1.Introdução à criptografia assimétrica

1.1 Algoritmos assimétricos

Os algoritmos assimétricos, também chamados algoritmos de chave pública, são projetados para que a chave usada para criptografia seja diferente da chave usada para descriptografia. A chave de descriptografia não pode, em uma quantidade razoável de tempo, ser calculada a partir da chave de criptografia e vice-versa.

Algoritmos assimétricos usam uma chave pública e uma chave privada. Ambas as chaves são capazes do processo de criptografia, mas a chave emparelhada complementar é necessária para descriptografar. O processo também é reversível. Os dados criptografados com a chave pública requerem a chave privada para descriptografar. Algoritmos assimétricos alcançam confidencialidade e autenticidade usando este processo.

Exemplos de protocolos que usam algoritmos de chave assimétrica incluem:

- **Internet Key Exchange (IKE):** É um componente fundamental das redes virtuais privadas IPsec (VPNs).
- **Secure Socket Layer (SSL):** Agora isso é implementado como TLS (Transport Layer Security) padrão da IETF.
- **Secure Shell (SSH):** Este protocolo fornece uma conexão segura de acesso remoto a dispositivos de rede.
- **Pretty Good Privacy (PGP):** Este programa de computador fornece privacidade e autenticação criptográficas. É frequentemente usado para aumentar a segurança das comunicações por email.

1.2 Componentes

A criptografia assimétrica, também conhecida como criptografia de chave pública, utiliza um par de chaves distintas para criptografar e descriptografar dados. Esse par de chaves consiste em uma chave pública e uma chave privada.

1.3 Geração do par de chaves

O primeiro passo é gerar o par de chaves, composto pela chave pública e pela chave privada. Essas chaves são matematicamente relacionadas, mas computacionalmente inviáveis de serem derivadas uma da outra. **A chave pública é divulgada publicamente, enquanto a chave privada é mantida em sigilo pelo proprietário**. Cada parte da comunicação (emissor e receptor) pode ter um par de chaves pública e privada.

2.Criptografia

Para enviar uma mensagem segura para um destinatário, o remetente utiliza uma das chaves (pública ou privada) para criptografar a mensagem. Isso é feito aplicando uma função matemática específica à mensagem original, juntamente com a chave usada. O resultado é a mensagem criptografada, que só pode ser descriptografada com a outra chave (privada ou pública) correspondente.

2.1 Transmissão da mensagem criptografada

A mensagem criptografada é transmitida e somente o destinatário que possui a chave correspondente poderá descriptografá-la.

2.2 Descriptografia

Ao receber a mensagem criptografada, o destinatário utiliza a chave correspondente para descriptografar a mensagem, o que resulta na recuperação da mensagem original.

Os algoritmos assimétricos **são substancialmente mais lentos** que os algoritmos simétricos. Seu design é baseado em problemas computacionais, como fatorar números extremamente grandes ou calcular logaritmos discretos de números extremamente grandes.

Por serem lentos, algoritmos assimétricos geralmente são usados em mecanismos criptográficos de baixo volume, como assinaturas digitais e troca de chaves. No entanto, o gerenciamento de chaves de algoritmos assimétricos tende a ser mais simples que os algoritmos simétricos, porque geralmente uma das duas chaves de criptografia ou descriptografia pode ser tornada pública.

3.Criptografia assimétrica para confidencialidade

A criptografia assimétrica é usada para garantir a confidencialidade dos dados durante a transmissão entre duas partes.

Vamos entender como os dados são cifrados usando criptografia assimétrica para confidencialidade, utilizando um exemplo envolvendo Alice e Bob:

- **Geração do par de chaves:** Bob gera um par de chaves composto por uma chave pública (PubBob) e uma chave privada (PrivBob). A chave pública é compartilhada publicamente, enquanto a chave privada é mantida em sigilo por Bob.
- **Cifragem dos dados por Alice:** Alice deseja enviar dados confidenciais para Bob. Ela obtém a chave pública de Bob (PubBob) e utiliza essa chave para cifrar os dados. Alice aplica um algoritmo de criptografia assimétrica à mensagem original, produzindo uma versão cifrada da mensagem.
- **Transmissão dos dados cifrados:** Alice envia os dados cifrados para Bob. Durante a transmissão, mesmo que um terceiro (como Eve ou Mallory) intercepte a mensagem cifrada, ele não poderá decifrá-la sem acesso à chave privada correspondente (PrivBob) de Bob.
- **Decifragem dos dados por Bob:** Ao receber os dados cifrados de Alice, Bob utiliza sua chave privada (PrivBob) para decifrar a mensagem. Ele aplica um algoritmo de descriptografia assimétrica à mensagem cifrada utilizando sua chave privada, o que resulta na recuperação da mensagem original enviada por Alice.

