# Java Advanced

Prof. Dr. Marcel Stefan Wagner

## Aula 12 – Criptografia Simétrica e Assimétrica

FIAP

# Criptografia de chave Simétrica e Assimétrica com Java

## Aula 12

**Prof. Dr. Marcel Stefan Wagner**

Curso **Análise e Desenvolvimento de Sistemas**

**FIAP** – Faculdade de Informática e Administração Paulista

Campus – **Lins** / **Paulista**

Curso de **graduação – 2025/2**

Turma – **TDS**

# Criptografia

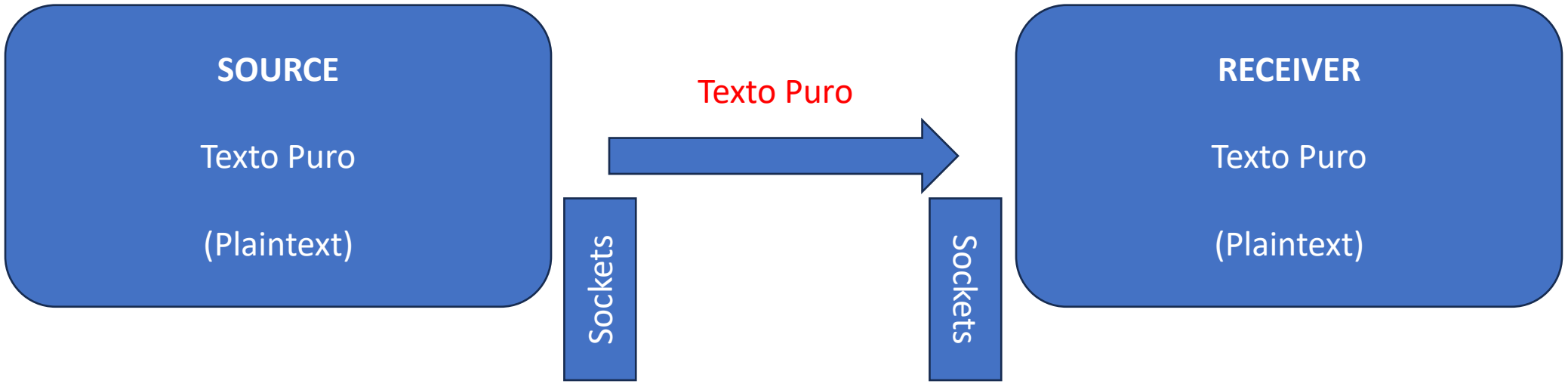
# Criptografia

- Um dos recursos utilizados para manter a segurança de sistemas distribuídos é a criptografia.
- Mesmo que mensagens enviadas entre diferentes sistemas sejam interceptadas, a segurança é mantida.

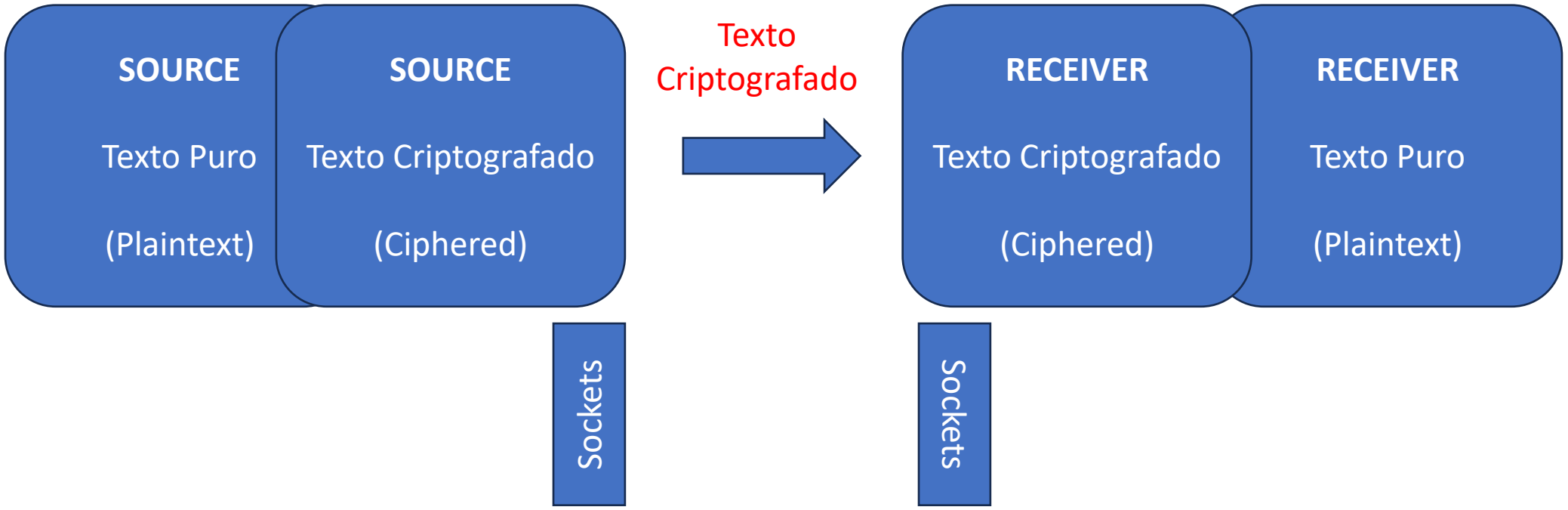
# Criptografia

- Na criptografia, tentamos converter um **texto puro** para um **texto cifrado**.
- O texto puro só pode ser lido pelas pessoas autorizadas.
- As pessoas não autorizadas só possuem acesso ao texto cifrado.
- Esse texto é incompreensível.

# Arquivo em uma comunicação sem Criptografia



# Arquivo em uma comunicação com Criptografia



# Criptografia

- O texto cifrado é obtido usando um algoritmo de criptografia (conhecido).
- O texto cifrado é traduzido utilizando uma chave de criptografia (secreta).
- O **princípio de Kerckhoffs** estabele que o segredo deve estar contido exclusivamente na chave e os algoritmos de criptografia devem ser públicos.

