EN-3611

Segurança de Redes

Aula 03
Criptografia assimétrica

Prof. João Henrique Kleinschmidt

Problema de distribuição de chaves

- A criptografia de chave simétrica pode manter seguros seus segredos (chave e informação), mas se precisar compartilhar informações secretas com outras pessoas deve-se também compartilhar as chaves que são secretas.

- Como duas ou mais pessoas podem, de maneira segura, enviar as chaves por meio de linhas inseguras?

Utilizando um terceiro confiável

- Alice e Bob podem tentar usar um terceiro confiável, chamada de Michele, que compartilha uma chave com cada funcionário da empresa.
- Alice visita Michele e solicita uma chave. Ela gera uma chave, armazena numa forma segura é dá uma cópia a Alice. Agora as duas compartilham uma chave. Bob também visita Michele, consegue a mesma chave e assim compartilha uma chave com Alice (e Michele).
- Quando Alice quiser se comunicar com Bob, basta criptografar a mensagem com a chave gerada por Michele.
- No esquema do terceiro confiável, duas das partes confiáveis são os correspondentes que se comunicam: Alice e Bob.
- Michele é o terceiro.
- Michele precisa ser confiável, ela tem as chaves de todas as pessoas. Quando Alice e Bob trocam mensagens criptografadas, elas, normalmente, são as únicas pessoas que podem decriptá-las. Mas Michelle também pode, pois ela também detém a chave de sessão.

Terceiro confiável

- O terceiro confiável pode ler todas as mensagens. Se os correspondentes puderem conviver com isso, o esquema funcionará.
- Quando o terceiro confiável deixa a empresa, um novo terceiro deverá existir e o processo começa novamente a partir do zero. Caso, contrário, o terceiro que deixou a empresa pode ganhar acesso a todos os materiais sigilosos.

- Alternativa: Existir um serviço externo. O terceiro não é um indivíduo, mas uma entidade corporativa.
- Novamente, baseado na confiança na entidade para que esta evite e verifique que seus funcionários tenham acesso às chaves.

Algoritmos Criptográficos Assimétricos ou de Chave Pública

- **Histórico:**
  - 1976 (Diffie, Hellman e Merkle)
  - Descoberta de algoritmos criptográficos assimétricos, onde a segurança se baseia nas dificuldades de
    1. Deduzir a mensagem a partir do criptograma
    2. Deduzir uma chave de cifragem a partir da outra chave
  - Desenvolvimento da criptografia moderna:
    - Mecanismos de distribuição de chaves
    - Autenticação de mensagens
    - Provas de identidade

Quem inventou a criptografia de chave pública?

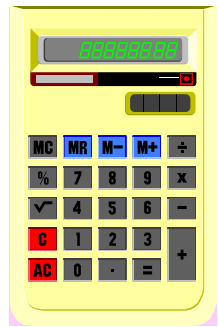
- Década de 60
- James Ellis, do *British Communications Electronic Security Group* (CESG)?
- *National Security Agency* (NSA) dos EUA?
- Usada nas bombas atômicas?
- Ex-URSS? China? Japão?

