em Matemática) - Pontifícia Universidade Católica do Rio de Janeiro. Disponível em: https://www.maxwell.vrac.puc-rio.br/54118/54118.PDF. Acesso em: 12 out. 2023.

JOHNSON, Don; MENEZES, Alfred; VANSTONE, Scott. **BThe Elliptic Curve Digital Signature Algorithm (ECDSA)**. International Journal of Information Security, v. 1, n. 1, p. 36-63, 2001. DOI https://doi.org/10.1007/s102070100002

KOBLITZ, Ann Hibner, *et. al*. **Elliptic Curve Cryptography: The Serpentine Course of a Paradigm Shift**. *Journal of Number Theory*, vol. 131, no 5, maio de 2011, p. 781–814. DOI.org (Crossref). DOI https://doi.org/10.1016/j.jnt.2009.01.006. Disponível em: https://www.sciencedirect.com/science/article/pii/S0022314X09000481?via%3Dihub.

Acesso em: 21 jan. 2022.

KOBLITZ, Neal. **Elliptic Curve Cryptosystems**. *Mathematics of Computation*, v. 4X, n. 177, p. 20.1-209, 1987. Disponível em: https://www.ams.org/journals/mcom/1987-48-177/S0025-5718-1987-0866109-5/S0025-5718-1987-0866109-5.pdf. Acesso em: 12 out. 2023.

MAIMON, Shir. Elliptic Curve Cryptography. **University of Rochester**, Rochester, p. 1-13, 10 maio 2018. Disponível em: https://www.sas.rochester.edu/mth/undergraduate/honorspaperspdfs/shirmaimon2018pdf. Acesso em: 15 out. 2023.

OLIVEIRA, Ronielton Rezende. **Criptografia simétrica e assimétrica - os principais algoritmos de cifragem**. *Segurança Digital* [Revista online], v. 31, p. 11-15, 2012. Disponível em: https://www.ronielton.eti.br/publicacoes/artigorevistasegurancadigital2012.pdf. Acesso em: 12 out. 2023.

PABÓN CADAVID, Jhonny Antonio. La criptografía y la protección a la información digital. **Revista La Propiedad Inmaterial**, [S. l.], n. 14, p. 59–90, 2010. Disponível em: https://revistas.uexternado.edu.co/index.php/propin/article/view/2476. Acesso em: 12 out. 2023.

REYAD, Omar. Text message encoding based on elliptic Curve Cryptography and a mapping methodology. *Information Sciences Letters*, v. 7, n. 1, p. 2, 2018. Disponível em: https://digitalcommons.aaru.edu.jo/cgi/viewcontent.cgi?article=1060&context=isl. Acesso em: 12 out. 2023.

SERAGIOTTO, Cesar. **Criptografia baseada em Identidade**. Disponível em: http://www.linux.ime.usp.br/~cef/mac499-04/monografias/rec/cesarse/monografia.html. Acesso em: 12 out. 2023.

VIANA, Cleberton Junio, *et. al*. Criptografia e segurança. **Revista Científica e-Locução**, v. 1, n. 22, p. 30, 19 dez. 2022. Disponível em: https://periodicos.faex.edu.br/index.php/e-Locucao/article/view/506. Acesso em: 12 out. 2023.

LENSTRA, H. W. **Factoring integers with elliptic curves**. *Annals of Mathematics*, 1987. Disponível em: https://wstein.org/edu/Fall2001/124/lenstra/lenstra.pdf. Acesso em: 12 out. 2023.

