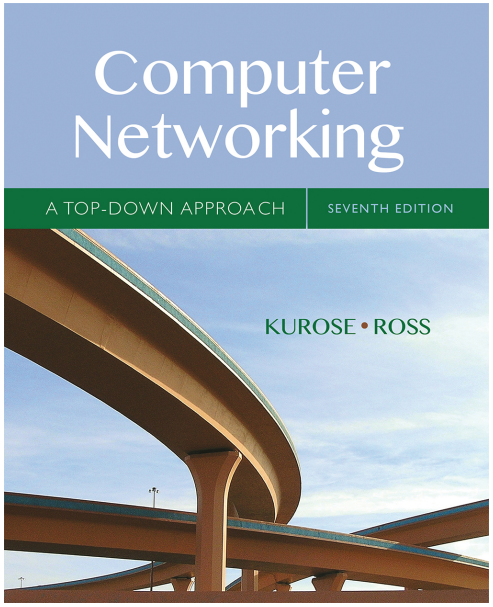
Segurança em Redes

Comunicações por Computador

Mestrado Integrado em Engenharia Informática

3º ano/2º semestre

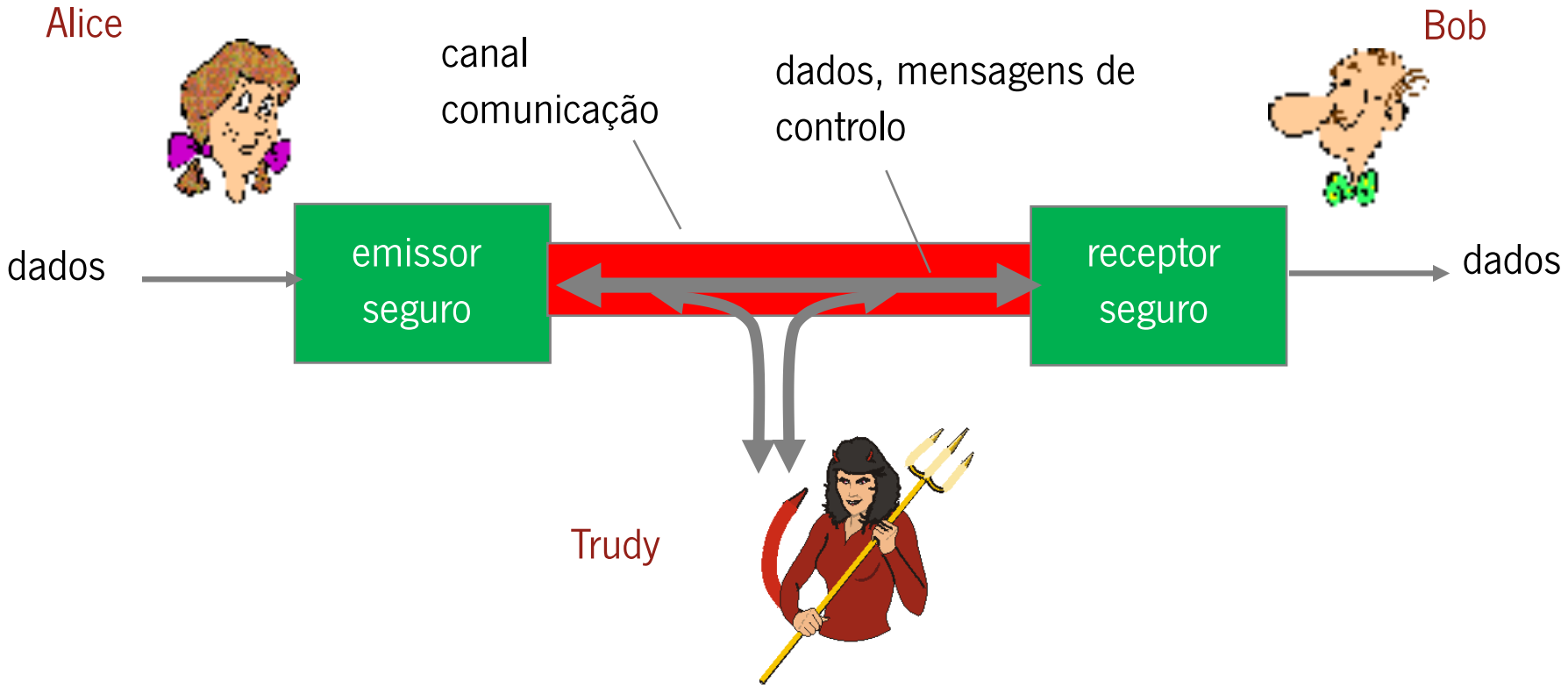
2019/2020



Capítulo 8: Security in Computer Networks

Atores: os amigos e os inimigos

- Personagens bem conhecidas do mundo da segurança ☺
- Alice e o Bob estão apaixonados e querem comunicar de forma segura;
- Que pode *Trudy* (a intrusa) fazer?