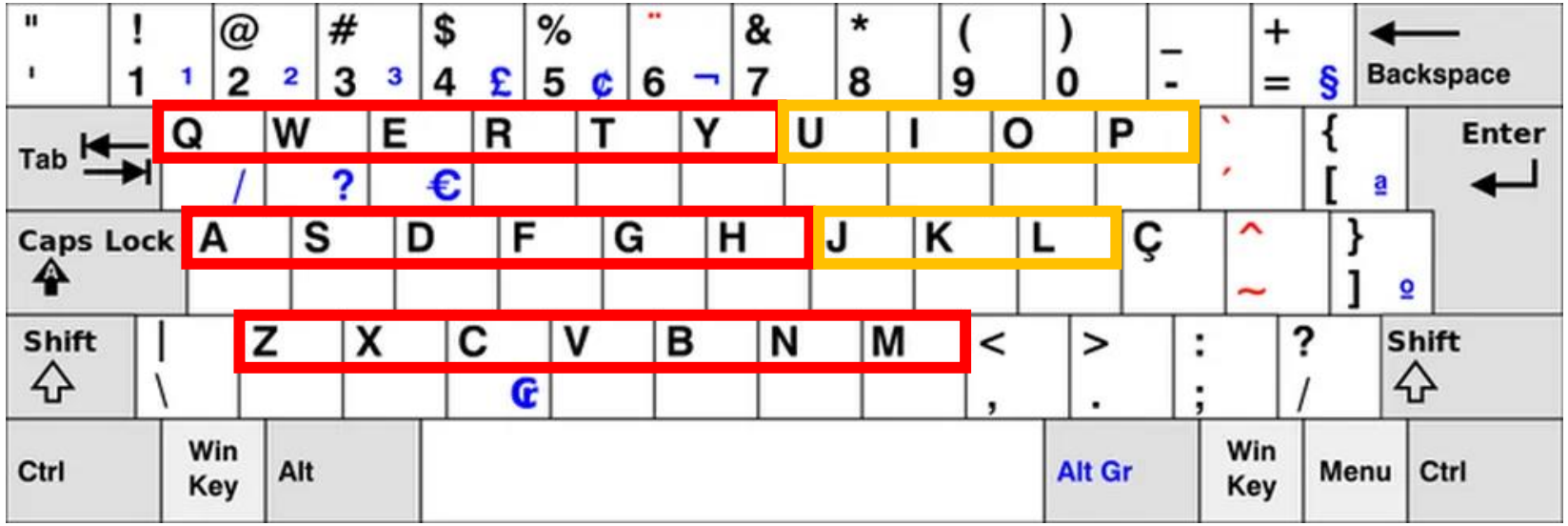
# Criptografia de Chave Simétrica

# Criptografia de chave privada

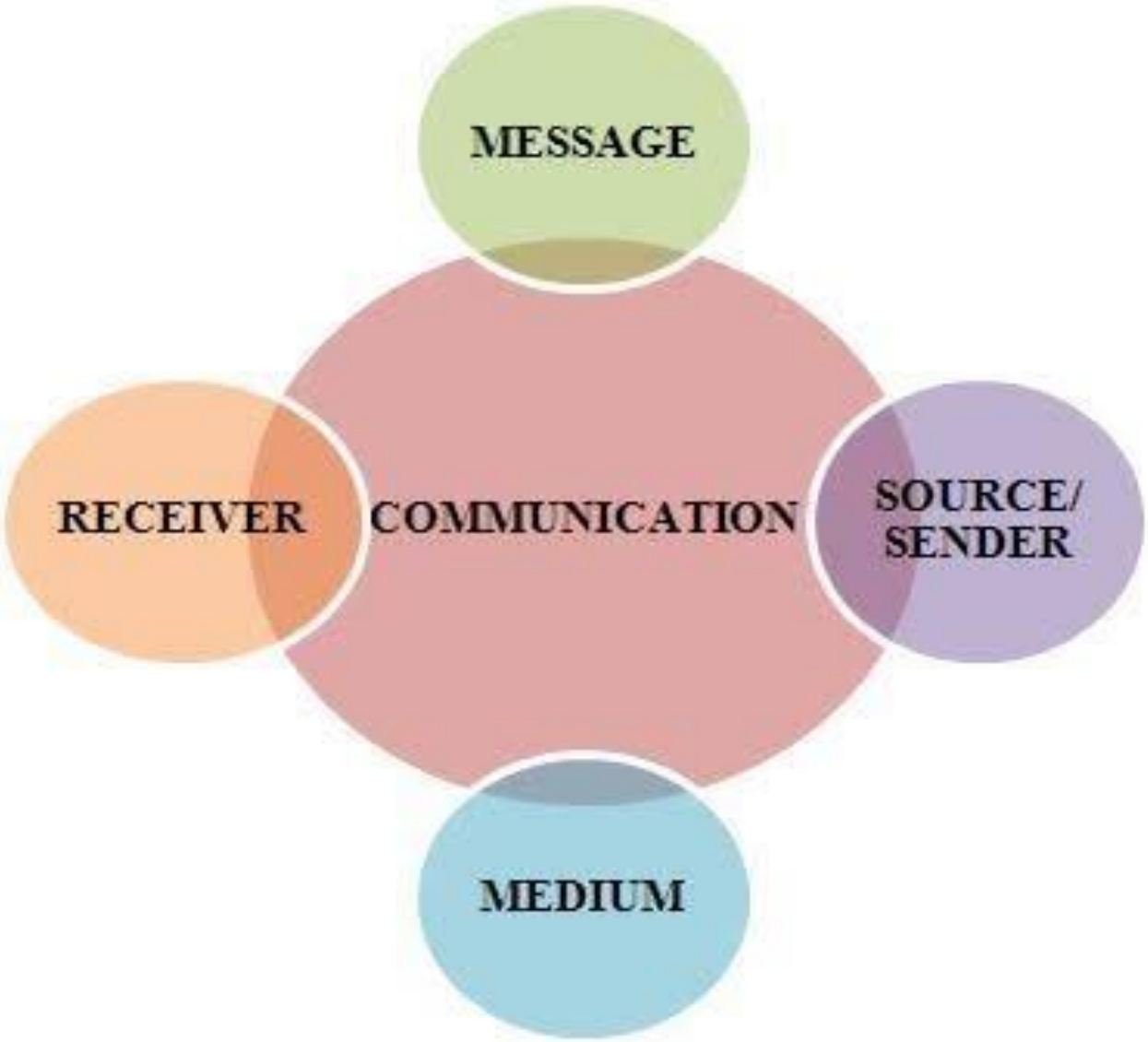
- Considere o seguinte texto puro e sua cifra:
  - puro: ABCDEFGHIJKLMNOPQRSTUVWXYZ
  - cifrado: QWERTYUIOPASDFGHJKLZXCVBNM
- Nesse caso, a chave privada é a cadeia de 26 letras que corresponde ao alfabeto.
- Se a chave for descoberta é fácil decodificar qualquer mensagem.
- Esse tipo de criptografia é chamada **criptografia de chave simétrica**.