ANEXO A – Exemplo de implementação do método de Koblitz com a biblioteca ecutils (Script em Bash)

```
#!/bin/bash

M_1="Matematica"

echo -n "M_1 = {"

for ((i=0; i<${#M_1}; i++)); do
    echo -n "${M_1:i:1}"
    if [ $i -lt $((${#M_1}-1)) ]; then
        echo -n ", "
    fi
done

echo "}"

echo -n "A_1 = {"

for ((i=0; i<${#M_1}; i++)); do
    echo -n "$(printf "%d" "'${M_1:i:1}'")"
    if [ $i -lt $((${#M_1}-1)) ]; then
        echo -n ", "
    fi
done

echo "}"

P=$(./ecutils -eck -eck-ec-get "secp192r1" \
    -eck-encode \
    -eck-encoding-type "ascii" \
    -eck-encode-message "$M_1")

J=$(echo "$P" | cut -d' ' -f3)

J_decimal=$(echo "ibase=16; $J" | bc)

echo "J = $J_decimal"
```

```
Px=$(echo "$P" | cut -d' ' -f1)
Py=$(echo "$P" | cut -d' ' -f2)

Px_decimal=$(echo "ibase=16; $Px" | bc)
Py_decimal=$(echo "ibase=16; $Py" | bc)

echo "P = ($Px_decimal, $Py_decimal)"

M_2=$(./ecutils -eck -eck-ec-get "secp192r1" \
    -eck-decode \
    -eck-encoding-type "ascii" \
    -eck-decode-px "$Px" \
    -eck-decode-py "$Py" \
    -eck-decode-j "$J"
)

echo -n "A_2 = {"

for ((i=0; i<${#M_2}; i++)); do
    echo -n "$(printf "%d" "'${M_2:i:1}'")"
    if [ $i -lt $((${#M_2}-1)) ]; then
        echo -n ", "
    fi
done

echo "}"

echo -n "M_2 = {"

for ((i=0; i<${#M_2}; i++)); do
    echo -n "${M_2:i:1}"
    if [ $i -lt $((${#M_2}-1)) ]; then
        echo -n ", "
    fi
done

echo "}"
```

ANEXO B – Exemplo de implementação do protocolo de Diffie-Hellman com a biblioteca ecutils (Script em Bash)

```
#!/bin/bash

Gx=4
Gy=5

echo "G = ($Gx, $Gy)"

d_A=9

echo "d_A = $d_A"

H_A=$(./ecutils -ec -ec-define \
    -ec-define-p "B" \
    -ec-define-a "1" \
    -ec-define-b "1" \
    -ec-trapdoor \
    -ec-trapdoor-k "$d_A" \
    -ec-trapdoor-gx "$Gx" \
    -ec-trapdoor-gy "$Gy")

H_Ax=$(echo "$H_A" | cut -d' ' -f1)
H_Ay=$(echo "$H_A" | cut -d' ' -f2)

echo "H_A = ($H_Ax, $H_Ay)"

d_B=5

echo "d_B = $d_B"

H_B=$(./ecutils -ec -ec-define \
    -ec-define-p "B" \
    -ec-define-a "1" \
    -ec-define-b "1" \
    -ec-trapdoor \
    -ec-trapdoor-k "$d_B" \
    -ec-trapdoor-gx "$Gx" \
```

```
    -ec-trapdoor-gy "$Gy")

H_Bx=$(echo "$H_B" | cut -d' ' -f1)
H_By=$(echo "$H_B" | cut -d' ' -f2)

echo "H_B = ($H_Bx, $H_By)"

S_A=$(./ecutils -ec -ec-define \
    -ec-define-p "B" \
    -ec-define-a "1" \
    -ec-define-b "1" \
    -ec-trapdoor \
    -ec-trapdoor-k "$d_A" \
    -ec-trapdoor-gx "$H_Bx" \
    -ec-trapdoor-gy "$H_By")

S_Ax=$(echo "$S_A" | cut -d' ' -f1)
S_Ay=$(echo "$S_A" | cut -d' ' -f2)

echo "S_A = ($S_Ax, $S_Ay) : S_A = d_A * H_B"

S_B=$(./ecutils -ec -ec-define \
    -ec-define-p "B" \
    -ec-define-a "1" \
    -ec-define-b "1" \
    -ec-trapdoor \
    -ec-trapdoor-k "$d_B" \
    -ec-trapdoor-gx "$H_Ax" \
    -ec-trapdoor-gy "$H_Ay")

S_Bx=$(echo "$S_B" | cut -d' ' -f1)
S_By=$(echo "$S_B" | cut -d' ' -f2)

echo "S_B = ($S_Bx, $S_By) : S_B = d_B * H_A"
```