Dessa forma, a confidencialidade dos dados é garantida, pois somente Bob, o destinatário legítimo com a chave privada, pode decifrar os dados criptografados. Os dados permanecem protegidos durante a transmissão, mesmo que sejam interceptados por terceiros.

3.1 Criptografia assimétrica para assinatura digital

As assinaturas digitais são mecanismos utilizados na criptografia para garantir a autenticidade e integridade de dados digitais. Por meio de técnicas criptográficas assimétricas, uma assinatura digital é criada pelo remetente, associada aos dados originais.

Essa assinatura digital é única e vinculada ao remetente, sendo verificada pelo destinatário utilizando a chave pública correspondente. A verificação da assinatura permite ao destinatário confirmar a autenticidade do remetente e a integridade dos dados, certificando-se de que a mensagem não foi alterada durante a transmissão.

As assinaturas digitais são amplamente utilizadas em transações eletrônicas, contratos digitais e outros cenários que requerem autenticação e confiança nos dados transmitidos. Vamos entender como os dados são cifrados usando criptografia assimétrica para assinaturas digitais, utilizando um exemplo envolvendo Alice:

- **Geração do par de chaves:** Alice gera um par de chaves composto por uma chave pública (PubAlice) e uma chave privada (PrivAlice). A chave pública é compartilhada publicamente, enquanto a chave privada é mantida em sigilo por Alice.
- **Criação da assinatura digital por Alice:** Alice deseja disponibilizar um documento (pode ser qualquer arquivo) assinado digitalmente por ela. Ela aplica uma função hash ao documento original para gerar um resumo criptográfico único e fixo. Em seguida, Alice utiliza sua chave privada (PrivAlice) para criptografar o resumo e o documento, formando o arquivo assinado digitalmente.
- **Transmissão dos dados assinados:** Alice disponibiliza o arquivo assinado digitalmente. Qualquer pessoa pode ter acesso ao arquivo.
- **Verificação da assinatura:** Ao receber o documento e a assinatura digital de Alice, a parte que vai verificar o arquivo assinado digitalmente (por exemplo Bob), utiliza a chave pública de Alice (PubAlice) para descriptografar a assinatura. Isso resulta na obtenção do documento digital e o resumo criptográfico original (da função hash). Bob, então, aplica a mesma função hash ao documento digital e compara o resumo gerado por ele com o resumo descriptografado que estava no arquivo assinado digitalmente por Alice. Se forem idênticos, a integridade do documento é confirmada e a assinatura digital é considerada válida, garantindo que foi Alice quem assinou o documento.

Dessa forma, a criptografia assimétrica é utilizada para criar uma assinatura digital única que está vinculada aos dados originais. A assinatura digital permite que Bob (ou qualquer outra pessoa) verifique a autenticidade do remetente e a integridade dos dados. Caso a assinatura seja inválida, indica que a mensagem pode ter sido modificada durante a transmissão ou que não foi enviada por Alice.

4.Desafios e considerações na implementação da criptografia assimétrica

A implementação da criptografia assimétrica apresenta alguns desafios e considerações que devem ser levados em conta:

- **Desempenho:** A criptografia assimétrica é computacionalmente mais intensiva do que a criptografia simétrica. Os algoritmos assimétricos envolvem operações matemáticas complexas, como exponenciação modular e multiplicação de grandes números. Portanto, é necessário considerar o desempenho do sistema ao implementar algoritmos assimétricos, especialmente em cenários de grande volume de dados ou em dispositivos com recursos limitados.
- **Gerenciamento de chaves:** A criptografia assimétrica requer o gerenciamento adequado das chaves públicas e privadas. É essencial garantir a segurança das chaves privadas, pois a divulgação indevida delas pode comprometer a segurança dos dados. Além disso, é necessário estabelecer mecanismos confiáveis para distribuir e armazenar as chaves públicas de forma que os usuários possam verificá-las corretamente.
- **Algoritmos e padrões confiáveis:** A escolha do algoritmo criptográfico é crucial para garantir a segurança da implementação. É necessário selecionar algoritmos amplamente aceitos, com revisões e análises independentes, que tenham resistência comprovada a ataques criptográficos conhecidos. Além disso, seguir padrões e protocolos bem estabelecidos é importante para interoperabilidade e segurança entre diferentes sistemas.
- **Criptografia híbrida:** Para combinar eficiência e segurança, é comum utilizar a criptografia assimétrica em conjunto com a criptografia simétrica. Isso é conhecido como criptografia híbrida. Nesse caso, a criptografia assimétrica é usada para estabelecer uma chave de sessão segura, que é então usada para criptografar os dados de forma eficiente com a criptografia simétrica. A implementação adequada dessa abordagem requer a sincronização adequada dos algoritmos, o gerenciamento correto das chaves e a proteção das chaves de sessão.
- **Atualização e segurança:** A criptografia assimétrica também está sujeita a avanços na criptoanálise e descoberta de novas vulnerabilidades. Portanto, é essencial acompanhar os desenvolvimentos e atualizações na área de criptografia para garantir a segurança contínua dos sistemas implementados.

5.RSA (Rivest Shamir Adleman)

O algoritmo RSA, desenvolvido por Ron Rivest, Adi Shamir e Leonard Adleman em 1977, é um dos algoritmos de criptografia assimétrica mais amplamente utilizados. Ele se baseia na dificuldade de fatorar grandes números inteiros para fornecer segurança na troca de informações. O funcionamento do algoritmo RSA é o seguinte:

- **Geração do par de chaves:** Primeiro, é gerado um par de chaves composto por uma chave pública e uma chave privada. A chave pública é compartilhada com todos, enquanto a chave privada é mantida em sigilo pelo proprietário.
- **Escolha de primos:** Em seguida, são escolhidos dois números primos grandes distintos, p e q. Esses números são mantidos em segredo.

- **Cálculo do módulo n:** O módulo n é calculado como o produto dos primos p e q, ou seja, n = p * q. Esse valor é usado como o módulo nas operações criptográficas.
- **Cálculo da função totiente de Euler:** A função totiente de Euler (φ) de n é calculada como φ(n) = (p - 1) * (q - 1). Essa função é importante para determinar a chave privada.
- **Escolha da chave pública:** Um número inteiro e relativamente primo a φ(n) é escolhido como a chave pública, geralmente chamado de "e". A chave pública consiste no par (n, e) e é compartilhada publicamente.
- **Cálculo da chave privada:** A chave privada, geralmente chamada de "d", é calculada como o inverso multiplicativo de "e" módulo φ(n). Essa chave privada é mantida em segredo.
- **Criptografia:** Para criptografar uma mensagem, o remetente a converte em um número inteiro representativo do espaço de mensagens. Em seguida, ele eleva esse número à potência "e" módulo n para obter o texto cifrado.
- **Descriptografia:** O destinatário utiliza a chave privada "d" para descriptografar o texto cifrado, elevando-o à potência "d" módulo n. O resultado é a mensagem original.

O tamanho das chaves seguras no algoritmo RSA depende da aplicação e do período de segurança desejado. Normalmente, são utilizadas chaves com tamanho de 2048 bits ou mais, consideradas seguras para a maioria dos cenários. O algoritmo RSA é amplamente utilizado em diversas tecnologias, como:

- **Criptografia de dados:** O RSA é usado para proteger a confidencialidade de dados sensíveis em comunicações seguras, como trocas de chaves em TLS/SSL, criptografia de emails e assinaturas digitais.
- **Criptografia de chave pública:** O RSA é usado para proteger chaves de criptografia simétrica em sistemas de gerenciamento de chaves, como PGP (Pretty Good Privacy) e SSH (Secure Shell).
- **Autenticação:** O RSA é usado para autenticar entidades em sistemas de autenticação de dois fatores, como tokens RSA SecurID.