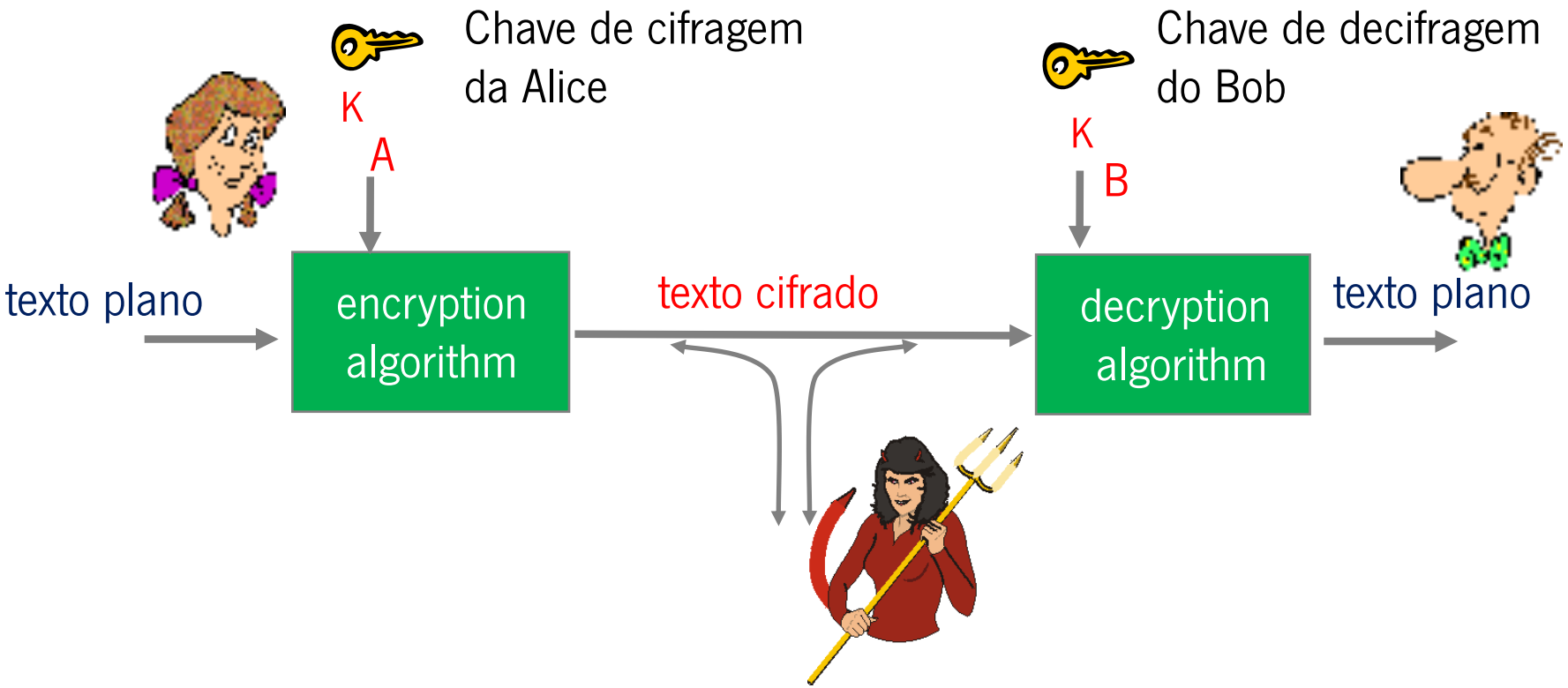
Que podem fazer os "maus"?