ANEXO C – Exemplo de implementação do protocolo de Massey-Omura com a biblioteca ecutils (Script em Bash)

```
#!/bin/bash

M_Ax=4
M_Ay=5

echo "M_A = ($M_Ax, $M_Ay)"

d_A=9
d_A_i=B

echo "d_A = $d_A"

H_A=$(./ecutils -ec -ec-define \
    -ec-define-p "B" \
    -ec-define-a "1" \
    -ec-define-b "1" \
    -ec-trapdoor \
    -ec-trapdoor-k "$d_A" \
    -ec-trapdoor-gx "$M_Ax" \
    -ec-trapdoor-gy "$M_Ay")

H_Ax=$(echo "$H_A" | cut -d' ' -f1)
H_Ay=$(echo "$H_A" | cut -d' ' -f2)

echo "H_A = ($H_Ax, $H_Ay)"

d_B=5
d_B_i=3

echo "d_B = $d_B"

H_B=$(./ecutils -ec -ec-define \
    -ec-define-p "B" \
    -ec-define-a "1" \
    -ec-define-b "1" \
    -ec-trapdoor \
```

```
    -ec-trapdoor-k "$d_B" \
    -ec-trapdoor-gx "$H_Ax" \
    -ec-trapdoor-gy "$H_Ay")

H_Bx=$(echo "$H_B" | cut -d' ' -f1)
H_By=$(echo "$H_B" | cut -d' ' -f2)

echo "H_B = ($H_Bx, $H_By)"

echo "d_A_i = $d_A_i"
S_A=$(./ecutils -ec -ec-define \
    -ec-define-p "B" \
    -ec-define-a "1" \
    -ec-define-b "1" \
    -ec-trapdoor \
    -ec-trapdoor-k "$d_A_i" \
    -ec-trapdoor-gx "$H_Bx" \
    -ec-trapdoor-gy "$H_By")

S_Ax=$(echo "$S_A" | cut -d' ' -f1)
S_Ay=$(echo "$S_A" | cut -d' ' -f2)

echo "S_A = ($S_Ax, $S_Ay)"

echo "d_B_i = $d_B_i"
M_A_2=$(./ecutils -ec -ec-define \
    -ec-define-p "B" \
    -ec-define-a "1" \
    -ec-define-b "1" \
    -ec-trapdoor \
    -ec-trapdoor-k "$d_B_i" \
    -ec-trapdoor-gx "$S_Ax" \
    -ec-trapdoor-gy "$S_Ay")

M_A_2x=$(echo "$M_A_2" | cut -d' ' -f1)
M_A_2y=$(echo "$M_A_2" | cut -d' ' -f2)

echo "M_A = ($M_A_2x, $M_A_2y)"
```