6.DSA (Digital Signature Algorithm)

O algoritmo DSA (Digital Signature Algorithm) é um algoritmo de assinatura digital, uma forma de criptografia assimétrica, desenvolvido pelo Governo dos Estados Unidos. Ele é amplamente utilizado para garantir a autenticidade e integridade de mensagens e documentos digitais. O funcionamento do algoritmo DSA é o seguinte:

- **Geração do par de chaves:** Assim como em outros algoritmos de criptografia assimétrica, o DSA utiliza um par de chaves composto por uma chave pública e uma chave privada. A chave pública é divulgada publicamente, enquanto a chave privada é mantida em segredo.
- **Escolha de parâmetros:** Antes de gerar as chaves, são escolhidos os parâmetros do algoritmo, que incluem um número primo grande (p), um número primo menor (q) que é um divisor de p-1, e um gerador (g) que é uma raiz primitiva módulo p. Esses

parâmetros são fixos para um conjunto de chaves e devem ser compartilhados entre o emissor e o receptor.

- **Geração da chave privada:** A chave privada é gerada selecionando aleatoriamente um número inteiro x, que deve ser mantido em segredo pelo proprietário.
- **Cálculo da chave pública:** A chave pública é calculada como y = g^x mod p, onde "^" representa a operação de exponenciação. O valor de y é a chave pública correspondente à chave privada x.
- **Criação da assinatura digital:** Para criar uma assinatura digital em uma mensagem, o emissor utiliza sua chave privada para calcular um valor chamado "assinatura". Isso é feito selecionando aleatoriamente um número inteiro k e calculando r = (g^k mod p) mod q, onde r é um componente da assinatura.
- **Verificação de assinatura:** O receptor recebe a mensagem e a assinatura digital correspondente. Ele utiliza a chave pública do emissor para calcular dois valores, s1 e s2. O valor s1 é calculado como s1 = (h(m) / r) mod q, onde h(m) é o resumo criptográfico (hash) da mensagem. O valor s2 é calculado como s2 = (k^-1 * (h(m) + x * s1)) mod q, onde k^-1 é o inverso multiplicativo de k módulo q. Se os valores de s1 e s2 coincidirem com a assinatura original, a assinatura é considerada válida.

O tamanho das chaves seguras no algoritmo DSA é geralmente de 1024 bits ou mais, dependendo dos requisitos de segurança. O DSA é amplamente utilizado em tecnologias como:

- **Criptografia de e-mails:** O DSA é utilizado para assinar digitalmente mensagens de emails, garantindo a autenticidade do remetente e a integridade da mensagem.
- **Certificados digitais:** O DSA é usado para criar assinaturas digitais em certificados digitais, que são utilizados em infraestruturas de chaves públicas (PKIs) para autenticação e segurança em comunicações online.
- **Protocolos de segurança:** O DSA é utilizado em diversos protocolos de segurança, como o protocolo de segurança de camada de transporte (TLS/SSL) para estabelecer conexões seguras na Internet.

O algoritmo DSA oferece um mecanismo eficiente e seguro para a geração de assinaturas digitais, garantindo a autenticidade e integridade de mensagens e documentos digitais.

7.Diffie-Hellman

O algoritmo Diffie-Hellman é um protocolo de troca de chaves que permite que duas partes estabeleçam uma chave secreta compartilhada, mesmo que estejam se comunicando por um canal inseguro. Ele foi desenvolvido por Whitfield Diffie e Martin Hellman em 1976 e é amplamente utilizado em criptografia de chave pública. O funcionamento do algoritmo Diffie-Hellman é o seguinte:

- **Escolha dos parâmetros:** Antes de iniciar a troca de chaves, as partes devem concordar com os parâmetros do algoritmo. Isso inclui a escolha de um número primo grande (p) e um número gerador (g), que é um elemento do grupo multiplicativo

módulo p. Esses parâmetros são fixos para um conjunto de chaves e devem ser compartilhados entre as partes.

- **Geração das chaves privadas:** Cada parte gera sua própria chave privada, que é um número aleatório e exclusivo para cada parte. Vamos chamar essas chaves privadas de "a" e "b".
- **Cálculo das chaves públicas:** Cada parte calcula sua chave pública, que é obtida elevando o número gerador "g" à potência da chave privada correspondente e realizando a operação de módulo "p". Ou seja, a chave pública de Alice é A = g^a mod p e a chave pública de Bob é B = g^b mod p.
- **Troca das chaves públicas:** Alice e Bob trocam suas chaves públicas pela rede insegura.
- **Cálculo da chave secreta compartilhada:** Utilizando a chave pública recebida e sua própria chave privada, cada parte realiza o cálculo da chave secreta compartilhada. Alice calcula a chave secreta compartilhada como S = B^a mod p, enquanto Bob calcula a chave secreta compartilhada como S = A^b mod p. Ambos chegam ao mesmo resultado, a chave secreta compartilhada S.