- espionagem: interceção indevida de mensagens
- inserção de mensagens numa conexão (comunicação)
- disfarce: pode fingir (*spoof*) endereços de origem nos pacotes (ou qualquer outro campo dos pacotes)
- desviar sessões (*hijacking*): "tomar conta" de conexões que estão a decorrer, remover o emissor ou o recetor, colocando-se no lugar destes
- negação de serviço: impedir premeditadamente que um serviço seja usado por outros (ex: sobrecarregando-o de algum modo)

Propriedades de uma comunicação segura

- Confidencialidade: só o emissor e o recetor indicado devem "perceber" o conteúdo das mensagens
- Autenticação: emissor e recetor pretendem confirmar a identidade um do outro
- Integridade da mensagem: emissor e recetor querem garantir que a mensagem não foi alterada (no percurso pela rede, antes do envio ou depois da receção) sem que tal possa ser imediatamente detetado
- Não Repúdio: evidências que impeçam intervenientes de negar comunicação
- Acesso e Disponibilidade: serviços devem estar acessíveis e com disponibilidade para os seus utilizadores

A “linguagem” da criptografia



criptografia de chave simétrica: emissor e recetor usam a *mesma* chave

criptografia de chave pública: uma chave para cifrar (publica) outra para decifrar (privada)

Criptografia de chave simétrica

Cifra de substituição: substituir uma coisa por outra

- cifra monoalfabética: substitui uma letra por outra

```
Text plano:   abcdefghijklmnopqrstuvwxyz
                        ↓
Texto cifrado:  mnbvcxzasdfghjklpoiuytrewq
```

Ex.:

```
Texto Plano: Alice, Amo-te. Bob.

Texto Cifrado: Mgsbc, Mhk-nc. Nkn.
```

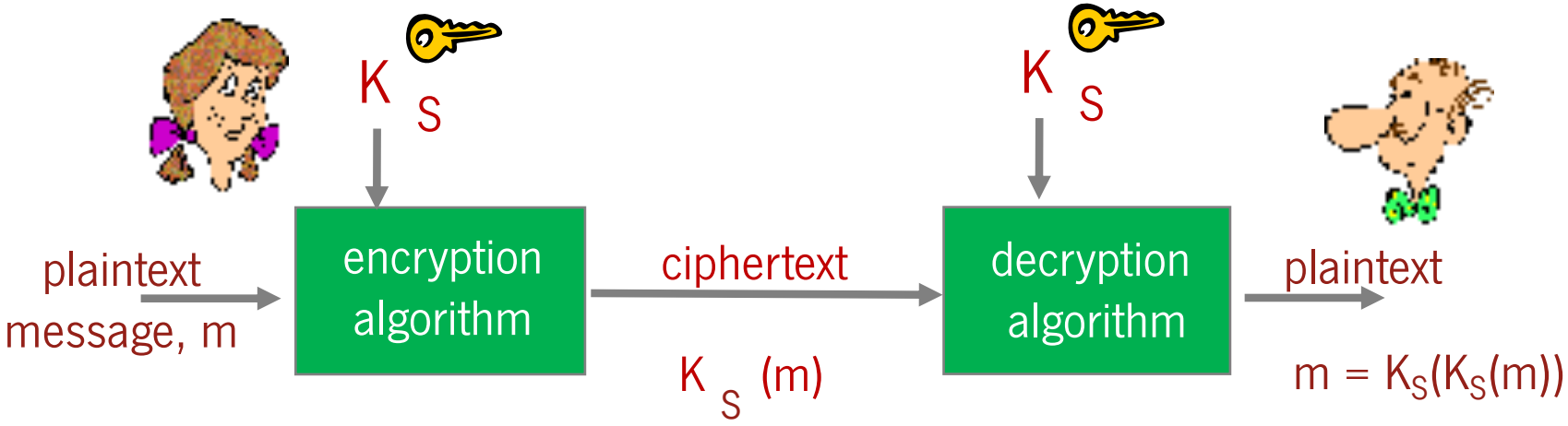
Q: Será fácil ou difícil quebrar esta cifra?

- Pela força bruta (difícil?)
- Outro método?