Criptografia de chave pública

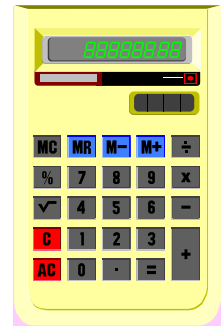
Algoritmo de Criptografia

Algoritmo de Decriptografia

Texto Simples



Texto Cifrado



Texto Simples

Chave Pública

Chave Privada

Criptografia de chave pública

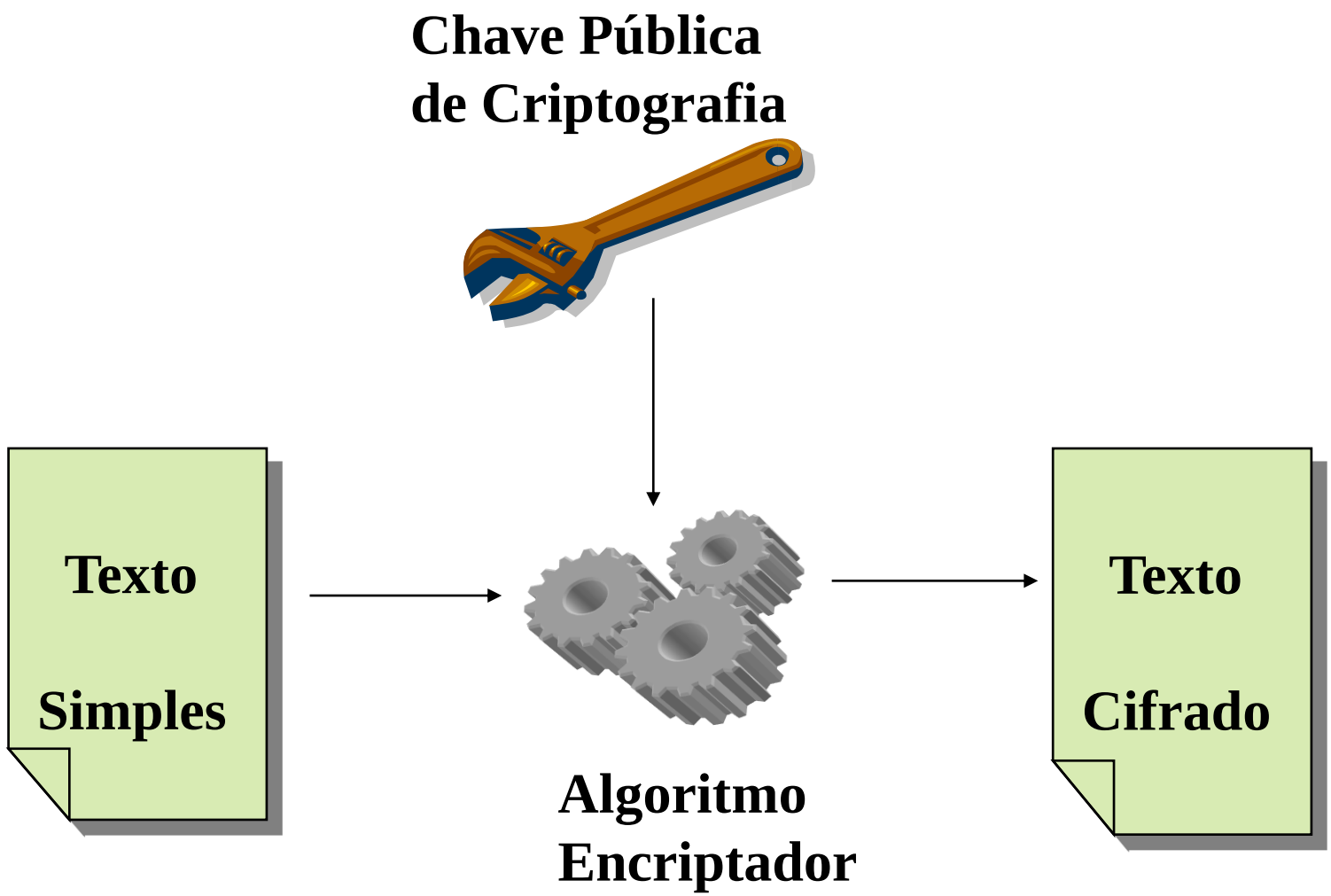
- Na **criptografia assimétrica** a chave utilizada para encriptar não é usada para decriptar.
- As chaves são significativamente diferentes:

  (K_e, K_d)
- Elas são parceiras. Estão relacionadas entre si:

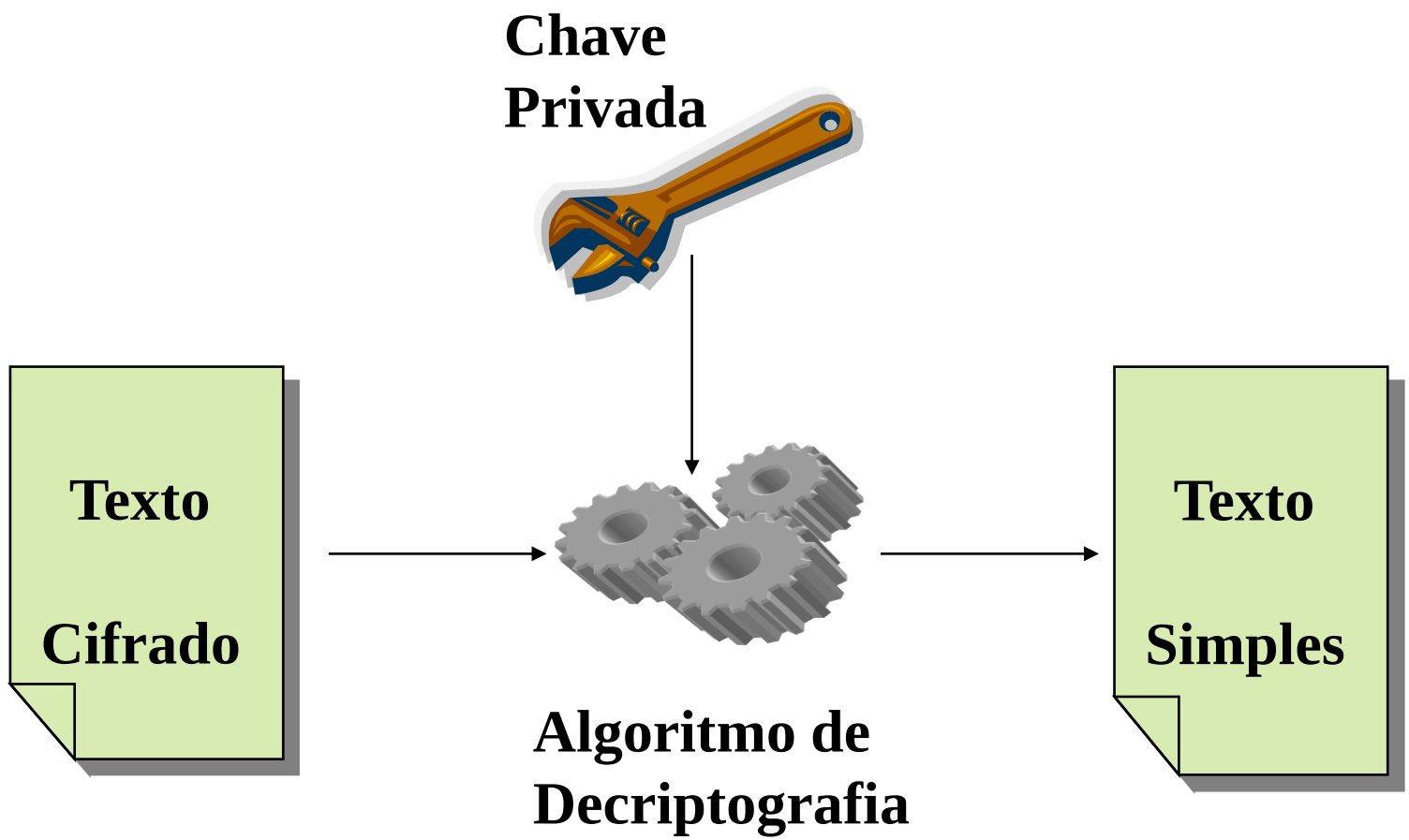
  K_d => K_e **K_e ? =>? K_d**
- O relacionamento é matemático; o que uma chave encripta a outra decripta:

  C = E(K_e, M) **D(K_d, C) = M**

Criptografando com chave pública



Descriptografando com chave privada



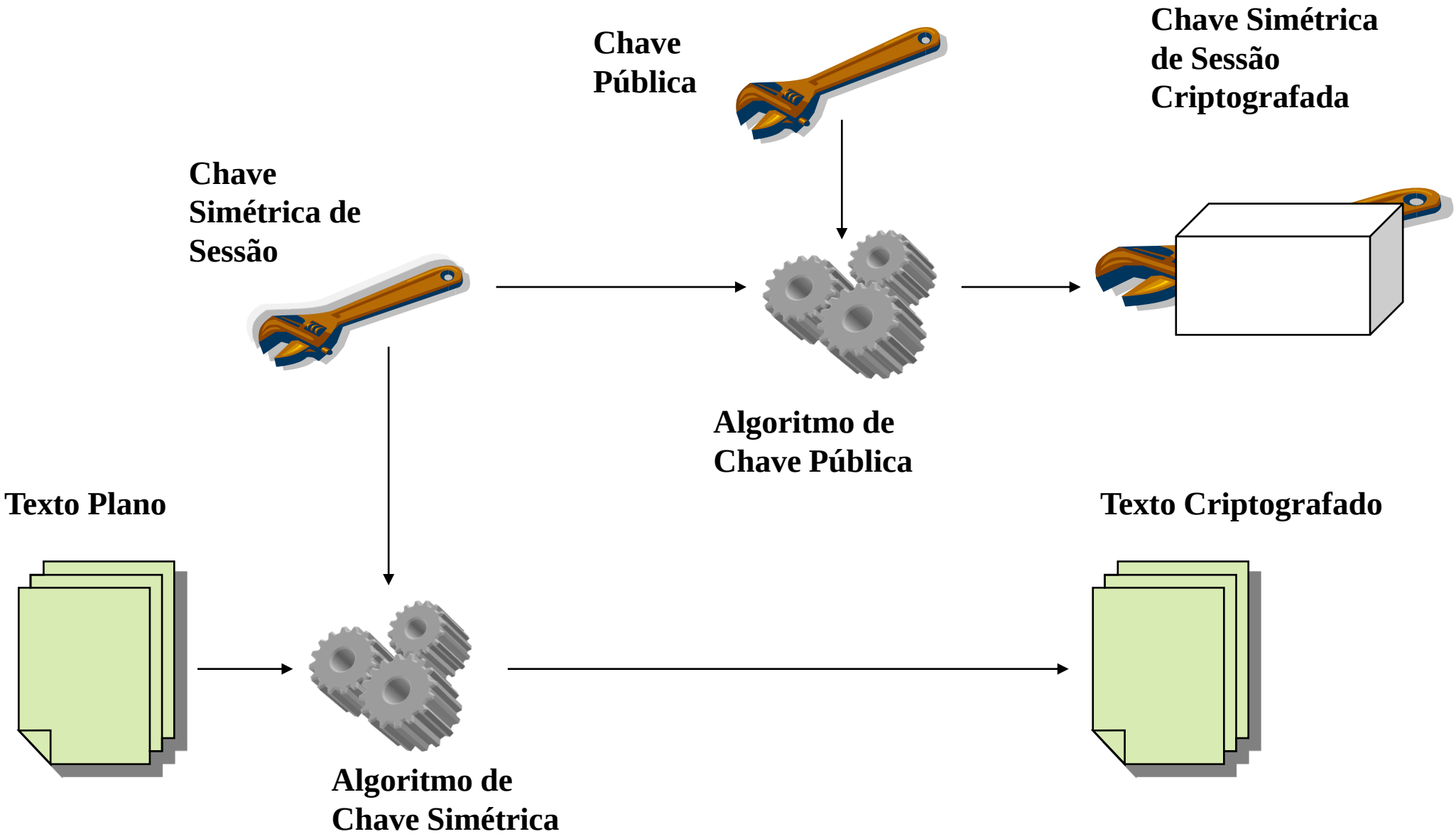
Desempenho

- Para informação em grande quantidade, algoritmos de chave pública são lentos. (20Kb a 200Kb) por segundo. Muito lento para processamento de dados em volume.