A segurança do algoritmo Diffie-Hellman está baseada na dificuldade do problema do logaritmo discreto. É computacionalmente inviável para um atacante calcular a chave secreta compartilhada S apenas conhecendo as chaves públicas e os parâmetros p e g.

O tamanho das chaves seguras no algoritmo Diffie-Hellman é geralmente de 2048 bits ou mais, dependendo dos requisitos de segurança. O Diffie-Hellman é utilizado em várias tecnologias, como:

- **Protocolo TLS/SSL:** O Diffie-Hellman é utilizado no protocolo de segurança de transporte (TLS/SSL) para estabelecer chaves compartilhadas entre um servidor e um cliente, permitindo a comunicação segura pela internet.
- **Protocolo SSH:** O Diffie-Hellman é usado no protocolo de acesso remoto seguro (SSH) para estabelecer chaves compartilhadas entre um cliente e um servidor, garantindo a autenticação e confidencialidade das comunicações.
- **VPN:** Em redes privadas virtuais (VPNs), o Diffie-Hellman é empregado para estabelecer chaves compartilhadas entre os dispositivos, permitindo a comunicação segura e a criação de túneis criptografados.
- **Criptografia de mensagens:** O Diffie-Hellman é utilizado para a troca de chaves em criptografia de chave simétrica, onde as chaves de criptografia são geradas de forma segura e compartilhadas entre as partes.

O algoritmo Diffie-Hellman é uma ferramenta fundamental na área de criptografia, permitindo a troca segura de chaves em ambientes inseguros, contribuindo para a confidencialidade e privacidade das comunicações.

8.ElGamal

O algoritmo ElGamal é um algoritmo de criptografia assimétrica que combina a criptografia de chave pública e a troca de chaves de Diffie-Hellman.

Ele é amplamente utilizado para garantir a confidencialidade das comunicações e proteger a privacidade dos dados. O funcionamento do algoritmo ElGamal é o seguinte:

- **Geração do par de chaves:** Assim como em outros algoritmos de criptografia assimétrica, o ElGamal utiliza um par de chaves composto por uma chave pública e uma chave privada. A chave pública é divulgada publicamente, enquanto a chave privada é mantida em segredo.
- **Escolha de parâmetros:** Antes de gerar as chaves, são escolhidos os parâmetros do algoritmo, que incluem um número primo grande (p) e um gerador (g) que é um elemento do grupo multiplicativo módulo p. Esses parâmetros são fixos para um conjunto de chaves e devem ser compartilhados entre o emissor e o receptor.
- **Geração da chave privada:** A chave privada é gerada selecionando aleatoriamente um número inteiro x, que deve ser mantido em segredo pelo proprietário.
- **Cálculo da chave pública:** A chave pública é calculada como y = g^x mod p, onde "^" representa a operação de exponenciação. O valor de y é a chave pública correspondente à chave privada x.
- **Criptografia:** Para criptografar uma mensagem, o emissor seleciona aleatoriamente um número inteiro k e calcula dois valores. O primeiro valor é r = g^k mod p, que é o componente de aleatoriedade. O segundo valor é c = (m * y^k) mod p, onde m é o valor numérico da mensagem original. O par (r, c) é a mensagem cifrada que será enviada ao receptor.
- **Descriptografia:** O receptor recebe a mensagem cifrada (r, c) e utiliza sua chave privada x para calcular o valor inverso de r, que é r^-x mod p. Em seguida, ele recupera a mensagem original m utilizando a fórmula m = (c * r^-x) mod p.

O tamanho das chaves seguras no algoritmo ElGamal geralmente é de 2048 bits ou mais, dependendo dos requisitos de segurança. O ElGamal é utilizado em tecnologias como:

- **Criptografia de e-mails:** O ElGamal é usado para criptografar emails, garantindo a confidencialidade das mensagens durante a transmissão.
- **Compartilhamento seguro de chaves:** O ElGamal é aplicado em protocolos de compartilhamento seguro de chaves, como o protocolo Diffie-Hellman, para estabelecer chaves compartilhadas entre duas partes sem que a chave seja exposta durante a troca.
- **Criptografia de arquivos:** O ElGamal é empregado em sistemas de criptografia de arquivos para proteger a privacidade e a confidencialidade dos dados armazenados.