# Criptografia de chave privada

- Considere o seguinte texto puro e sua cifra:
  - puro: ABCDEFGHIJKLMNOPQRSTUVWXYZ
  - cifrado: QWERTYUIOPASDFGHJKLZXCVBNM

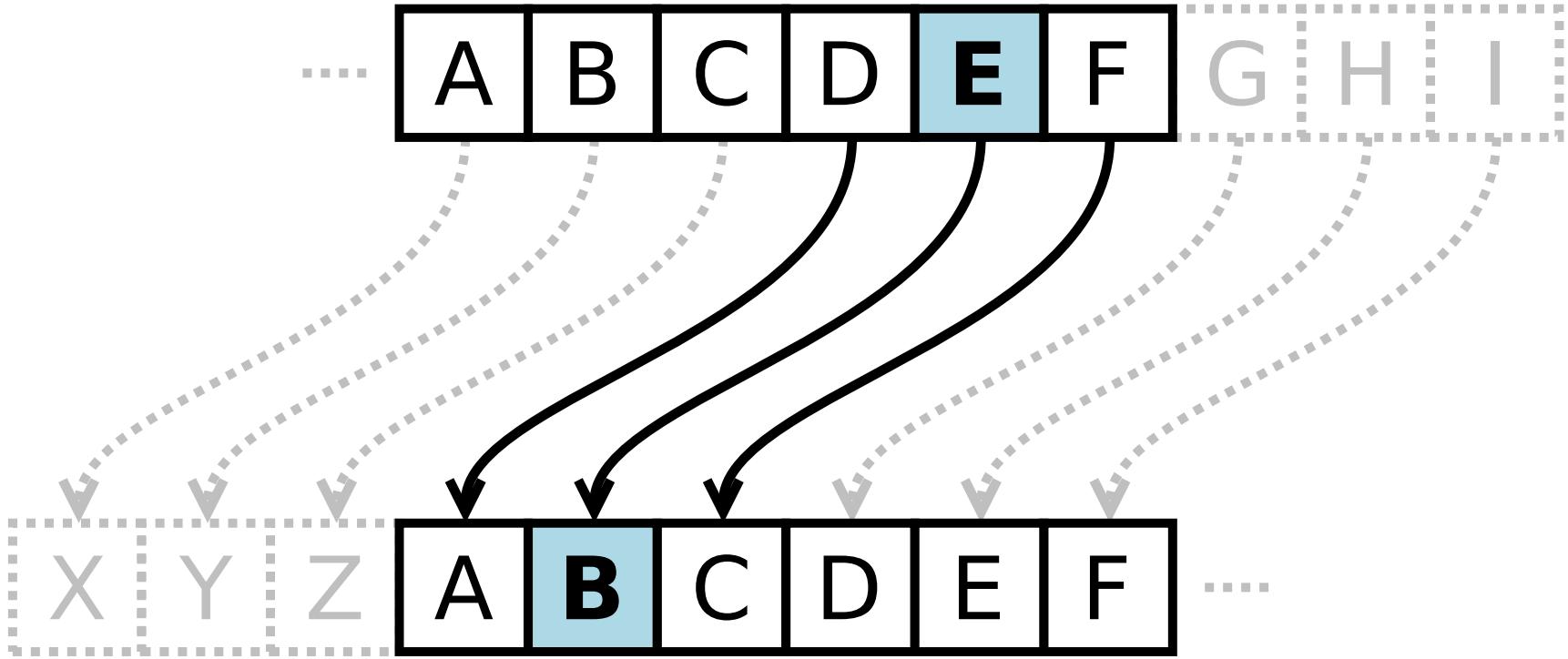


# Visão Geral de uma comunicação



## Cifra de César

Caesar Cipher ou Cifra de César é um algoritmo de criptografia que executa o deslocamento de posições do alfabeto para a transmissão de dados (deslocamento à esquerda ou à direita, com um fator de deslocamento que é 3 posições).



# Criptografia de Chave Pública

# Criptografia de Chave Pública

- Na criptografia de chave simétrica (privada), tanto o receptor quanto o remetente precisam ter em mãos a chave privada.
- A **criptografia de chave pública ou assimétrica** tenta contornar esse problema.
- Nela, as chaves para criptografar e para traduzir a mensagem são diferentes.
- Sob essa circunstância a chave para criptografar a mensagem pode ser pública, mas a chave para traduzi-la deve ser privada.

# Criptografia de Chave Pública

- A encriptação faz uso de uma operação fácil, mas a decriptação sem a chave exige uma operação complexa (que provavelmente não pode ser terminada em uma vida).
  - Operação fácil: multiplicação.
  - Operação difícil: fatoração de primos.