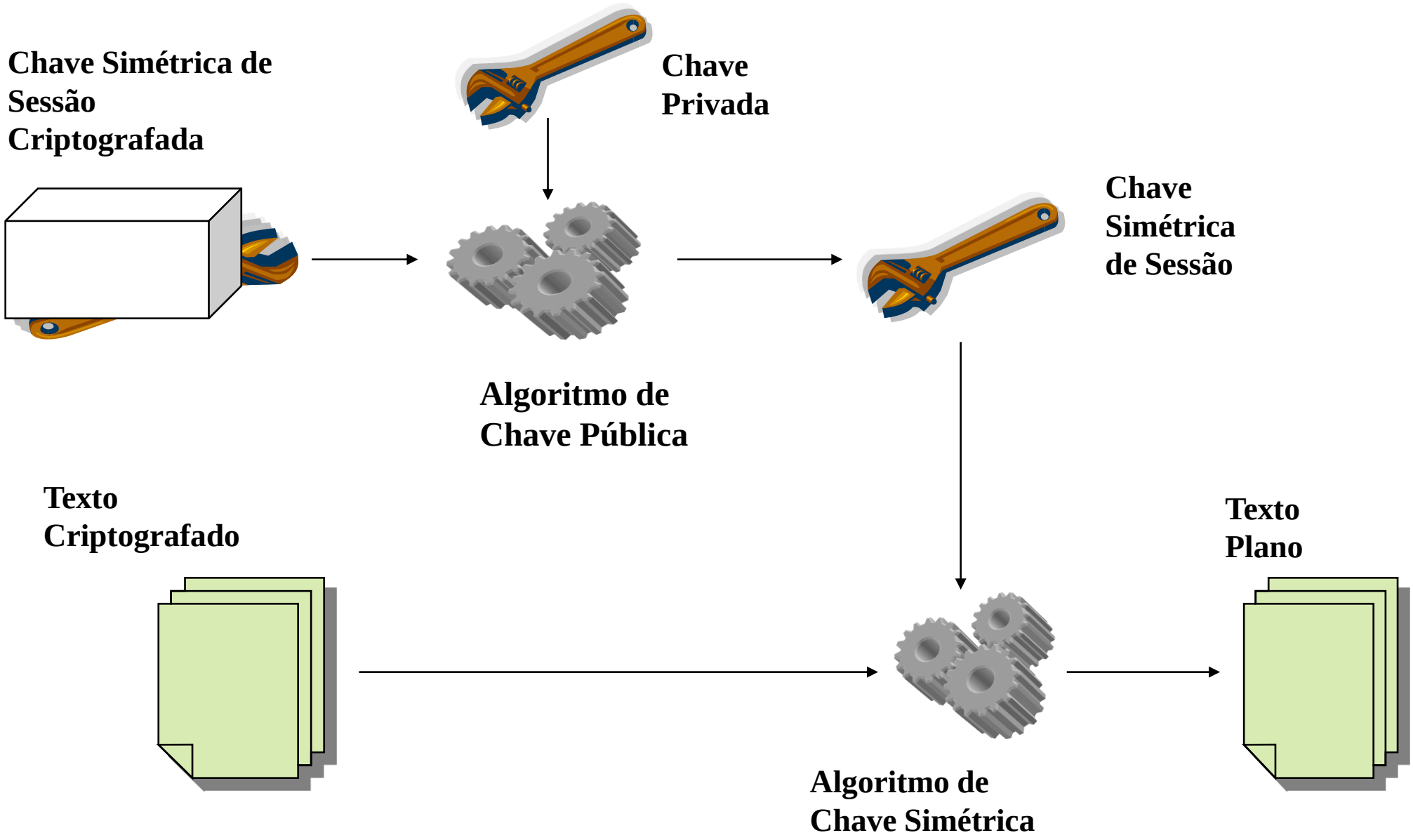
- Algoritmos de chave simétrica podem encriptar informação em grande quantidade bem mais rapidamente. (10Mb, 20Mb, 50 Mb ou mais) por segundo.

- Solução: usar a combinação de criptografia de chave simétrica e de chave pública.

Criptografando em Envelope Digital



Descriptografando o Envelope Digital



Vantagens do envelope digital

- Segredo não precisa ser compartilhado antecipadamente.
- Segredo compartilhado através da chave simétrica de sessão.
- Manter uma chave separada para cada pessoa (chave pública) que não precisa estar protegida.
- Não é preciso armazenar as próprias chaves públicas. Diretórios de chaves públicas podem estar disponíveis.

Gerenciamento de chaves públicas

- Se Alice e Bob não conhecem um ao outro, como eles irão obter as respectivas chaves públicas para iniciar a comunicação entre eles?
- Como Alice (Bob) pode ter certeza que está realmente obtendo a chave pública de Bob (Alice) ?
- Bob coloca sua chave pública na sua página Web. Não funciona!!!
- Se Eva intercepta a solicitação e responde a Alice com uma página falsa, fazendo a substituição da chave pública de Bob pela chave pública dela.
- Quando Alice envia sua primeira mensagem criptografada, será com a chave pública de Eva.
- E necessário um mecanismo apropriado para que se possa disponibilizar chaves: uma autoridade certificadora (AC). Esta AC é responsável em certificar e distribuir chaves públicas.