Criptografia de chave simétrica

- **ataque baseado no texto cifrado:** Trudy tem textos cifrados que pode analiaar
- **duas abordagens:**
  - força bruta: procura total no espaço de chaves
  - análise estatística
- **ataque baseado num texto conhecido:** Trudy conhece o texto original e o texto cifrado
  - ex: ataque à enigma da 2ª Guerra Mundial
- **ataque baseado num texto previamente escolhido:** Trudy consegue obter uma versão cifrada de um texto escolhido por ela

Criptografia de chave simétrica



Chave simétrica: Alice e Bob conhecem a mesma chave (simétrica) K_S

- Ex: conhecem o padrão de substituição do alfabeto! (ou a máquina de escrever, como a famosa ***Enigma*** da 2ª guerra mundial)

- **Pergunta: Como podem eles combinar a chave?**

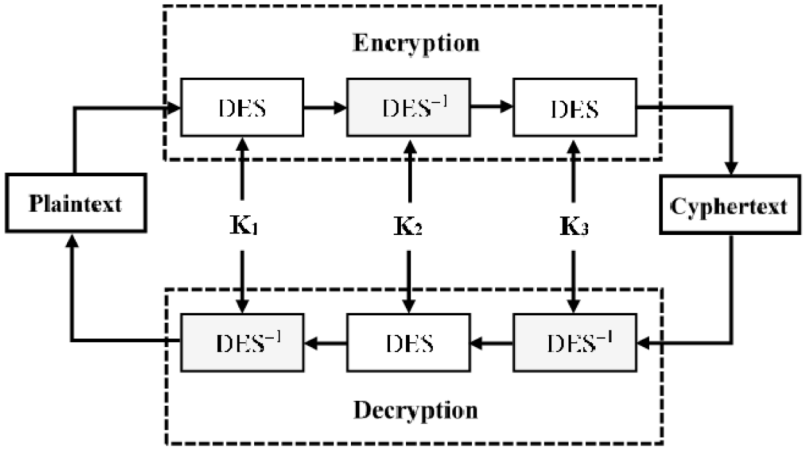
Algoritmos mais usados

- **DES – Data Encryption Standard** (fraco)
  - Chaves de 56 bits que processam blocos de 64 bits de cada vez
  - Quebra-se por força bruta em **menos de 1 dia**!
- **3-DES (3 x DES)**
  - Usa 3 chaves DES sequencialmente

    Cifra com K1, decifra com K2, cifra com K3
  - Chaves:

    56 bit (todas iguais, compatível DES),

    112 bit (k1=K3) ou 168 bit (3 distintas)
  - Prevê-se que possa ser usado até ao ano 2030..

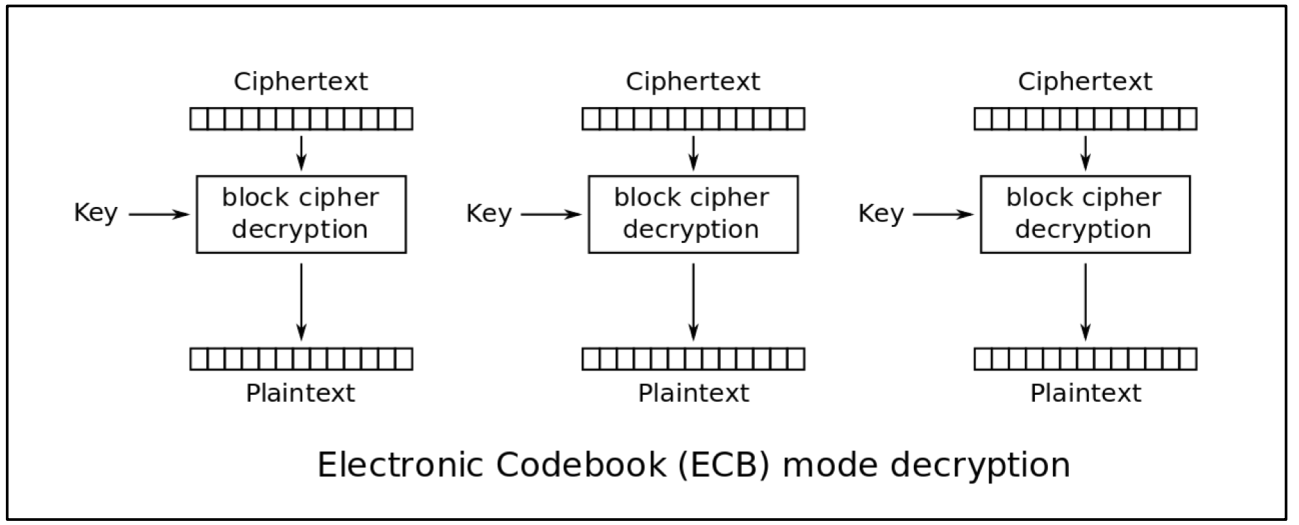


- **AES – Advanced Encryption Standard**
  - Veio em 2001 para substituir o velho DES
  - Processa blocos de 128 bits de cada vez
  - Chaves de 128, 192 ou 256 bits de tamanho
  - Pela força bruta, o que no DES demora um segundo a quebrar ...