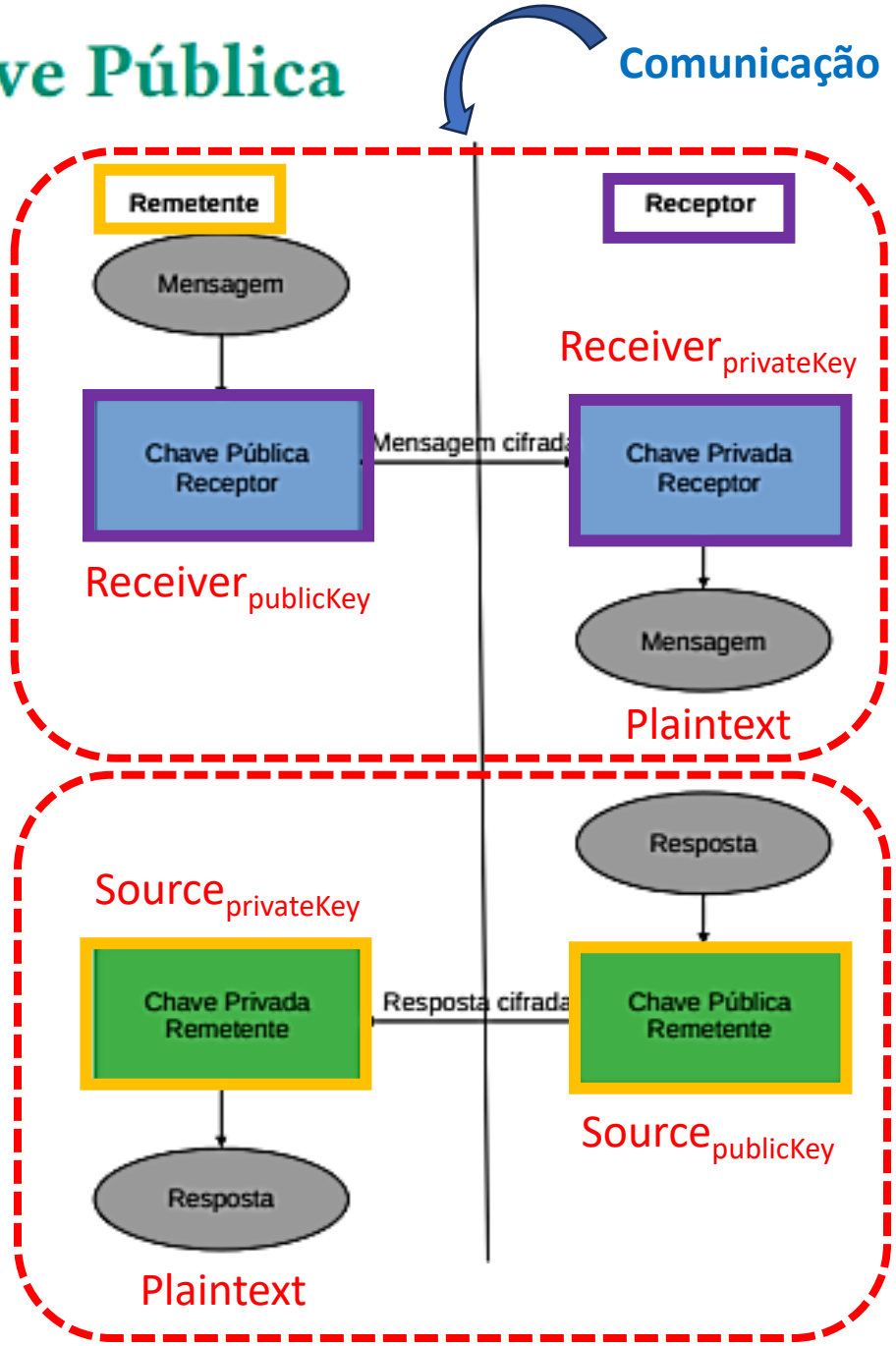
# Criptografia de Chave Pública

- Na criptografia de chave assimétrica, um correspondente encripta a mensagem usando a chave pública do receptor.
  - Apenas o receptor poderá descriptografar essa mensagem já que ele é o único que possui a chave privada.

# Criptografia de Chave Pública

Procedimento de envio de informação de uma mensagem criptografada

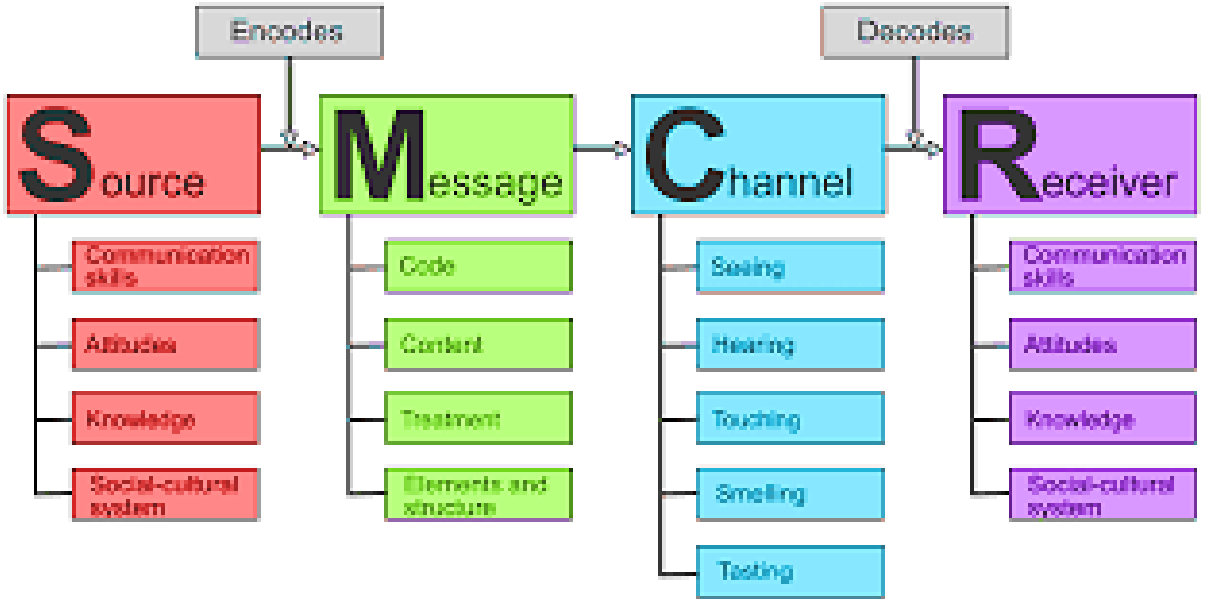
Procedimento de recebimento de informação de uma mensagem criptografada




No início da comunicação, tanto o **Remetente** (*Sender* ou *Source*) quanto o **Receptor** (*Receiver* ou Destino), geram as suas **chaves Pública e Privada**.