O algoritmo ElGamal oferece uma abordagem segura e eficiente para criptografia de chave pública, fornecendo confidencialidade e privacidade na comunicação.

9.ECC (Elliptic Curve Cryptography)

O algoritmo ECC (Elliptic Curve Cryptography) é um sistema criptográfico que se baseia na teoria das curvas elípticas sobre corpos finitos. Nesse algoritmo, a chave pública

e a chave privada são geradas com base em operações matemáticas na curva elíptica. A curva elíptica utilizada é definida por uma equação matemática específica.

A forma geral da equação de uma curva elíptica é: $y^2 = x^3 + ax + b$, onde a e b são constantes que determinam a forma e a posição da curva. A escolha adequada dos parâmetros a e b é fundamental para garantir a segurança e a eficiência do algoritmo ECC.

Uma característica importante das curvas elípticas é que elas possuem um grupo aditivo associado a elas. Essa propriedade permite realizar operações matemáticas, como adição e multiplicação, entre pontos na curva. Essas operações são utilizadas na geração de chaves e no processo de criptografia e descriptografia.

No contexto do ECC, a segurança é baseada na complexidade do problema matemático conhecido como "***problema do logaritmo discreto***". Esse problema envolve encontrar o valor de um expoente desconhecido quando dado uma base e o resultado da exponenciação. A complexidade desse problema torna o ECC resistente a ataques de força bruta e algoritmos de fatoração.

Diferente de outros algoritmos de criptografia assimétrica, como o RSA e o DSA, o ECC oferece um nível de segurança similar com chaves significativamente menores. O funcionamento do algoritmo ECC é o seguinte:

- **Escolha da curva elíptica:** Antes de iniciar a geração de chaves, é necessário escolher uma curva elíptica adequada. Existem várias curvas elípticas disponíveis, cada uma com diferentes níveis de segurança. As curvas mais comumente utilizadas são definidas sobre corpos finitos primos.
- **Geração das chaves privada e pública:** Assim como em outros algoritmos de criptografia assimétrica, cada parte gera sua própria chave privada, que é um número aleatório e exclusivo para cada parte. A chave privada é geralmente um número inteiro gerado dentro de um intervalo seguro. A chave pública é obtida multiplicando a chave privada pelo ponto gerador da curva elíptica.
- **Troca das chaves públicas:** Às partes enviam suas chaves públicas para a outra parte de forma segura.
- **Cálculo da chave secreta compartilhada:** Utilizando a chave pública recebida e sua própria chave privada, cada parte realiza o cálculo da chave secreta compartilhada. Isso envolve a multiplicação da chave pública pela chave privada da outra parte.

O ECC oferece uma segurança equivalente a outros algoritmos de criptografia assimétrica, mas com chaves muito menores. Em geral, uma chave ECC de 256 bits é considerada segura o suficiente para a maioria das aplicações, enquanto um nível de segurança semelhante com outros algoritmos requer chaves de 2048 bits ou mais.

O algoritmo ECC é utilizado em várias tecnologias, como:

- **Criptografia de curva elíptica em TLS/SSL:** O ECC é suportado como um método de criptografia em protocolos de segurança de transporte (TLS/SSL), oferecendo uma alternativa mais eficiente e segura em comparação com algoritmos tradicionais.
- **Smart cards e dispositivos de segurança:** Devido ao seu baixo consumo de energia e requisitos de armazenamento, o ECC é amplamente utilizado em smart cards e outros dispositivos de segurança para autenticação e assinatura digital.
- **Internet das coisas (IoT):** O ECC é uma escolha popular para criptografia em dispositivos IoT devido aos seus requisitos de recursos reduzidos e capacidade de oferecer segurança adequada mesmo em dispositivos de baixo poder de processamento.

O ECC se tornou uma alternativa viável e eficiente para algoritmos de criptografia assimétrica, oferecendo segurança sólida com tamanhos de chaves menores, o que é especialmente importante em ambientes com restrições de recursos.