ANEXO D – Exemplo de implementação do protocolo de ECDSA com a biblioteca ecutils (Script em Bash)

```
#!/bin/bash

# Parâmetros da curva elíptica
ec_p_hex="5"
ec_p_decimal=$(echo "ibase=16; $ec_p_hex" | bc)

ec_a_hex="1D5D"
ec_a_decimal=$(echo "ibase=16; $ec_a_hex" | bc)

ec_b_hex="3CB"
ec_b_decimal=$(echo "ibase=16; $ec_b_hex" | bc)

ec_gx_hex="0"
ec_gx_decimal=$(echo "ibase=16; $ec_gx_hex" | bc)

ec_gy_hex="1"
ec_gy_decimal=$(echo "ibase=16; $ec_gy_hex" | bc)

ec_n_hex="7"
ec_n_decimal=$(echo "ibase=16; $ec_n_hex" | bc)

ec_h_hex="1"
ec_h_decimal=$(echo "ibase=16; $ec_h_hex" | bc)

# Exibir informações da curva elíptica
echo "Parâmetros da curva elíptica:"
echo "p = $ec_p_decimal"
echo "a = $ec_a_decimal"
echo "b = $ec_b_decimal"
echo "n = $ec_n_decimal"
echo "h = $ec_h_decimal"
echo "G = ($ec_gx_decimal, $ec_gy_decimal)"

# Chave privada e mensagem
d_hex="3"
d_decimal=$(echo "ibase=16; $d_hex" | bc)
```

```
M_hex="3C5B"
M_decimal=$(echo "ibase=16; $M_hex" | bc)

# Exibir chave privada e mensagem
echo "Chave privada (d) = $d_decimal"
echo "Mensagem (M) = $M_decimal"

# Gerar chave pública (Q)
Q=$(./ecutils -ecdsa -ecdsa-ec-define \
    -ecdsa-ec-define-p "$ec_p_hex" \
    -ecdsa-ec-define-a "$ec_a_hex" \
    -ecdsa-ec-define-b "$ec_b_hex" \
    -ecdsa-ec-define-gx "$ec_gx_hex" \
    -ecdsa-ec-define-gy "$ec_gy_hex" \
    -ecdsa-ec-define-n "$ec_n_hex" \
    -ecdsa-ec-define-h "$ec_h_hex" \
    -ecdsa-private-key "$d_hex" \
    -ecdsa-get-public-key)

Qx_hex=$(echo "$Q" | cut -d' ' -f1)
Qx_decimal=$(echo "ibase=16; $Qx_hex" | bc)

Qy_hex=$(echo "$Q" | cut -d' ' -f2)
Qy_decimal=$(echo "ibase=16; $Qy_hex" | bc)

# Exibir chave pública (Q)
echo "Chave pública (Q) = ($Qx_decimal, $Qy_decimal)"

# Assinar a mensagem (M)
U=$(./ecutils -ecdsa -ecdsa-ec-define \
    -ecdsa-ec-define-p "$ec_p_hex" \
    -ecdsa-ec-define-a "$ec_a_hex" \
    -ecdsa-ec-define-b "$ec_b_hex" \
    -ecdsa-ec-define-gx "$ec_gx_hex" \
    -ecdsa-ec-define-gy "$ec_gy_hex" \
    -ecdsa-ec-define-n "$ec_n_hex" \
    -ecdsa-ec-define-h "$ec_h_hex" \
    -ecdsa-private-key "$d_hex" \
```

```
    -ecdsa-signature \
    -ecdsa-signature-message "$M_hex")

R_hex=$(echo "$U" | cut -d' ' -f1)
R_decimal=$(echo "ibase=16; $R_hex" | bc)

S_hex=$(echo "$U" | cut -d' ' -f2)
S_decimal=$(echo "ibase=16; $S_hex" | bc)

# Exibir assinatura
echo "Assinatura:"
echo "R = $R_decimal"
echo "S = $S_decimal"

# Verificar a assinatura
V=$(./ecutils -ecdsa -ecdsa-ec-define \
    -ecdsa-ec-define-p "$ec_p_hex" \
    -ecdsa-ec-define-a "$ec_a_hex" \
    -ecdsa-ec-define-b "$ec_b_hex" \
    -ecdsa-ec-define-gx "$ec_gx_hex" \
    -ecdsa-ec-define-gy "$ec_gy_hex" \
    -ecdsa-ec-define-n "$ec_n_hex" \
    -ecdsa-ec-define-h "$ec_h_hex" \
    -ecdsa-verify-signature \
    -ecdsa-verify-signature-public-key-px "$Qx_hex" \
    -ecdsa-verify-signature-public-key-py "$Qy_hex" \
    -ecdsa-verify-signature-r "$R_hex" \
    -ecdsa-verify-signature-s "$S_hex" \
    -ecdsa-verify-signature-signed-message "$M_hex")

# Verificar o resultado da verificação e exibir apropriado
if [ "$V" == "1" ]; then
    echo "A assinatura é válida."
else
    echo "A assinatura é inválida."
fi
```