Depois de geradas as chaves, tanto o **Remetente** (*Sender* ou *Source*) quanto o **Receptor** (*Receiver* ou Destino), trocam (transmitem) as suas **chaves Públicas**.

# Visão Geral de comunicação *Source-Receiver*



# Exercício de Criptografia de Chave Assimétrica

Exercício 1

Construa uma aplicação em linguagem Java que funcione apenas de forma local, ou seja, apenas no seu *host*, para realizar a criptografia (cifragem) de uma mensagem que deve ser inserida via terminal/console.

O seu App deve ser capaz de receber como informação a mensagem em forma de texto, realizar a criptografia de chave assimétrica (uso de chave Pública e Privada) e mostrar a mensagem que foi recebida e a criptografada no terminal/console.

# Exemplo em Java

```java
public static KeyPair gerarChavesPublicoPrivada() throws NoSuchAlgorithmException{
   KeyPairGenerator geradorChave = KeyPairGenerator.getInstance("RSA");
   geradorChave.initialize(2048);
   KeyPair par = geradorChave.generateKeyPair();
   return par;
}

public static String
      cifrar(String mensagem, PublicKey publicKey) throws Exception {

   byte[] messageToBytes = mensagem.getBytes();
   Cipher cifrador = Cipher.getInstance("RSA/ECB/PKCS1Padding");

   // Cifrar mensagem
   cifrador.init(Cipher.ENCRYPT_MODE, publicKey);
   byte[] bytesCripto = cifrador.doFinal(messageToBytes);

   return Base64.getEncoder().encodeToString(bytesCripto);
}
```

# Exemplo em Java

```
public static String
     decifrar(String mensagem, PrivateKey privateKey) throws Exception {

   // Converte a mensagem cifrada para bytes
   byte[] bytesCifrados = Base64.getDecoder().decode(mensagem);
   Cipher cifrador = Cipher.getInstance("RSA/ECB/PKCS1Padding");

   // Decriptografa os bytes
   cifrador.init(Cipher.DECRYPT_MODE, privateKey);
   byte[] mensagemDecifrada = cifrador.doFinal(bytesCifrados);

   // Cria os bytes da mensagem decifrada para uma string
   return new String(mensagemDecifrada, "UTF8");
}

// Converte os bytes de uma chave pública enviado pelo socket de volta para a chave
public static PublicKey bytesParaChave(byte[] bytesChave) throws Exception {
   X509EncodedKeySpec keySpec = new X509EncodedKeySpec(bytesChave);
   KeyFactory keyFactory = KeyFactory.getInstance("RSA");
   return keyFactory.generatePublic(keySpec);
}
```

# Exemplo em Java

```
public static void main(String[] args) {
   try {
      Scanner sc = new Scanner(System.in);

      KeyPairGenerator geradorChave = KeyPairGenerator.getInstance("RSA");
      geradorChave.initialize(2048); // Iniciamos uma chave de 2048 bits
      KeyPair par = geradorChave.generateKeyPair(); // Gera um par chave pública e
          privada

      System.out.println("Digite a mensagem secreta:");
      String segredo = sc.nextLine();

      PrivateKey privateKey = par.getPrivate();
      PublicKey publicKey = par.getPublic();
```

# Exemplo em Java

```
            try {
                String mensagemCifrada = Criptografia.cifrar(segredo, publicKey);
                System.out.println("Essa éa mensagem cifrada:\n" + mensagemCifrada);

                String mensagemDecifrada = Criptografia.decifrar(mensagemCifrada,
                    privateKey);
                System.out.println("A mensagem: " + mensagemDecifrada + " foi decifrada
                    com sucesso.");
            }catch (Exception e) {
                System.err.println(e.getMessage());

            }finally {
                sc.close();
            }
        }catch (Exception e) {
            System.err.println(e.getMessage());
        }
    }
```

# Exemplo em Java

## Resposta:

Digite a mensagem secreta:
Hello World
Essa é a mensagem cifrada:
b/NV3Kd22fiRJ5klEHhgLgc9zJeDs2L9DGvb9x1bW5RwwirMAjXds...
A mensagem: Hello World foi decifrada com sucesso.

# O que Aconteceu?

- Utilizamos o algoritmo de chave assimétrica RSA para cifrar uma mensagem.
  - Primeiro geramos uma chave pública e uma privada aleatórias.
  - Utilizamos essas chaves, respectivamente, para cifrar e decifrar a mensagem.
- Utilizamos o algoritmo RSA para gerar as chaves.

# O que Aconteceu?

- O RSA trabalha gerando números primos muito grandes.
  - Em nosso caso, números de 2048 bits (números com 617 casas).
- As chaves pública e privada são calculadas com base em operações realizadas nos números primos gerados.
  - Operações para encontrar a chave privada são muito custosas.

# O que Aconteceu?

- As mensagens cifradas possuem tamanho constante.
  - Não importa o tamanho da mensagem enviada, a mensagem cifrada sempre terá o mesmo tamanho.
  - Impossível decifrá-la sem a chave privada.