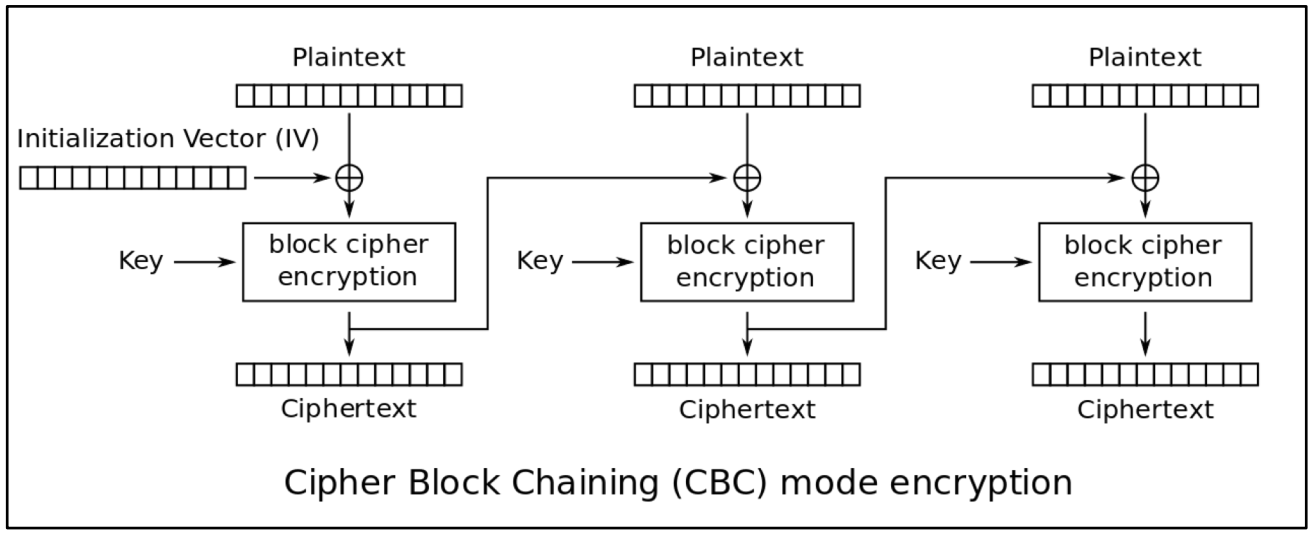
    ... demorará **149 triliões de anos** no AES!!!

Criptografia de chave simétrica

- **ECB:**
  - O mesmo padrão na entrada, produz a mesma saída!
  - $m_{(i)}$ = "HTTP/1.1" → C(i) = ""k329aM02" qualquer que seja (i)



- **CBC:**
  - Antes de cifrar, mistura (XOR) o bloco de texto de entrada com o bloco anterior cifrado
  - $C_{(0)} = IV$ (Vetor inicial enviado às claras)
  - $C_{(i)} = K_S (m_{(i)} \underline{XOR} C_{(i-1)})$



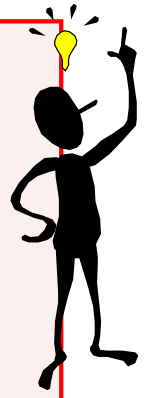
Imagens: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

Criptografia de Chave Pública

Criptografia de chave simétrica

- exige que *emissor* e *recetor* conheçam a mesma chave secreta
- Pergunta: como podem combinar uma, se, por exemplo, não se conhecem ou nunca estiveram juntos?