Algoritmos de Chave Pública

- Dos algoritmos assimétricos até hoje propostos, poucos são **considerados** seguros e práticos:
  - RSA
  - El Gamal
  - Rabin
  - ECC – Criptografia de Curva Elíptica
- Outros algoritmos:
  - Úteis apenas para assinatura
  - Ou inseguros, muito lentos ou requerem chaves muito longas

Acordo de chaves Diffie-Hellman

- Resolve problema de **distribuição de chaves**, criando uma **chave compartilhada**.
- Alice e Bob têm que concordar sobre dois grandes números:
  - **p** (um número primo)
  - **g** (um número pseudo-aleatório)
- **p** é um número primo gerado a partir de um gerador de números pseudo-aleatórios (PRNG), sendo verificado se é primo pelo Teste de Fermat.
- **g** é um número gerado por um PRNG, que se relaciona com o valor de **p** (**g < p** e **g** é uma raiz primitiva de **p**)

  $(g^{\emptyset(p)} = 1 \bmod p)$.
- Estes números podem ser **públicos**, assim, **qualquer uma** das partes pode escolher **p** e **g** e dizer ao outro abertamente.

Acordo de chaves Diffie-Hellman

- Alice gera por um PRNG, um número grande chamado **x.** Ela guarda **x** como **secreto**.
- Alice tem agora **(p, x)** que define a **chave privada.**
- Alice calcula $\mathbf{g^x \bmod p}$.
- Alice inicia o protocolo do acordo de chave enviando a Bob uma mensagem contendo $\mathbf{(p, g, g^x \bmod p)}$.
- Bob tem agora um número grande $\mathbf{g^x \bmod p}$ definindo a tripla $\mathbf{(p, g, g^x \bmod p)}$ a qual é transmitida para Bob, como a **chave pública** de Alice.
- Bob escolhe um número **y** secreto.
- Bob responde enviando a Alice uma mensagem contendo $\mathbf{(g^y \bmod p)}$

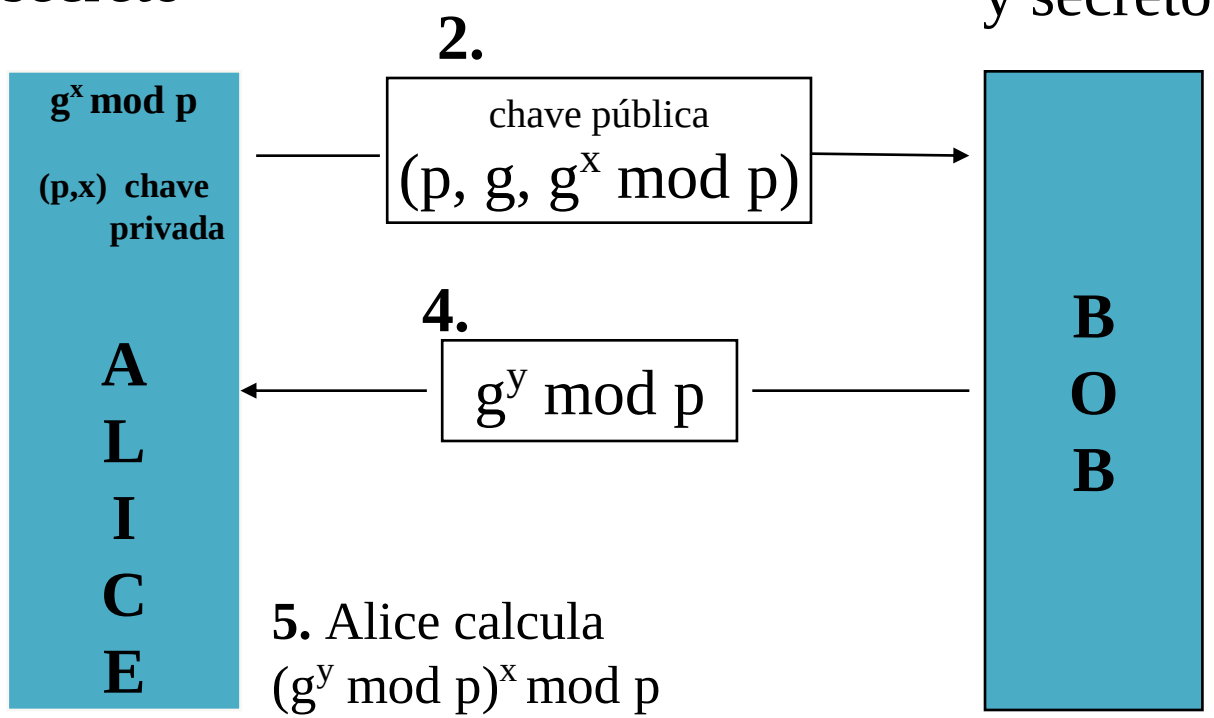
Acordo de chaves Diffie-Hellman

- Alice calcula **$(g^y \bmod p)^x \bmod p$**
- Bob calcula **$(g^x \bmod p)^y \bmod p$**
- Pela lei da aritmética modular, ambos os cálculos resultam em **$g^{xy} \bmod p$** .
- Alice e Bob, agora **compartilham uma chave secreta**: **$g^{xy} \bmod p$**