# Chave Pública

```
Sun RSA public key, 2048 bits
  params: null
  modulus: 18783725138160042491088895584092771759374611948499813677552633632534165793849034003110284705
401127212234888223158933107138988968148555295172910997522685950190739676989458044774946837193036596691 8
825278931055879284511113645796330721576305690034568489230161569945947736016545002955502639371890848710 6
102667601226120317687521796146528313650027003745578135857630701422229729767392494902563126173786659063 6
632209483752589456354021530340160476274352106816929877726767886342927749709059399389922425521234682174 5
325792621941371815239792295404354128862609062173162254942128795122425716548205119908590648280588516312 3
5726538413
  public exponent: 65537
```

# Chave Privada

```
SunRsaSign RSA private CRT key, 2048 bits
  params: null
  modulus: 18783725138160042491088895584092771759374611948499813677552633632534165793849034003110284705
401127212234888223158933107138988968148555295172910997522685950190739676989458044774946837193036596691 8
825278931055879284511113645796330721576305690034568489230161569945947736016545002955502639371890848710 6
102667601226120317687521796146528313650027003745578135857630701422229729767392494902563126173786659063 6
632209483752589456354021530340160476274352106816929877726767886342927749709059399389922425521234682174 5
325792621941371815239792295404354128862609062173162254942128795122425716548205119908590648280588516312 3
5726538413
  private exponent: 109623559071111784976298591375922049262639449136552004751366881483309698839302661 48
144745249434400622756000499860885812317500801894287851591825988802919300989580109784212169698363768399 4
176953890356480851278708343986004047243245409287965581647340521548929570668875445498087076467043799791 8
856800713217698462035766184409200087434647729034425479596642917640211520660604758382843197078162218178 0
793157031229665835709274903755021566785264748602298356399176334821793038674564746443740587499583078766 7
482389396178478282089479226112503382294742572241894271901257979826033755051240375007563501768749664967 6
6068154984466826 25
```

- Não mostre isso para os outros.

# Chaves e Sistemas Distribuídos

- Vimos que o receptor precisa da chave pública do recipiente para cifrar as mensagens (e vice-versa).
  - A chave pública do recipiente é gerada no recipiente.
  - Como o receptor pode ter acesso a essa chave?
  - A chave é pública...

# Chaves e Sistemas Distribuídos

- Podemos simplesmente enviar a chave pública para o outro lado.
  - Ela só vai servir para cifrar mensagens, não vai ajudar a decifrar nenhuma.
  - Isso pode ser feito utilizando sockets.

# X.509

## Serviço de autenticação X.509

X.509 é um certificado digital que é construído sobre um padrão amplamente confiável conhecido como ITU (*International Telecommunication Union*) X.509 *standard*, no qual o formato dos certificados PKI (*Public Key Infrastructure*) é definido.

O certificado digital X.509 é uma estrutura de segurança de autenticação baseada em certificado que pode ser usada para fornecer processamento de transações seguras e informações privadas. Eles são usados principalmente para lidar com a segurança e identidade em redes de computadores e comunicações baseadas na Internet.

## Funcionamento do certificado de serviço de autenticação X.509

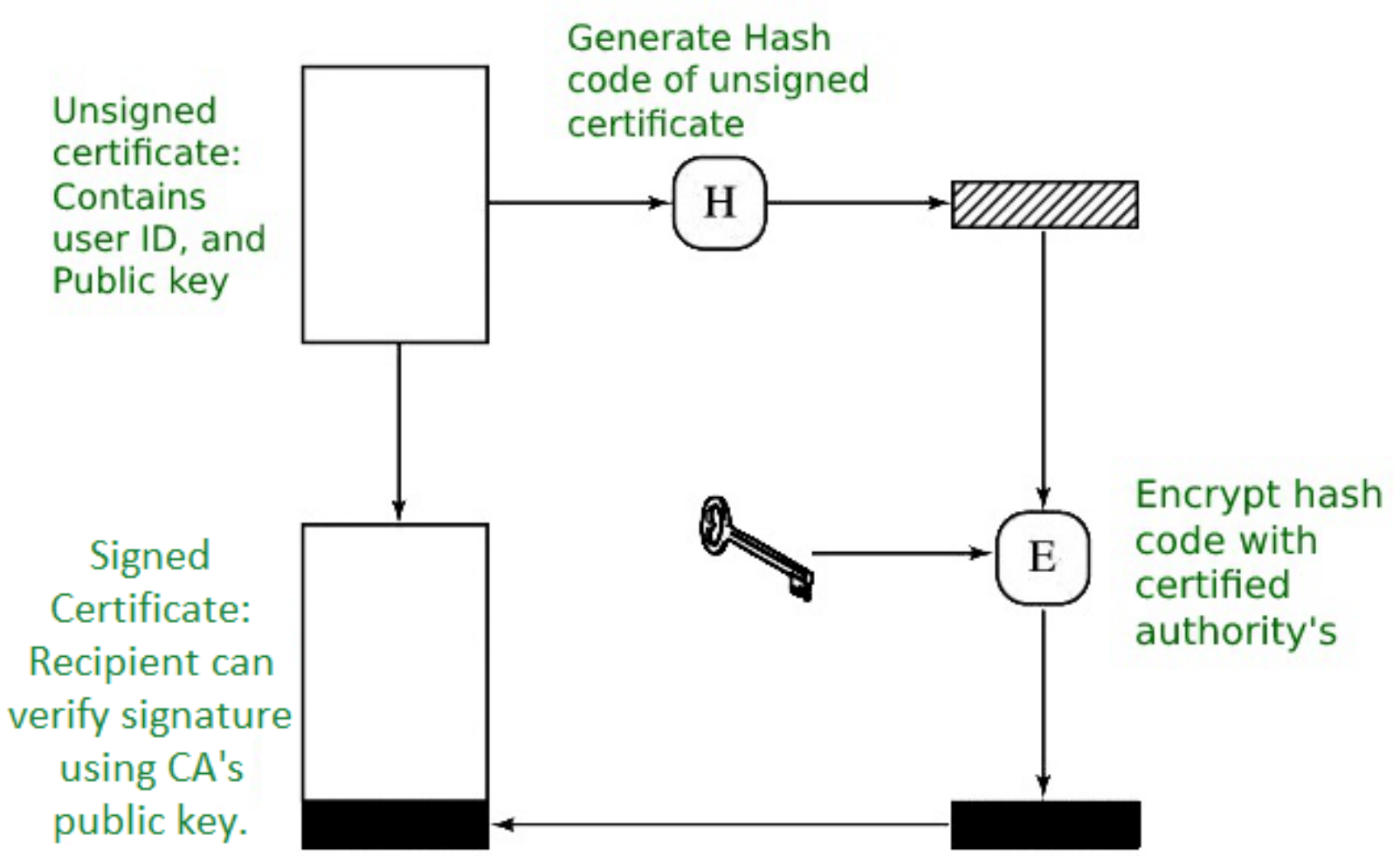
O núcleo do serviço de autenticação X.509 é o certificado de chave pública conectado a cada usuário. Esses certificados de usuário são supostamente produzidos por uma autoridade de certificação confiável e posicionados no diretório pelo usuário ou pela autoridade certificadora. Esses servidores de diretório são usados apenas para fornecer um local de fácil acesso para todos os usuários, para que eles possam adquirir certificados.