ANEXO E – Exemplo de implementação da soma e multiplicação de pontos em uma Curva Elíptica sobre um campo finito (Script em Python)

```
def elliptic_curve_addition(P, Q):
    """
    Realiza a adição de dois pontos em uma curva elíptica.

    Args:
        P (tuple): As coordenadas do ponto P na forma (x, y).
        Q (tuple): As coordenadas do ponto Q na forma (x, y).

    Returns:
        tuple: As coordenadas do ponto resultante da adição,
         na forma (x, y).
            Se a adição resultar no ponto no infinito,
            retorna (None, None).
    """
    (x_1, y_1) = P
    (x_2, y_2) = Q

    if P == (None, None):
        return Q

    if Q == (None, None):
        return P

    if P == Q:
        n = (3 * x_1**2 + a) % p
        d = (2 * y_1) % p
        try:
            inv = pow(d, -1, p)
        except:
            return (
                None,
                None,
            )  # Ponto no infinito
        s = (n * inv) % p
        x_3 = (s**2 - x_1 - x_1) % p
        y_3 = (s * (x_1 - x_3) - y_1) % p
```

```
        return (x_3, y_3)
    else:
        x_2, y_2 = Q
        n = (y_2 - y_1) % p
        d = (x_2 - x_1) % p
        try:
            inv = pow(d, -1, p)
        except:
            return (
                None,
                None,
            )  # Ponto no infinito
        s = (n * inv) % p
        x_3 = (s**2 - x_1 - x_2) % p
        y_3 = (s * (x_1 - x_3) - y_1) % p
        return (x_3, y_3)


def elliptic_curve_multiplication(k, G):
    """
    Realiza a multiplicação de um ponto por um escalar em
    uma curva elíptica.

    Args:
        k (int): O escalar pelo qual o ponto será multiplicado.
        G (tuple): As coordenadas do ponto G na forma (x, y),
         que é a base da curva.

    Returns:
        tuple: As coordenadas do ponto resultante da
        multiplicação, na forma (x, y).
    """
    if k == 0 or k >= n:
        raise ValueError("k não está no intervalo 0 < k < n")

    R = None

    num_bits = k.bit_length()
```

```
for i in range(num_bits - 1, -1, -1):
    if R is None:
        R = G
        continue

    if R == (None, None):
        R = G

    R = elliptic_curve_addition(R, R)

    if (k >> i) & 1:
        if R == (None, None):
            R = G
        else:
            R = elliptic_curve_addition(R, G)

return R
```