Acordo de chaves Diffie-Hellman

1. Alice escolhe p, g públicos e x secreto

3. Bob escolhe y secreto



5. Alice calcula $(g^y \bmod p)^x \bmod p = g^{xy} \bmod p$ (chave secreta compartilhada)

6. Bob calcula $(g^x \bmod p)^y \bmod p = g^{xy} \bmod p$ (chave secreta compartilhada)

Acordo de chaves Diffie-Hellman

- O algoritmo não criptografa os dados.
- Duas partes geram a mesma chave (**g^{xy} mod p)**.
- Este procedimento é chamado **Acordo de Chave**.
- **Dificuldade de quebra do algoritmo**:

Um intruso conhece **g** e **p**. Se pudesse descobrir **x** e **y**, descobriria a chave secreta.

O problema é que dado (**g^{x} mod p**) e (**g^{y} mod p**), **não pode descobrir x** nem **y**.

Nenhum algoritmo é conhecido para computar o **módulo de logaritmo discreto** de um **número primo p muito grande** num tempo aceitável**.**

Algoritmo RSA

- O nome é formado com iniciais dos autores: Rivest, Shamir & Adleman (1978)
- Mais usado e fácil de implementar entre os algoritmos assimétricos
- O algoritmo vem resistindo a quase 20 anos de análise
- Baseado em alguns princípios da **Teoria dos Números**.
- Sua segurança é baseada na dificuldade de se fatorar números inteiros

Algoritmo RSA

- **Sumário do algoritmo**:

1. Escolher dois números primos grandes, **p** e **q** (tipicamente entre 10^{75} e 10^{100})
   Um PRNG escolhe **p**; Teste de Fermat localiza **q**.
2. Calcule **n = p.q**
3. Calcule Ø(**n**) = **z = (p-1).(q-1)** (função de Euler)
4. Escolher um ***número relativamente primo*** **a z** e chamá-lo de **e** (isto é, tal que **e não tenha fatores primos comuns com z**)
5. Encontre **d** tal que **e.d mod z = 1** e **d < z**
6. Dividir o texto plano (uma cadeia de bits) em blocos, de modo que cada mensagem do texto plano **M**i (bloco) caia no intervalo **0 <= M < n**.

   Isto pode ser feito agrupando-se o texto plano dentro de blocos iguais de **k** bits, onde **k** é o maior inteiro para o qual $\mathbf{2^k < n}$. Em aplicações práticas **k** varia de 512 a 1024 bits.

Algoritmo RSA

- 6. Para **encriptar** um mensagem **M**,
  calcule **$E(e,n,M) = C = M^e \bmod n$.**
  Para **decriptar**, calcule
  $D(d,n,C) = M = C^d \bmod n$

- Para **encriptar** precisamos de **e** e **n**. Para **decriptar** precisamos de **d** e **n**. Assim, a **chave pública** consiste do par (**e,n**) e a **chave privada** do par (**d,n**).
- Pode ser provado que para todo **M,** essas funções são inversas:
  $E(D(x)) = D(E(x)) = x$

Segurança do RSA

- A segurança do método é baseada na dificuldade de se **fatorar números grandes**.
- Se um cripto-analista puder **fatorar n**, ele poderia então descobrir **p** e **q**, e destes, **z**.
- RSA Corporation tem emitido uma série de desafios para fatorar números de mais de 100 dígitos decimais.
- Números de até 174 dígitos decimais (576 bits) têm sido fatorados, e assim o uso do algoritmo RSA com chaves de 512 bits é inaceitável para muitos propósitos.

Segurança do RSA

- RSA Corporation (que retém a patente do algoritmo RSA) recomenda um comprimento de chave de ao menos 768 bits (em torno de 230 dígitos decimais), por um período de segurança a longo prazo de aproximadamente 20 anos.
- Chaves de 1024 bits são utilizadas.
- Chaves tão grandes quanto 2048 bits são usadas em algumas aplicações.
- Os algoritmos de fatoração usados são os melhores disponíveis.
- Algoritmos criptográficos assimétricos que usam multiplicação de números primos como *função de uma via* estarão vulneráveis quando um algoritmo de fatoração mais rápido for descoberto.