O padrão X.509 é construído em uma IDL (*Interface Description Language*) conhecida como ASN.1 (*Abstract Syntax Notation*). O formato do certificado X.509 usa um par de chaves pública e privada associado para criptografar e descriptografar uma mensagem.

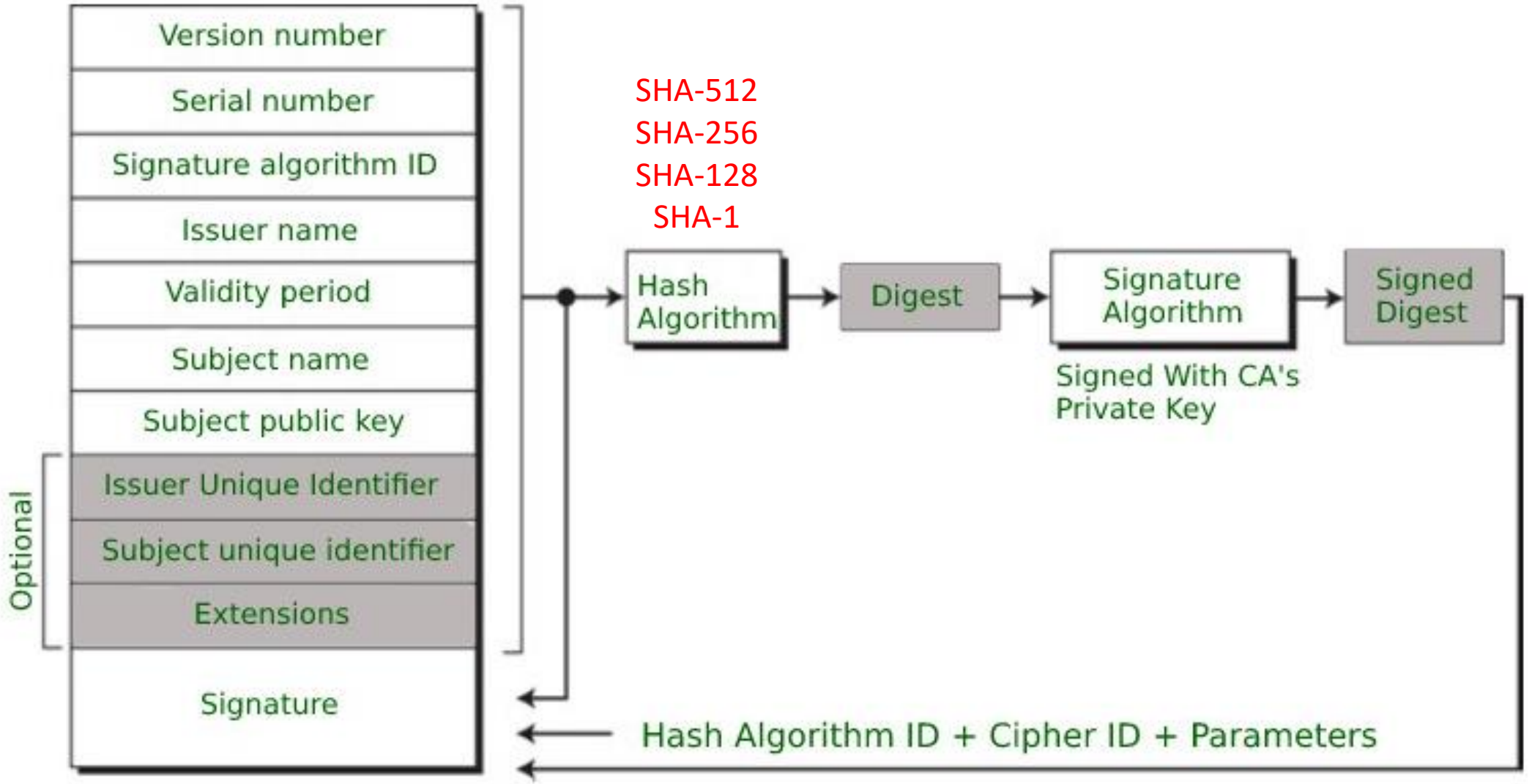
## Funcionamento do certificado de serviço de autenticação X.509

Uma vez que um certificado X.509 é fornecido a um usuário pela autoridade certificadora, esse certificado é anexado a ele como um cartão de identidade. As chances de alguém roubá-lo ou perdê-lo são menores, ao contrário de outras senhas não seguras. Com a ajuda dessa analogia, é mais fácil imaginar como essa autenticação funciona: o certificado é basicamente apresentado como uma identidade no recurso que requer autenticação.

# Funcionamento do certificado de serviço de autenticação X.509



# Formato do certificado de serviço de autenticação X.509



# RSA

## O que é o RSA?

O RSA é o algoritmo de chave pública mais usado no mundo. As iniciais RSA significam *Rivest Shamir Adleman*, em homenagem ao matemático e dois cientistas da computação que descreveram publicamente o algoritmo em 1977.

Muitos protocolos, como *Secure Shell* (SSH), SSL-TLS, S/MIME e OpenPGP, contam com criptografia RSA e funções seguras de assinatura digital.

O sistema de criptografia RSA resolveu o que já foi um problema significativo na criptografia: como enviar uma mensagem codificada para alguém sem compartilhar previamente o código com eles.

## Vantagens do RSA

**Segurança**: a criptografia RSA tem um algoritmo seguro que protege a transmissão de dados.

**Criptografia de chave pública**: o RSA usa um algoritmo de criptografia de chave pública para segurança. Isso significa que usa duas chaves diferentes para criptografar e descriptografar dados.

**Troca de chaves**: como o algoritmo RSA utiliza duas chaves para criptografia e descriptografia, é possível trocar chaves secretas sem realmente enviar a chave privada pela rede. Ele permite a criptografia e transmissão segura de dados sem enviar chaves de descriptografia com antecedência.