ANEXO F – Exemplo de implementação do método de Koblitz (Script em Python)

```
def koblitz_encode(message: str) -> tuple:
    """
    Codifica uma mensagem utilizando o algoritmo de
    Koblitz.

    Args:
        message: A mensagem a ser codificada.

    Returns:
        Uma tupla contendo o ponto codificado (x, y) e
        o valor de j.

    """
    # Definindo os parâmetros do algoritmo
    alphabet_size = (
        2**8
    )  # Tamanho do alfabeto, neste caso ASCII de 8 bits
    message_length = len(
        message
    )  # Tamanho da mensagem de entrada
    message_decimal = sum(
        ord(message[k]) * alphabet_size**k
        for k in range(message_length)
    )  # Convertendo a mensagem em um número decimal
    # Encontrando um ponto válido na curva de Koblitz
    d = 100
    for j in range(1, d - 1):
        x = (
            d * message_decimal + j
        ) % p  # Para cada valor de j, calcula-se um
        # valor x
        s = (
            x**3 + a * x + b
        ) % p  # Calcula-se o valor de s utilizando a
        # fórmula da curva elíptica
        if s == pow(
            s, (p + 1) // 2, p
```

```
        ):  # Verifica se s é um quadrado perfeito
            # (respeita a equação do ponto na curva)
            y = pow(
                s, (p + 1) // 4, p
            )  # Encontra o valor de y utilizando a fórmula
            # da curva elíptica
            if (y**2) % p == (
                x**3 + a * x + b
            ) % p:  # Verifica se o ponto (x, y)
                # está na curva
                break

    # Retorna o ponto codificado e o valor de j
    return (x, y), j


def koblitz_decode(point, j) -> str:
    """
    Decodifica um ponto codificado utilizando o algoritmo
     de Koblitz.

    Args:
        point: O ponto codificado (x, y).
        j: O valor de j utilizado na codificação.

    Returns:
        A mensagem decodificada.

    """
    # Decodifica o ponto (x, y) e o valor j para a mensagem
    # original
    d = 100
    message_decimal = (
        point[0] - j
    ) // d  # Calcula o valor decimal da mensagem utilizando
    # o valor de x, o valor de j e o parâmetro d
    alphabet_size = (
        2**8
    )  # Tamanho do alfabeto, neste caso ASCII de 8 bits
```

```
    decoded_message = []
    while (
        message_decimal != 0
    ):  # Converte o número decimal em uma sequência de
        # caracteres
        decoded_message.append(
            chr(message_decimal % alphabet_size)
        )
        message_decimal //= alphabet_size

    # Retorna a mensagem decodificada como uma string
    return "".join(decoded_message)


def elliptic_curve_addition(P, Q):
    """
    Realiza a adição de dois pontos em uma curva elíptica.

    Args:
        P (tuple): As coordenadas do ponto P na forma (x, y).
        Q (tuple): As coordenadas do ponto Q na forma (x, y).

    Returns:
        tuple: As coordenadas do ponto resultante da adição,
         na forma (x, y).
            Se a adição resultar no ponto no infinito,
            retorna (None, None).
    """
    (x_1, y_1) = P
    (x_2, y_2) = Q

    if P == (None, None):
        return Q

    if Q == (None, None):
        return P

    if P == Q:
        n = (3 * x_1**2 + a) % p
```

```
        d = (2 * y_1) % p
        try:
            inv = pow(d, -1, p)
        except:
            return (
                None,
                None,
            )  # Ponto no infinito
        s = (n * inv) % p
        x_3 = (s**2 - x_1 - x_1) % p
        y_3 = (s * (x_1 - x_3) - y_1) % p
        return (x_3, y_3)
    else:
        x_2, y_2 = Q
        n = (y_2 - y_1) % p
        d = (x_2 - x_1) % p
        try:
            inv = pow(d, -1, p)
        except:
            return (
                None,
                None,
            )  # Ponto no infinito
        s = (n * inv) % p
        x_3 = (s**2 - x_1 - x_2) % p
        y_3 = (s * (x_1 - x_3) - y_1) % p
        return (x_3, y_3)
```

ANEXO G – Exemplo de implementação do protocolo de ECDSA (Script em Python)

```
from random import randint


def ecdsa_generate_signature(private_key, message_hash):
    """
    Gera uma assinatura ECDSA para uma determinada chave
    privada e hash de mensagem.

    Args:
    private_key (int): A chave privada.
    message_hash (int): O hash da mensagem a ser assinada.

    Returns:
    tuple: A assinatura ECDSA (r, s).

    """
    (r, s) = (0, 0)
    while r == 0 or s == 0:
        k = randint(1, n)
        (x, _) = elliptic_curve_multiplication(k, G)
        r = x % n
        s = (
            (message_hash + r * private_key) * pow(k, -1, n)
        ) % n
    return r, s


def ecdsa_verify_signature(public_key, message_hash, r, s):
    """
    Verifica a validade de uma assinatura ECDSA.

    Args:
    public_key (tuple): O ponto de chave pública (x, y).
    message_hash (int): O hash da mensagem assinada.
    r (int): O primeiro componente da assinatura.
    s (int): O segundo componente da assinatura.
```

```
Returns:
bool: True se a assinatura é válida, False caso contrário.

"""
w = pow(s, -1, n)
u_1 = (message_hash * w) % n
u_2 = (r * w) % n
X = elliptic_curve_addition(
    elliptic_curve_multiplication(u_1, G),
    elliptic_curve_multiplication(u_2, public_key),
)
v = X[0] % n
return v == r
```