Ataque em Módulo Comum

- Possível quando:
  - A distribuição de chaves para a cifra que usa o RSA atribui chaves com o mesmo módulo a usuários distintos
  - Qualquer mensagem encriptada por mais de um usuário pode ser facilmente vazada

Ataque com Expoentes Pequenos de Encriptação

- A encriptação/verificação de assinatura no RSA:
  - Mais rápida de acordo com o tamanho da chave pública
  - Menor chave: + rápido
  - Este tipo de ataque é possível com a encriptação de:
    - ***e(e+1)/2*** mensagens linearmente dependentes
    - Portanto a dependência linear deve ser evitada

Prevenção contra Ataques Conhecidos ao RSA

1. O conhecimento de um par **(e,d)** permite a fatoração do módulo **n**
2. O conhecimento de um par **(e,d)** permite encontrar outros pares para o mesmo **n**
3. O módulo comum não deve ser usado em serviços de rede
4. As mensagens devem ser preenchidas com bits aleatórios até atingir o tamanho de **n**
5. O expoente da chave pública deve ser “grande”, e a assinatura anteceder a cifra

RSA e envelope digital

- Pode-se usar o RSA para criptografar dados diferentes do que uma chave de sessão, como no processo do envelope digital, mas o RSA não é tão rápido quanto os algoritmos simétricos.

RSA: Padronização e Patentes

- O RSA é um padrão de fato para criptografia assimétrica
  - Norma ISO 9796
  - Norma ANSI
  - Padrão bancário na França e Austrália
  - Não era padrão nos EUA por problemas de disputa sobre direitos de patente
    - A patente, válida somente nos EUA, expirou em 20/9/2000

Outros Algoritmos Assimétricos

- El Gamal
  - Dificuldade de se extrair logaritmos discretos em corpos finitos
- Rabin
  - Dificuldade de se extrair raiz quadrada em anéis.
- Curvas elípticas
- Algoritmos baseados:
  - No problema da mochila
  - Em códigos corretores de erros
  - Em autômatos finitos

Curvas elípticas

- Algoritmo ECDH (Elliptic Curve Diffie-Hellman)

- Encontra um curva elíptica *E* e um ponto *P* nesta curva.
- Gera um número pseudo-aleatório chamado de *d.*
- Multiplica *d* por *P*.
- O resultado é *Q*, que também é um ponto na curva.
- *E*, *P* e *Q* são publicados. O desafio é encontrar *d*.
- Esse problema é conhecido como o problema de log discreto de curva elíptica (mais difícil de resolver que o log discreto ou fatoração).
- Se a curva for grande, não se conhece uma solução que seja encontrada num curto espaço de tempo.

- Algoritmos Baseados no Problema da Mochila: *Merkle-Hellman:*
- Problema combinatório de se encontrar uma partição em um conjunto fixo de inteiros onde a soma de elementos seja igual ao argumento (Problema da Mochila)
- São inseguros:
  - A chave privada pode ser obtida em tempo polinomial a partir da pública

- Algoritmos baseados em códigos corretores de erros: *McEliece* - Usam códigos de Goppa
- Rápido e, até hoje, seguro
- Pouco usado porque expande a mensagem **m** e utiliza chaves muito grandes

Implementações

- Padronização:
  - Por normatização (Ex.: ISO) ou por forças de mercado
  - Interoperabilidade
    - A segurança na computação tende naturalmente à busca de padrões
    - Esta tendência se torna mais imperativa com o advento das redes globais e ambientes de computação distribuída

Implementações

- Limitações:
  - Sistemas legados cuja concepção original não contemplava segurança e/ou interoperabilidade
  - Legislação local e/ou internacional
    - Omissa
    - Desatualizada
    - Mal pensada
  - Interesses privados

Implementações

- Desafios
  - Novas aplicações ou tecnologias
    - Exigem estudos específicos
    - Exigem proteção aos dados
  - A necessidade de criptografia:
    - Aumenta a complexidade do problema computacional originalmente proposto
  - Em um sistema, é necessário pensar sempre em:
    - Segurança
    - Desempenho
    - Usabilidade