## Vantagens do RSA

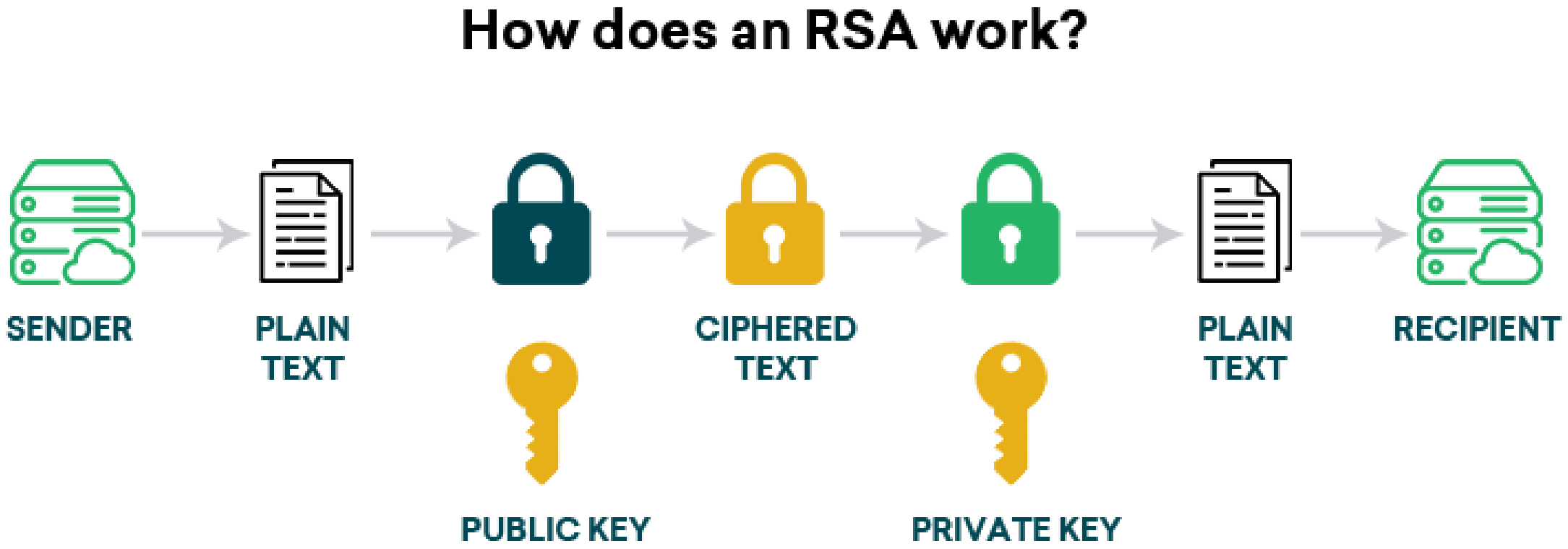
**Assinaturas digitais**: o algoritmo RSA é ideal para assinaturas digitais, pois o remetente pode assinar um documento ou mensagem usando uma chave privada, enquanto o destinatário verifica a assinatura usando uma chave pública.

## Desvantagens do RSA

**Velocidade de processamento lenta**: o algoritmo RSA tem uma velocidade de processamento lenta em comparação com outros algoritmos de criptografia ao lidar com grandes quantidades de dados. Não é sempre adequado para aplicações que exigem criptografia e descriptografia regulares de grandes volumes de dados.

**Tamanho de chave grande**: a criptografia RSA exige o uso de chaves de tamanho grande para garantir a segurança. Portanto, ele exige mais potência computacional, recursos e armazenamento.

# Visão Geral do RSA



# Referências

George Coulouris, Jean Dollimore, Tim Kindberg, and Gordon Blair. **Sistemas Distribuídos: Conceitos e Projeto**. Bookman Editora, 5 edition, 2013.

Harvey M Deitel, Paul J Deitel, David R Choffnes, et al. **Sistemas Operacionais**. Pearson/Prentice Hall, 3 edition, 2005.

Maarten Van Steen and A Tanenbaum. **Sistemas Distribuídos: Princípios e Paradigmas**. Pearson/Prentice Hall, 2 edition, 2007.

Harvey M Deitel and Paul J Deitel. **Java, como programar**. Ed. Pearson/Prentice Hall, 8 edition, 2010.

StackOverflow. Disponível em: <https://pt.stackoverflow.com/>. Acesso em: abril de 2023.

GAMMA, Erich et al. **Elements of Reusable Object-Oriented Software**. Design Patterns, 1995.

COOPER, James William. **Java design patterns: a tutorial**. 2000.

GUERRA, Eduardo. **Design Patterns com Java: Projeto orientado a objetos guiado por padrões**. Editora Casa do Código, 2014.

# Referências

AWS Amazon – O que é RESTful. Disponível em: <https://aws.amazon.com/pt/what-is/restful-api/#:~:text=Representational%20State%20Transfer%20(REST)%20is,complex%20network%20like%20the%20internet.>. Acesso em: maio de 2023.

InfoQ. Disponível em: <https://www.infoq.com/minibooks/emag-03-2010-rest>. Acesso em: maio de 2024.

Michelli Brito. Disponível em: <https://www.youtube.com/@MichelliBrito>. Acesso em: abril de 2024.

Obeautifulcode. Disponível em: <http://blog.obeautifulcode.com/API/Learn-REST-In-18-Slides/>. Acesso em: junho de 2022.

Berkeley. Disponível em: <http://courses.ischool.berkeley.edu/i290-rmm/s12/slides/Lecture3%20REST.pdf>. Acesso em: maio de 2023.

Trybe. Disponível em: <https://blog.betrybe.com/tecnologia/deploy/>. Acesso em: julho de 2024.

# Obrigado!

Agradecimentos ao professor:
Prof. MSc. Gustavo Torres Custódio

Contato: profmarcel.wagner@fiap.com.br

Cursos:
Tecnologia em Análise e Desenvolvimento de Sistemas (TDS)
Tecnologia em Defesa Cibernética (TDC)
Engenharia de Software (ES)