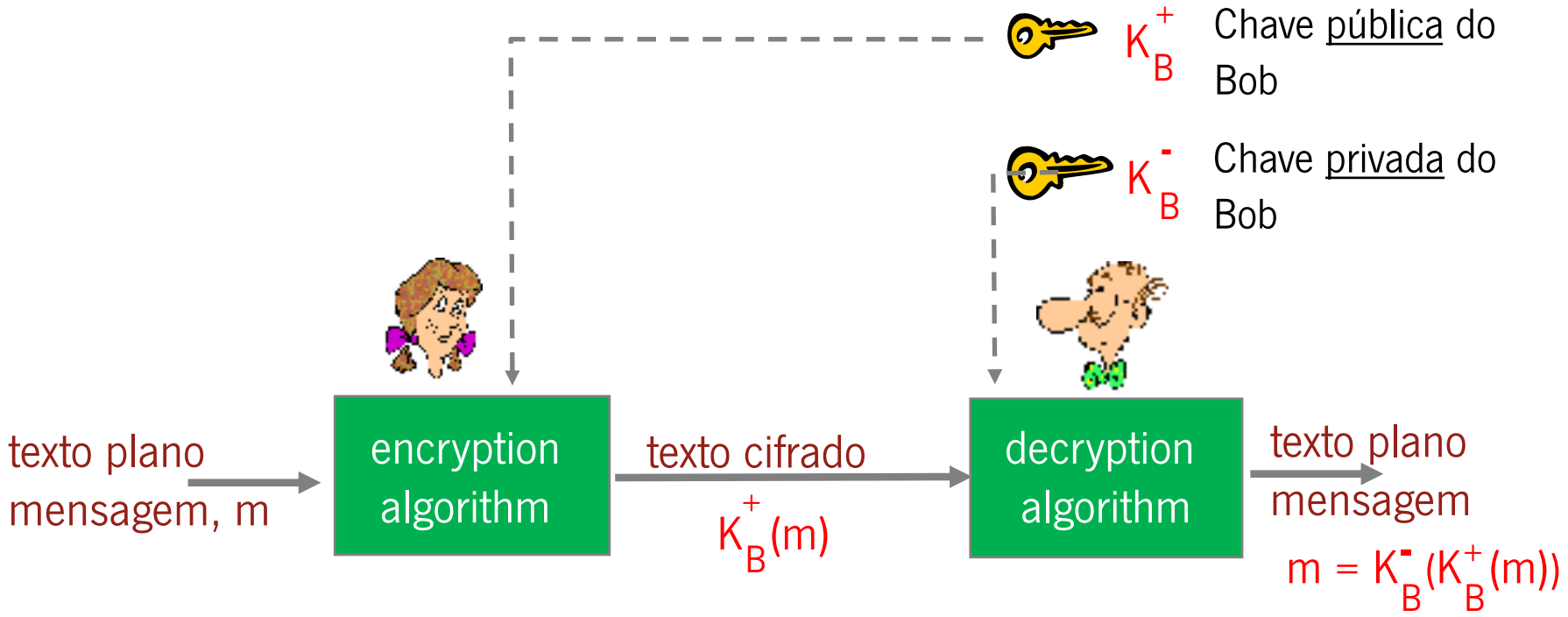
Criptografia de Chave Pública



- abordagem radicalmente diferente
  [Diffie-Hellman76, RSA78]
- emissor e receptor não partilham nenhum segredo!
- Usa um par de chaves
- Chave pública conhecida por todos
- Chave Privada apenas conhecida pelo recetor

Criptografia de Chave Pública

Confidencialidade (e também integridade)



Só o Bob, na posse da sua chave privada, poderá decifrar a mensagem
Mais ninguém pode fazê-lo – total confidencialidade!
Se não decifrar é porque não mantém a integridade

Criptografia de Chave Pública

Requisitos:

1. necessário um par de chaves tais que

$$K_B^-(K_B^+(m)) = m$$

2. **deverá ser impossível obter a chave privada a partir da chave pública!**

RSA: Algoritmo Rivest, Shamir, Adleman

Criptografia de Chave Pública

A seguinte propriedade é muito útil:

$$\underbrace{K_B^-(K_B^+(m))}_{\text{Usar a } \textit{chave pública} \text{ e depois a } \textit{privada}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{Usar a } \textit{chave privada} \text{ e depois a } \textit{pública}}$$

O resultado é o mesmo!

Propriedade matemática (explicação no livro):

- $(m^d \bmod n)^e \bmod n = m^{de} \bmod n = m^{ed} \bmod n = (m^e \bmod n)^d \bmod n$
- $n = pq$ (p e q numeros primos)

Questões

- **Usa criptografia de chave pública, que propriedades tem a seguinte comunicação:**
  - Mensagem cifrada com a chave pública do originador
  - Mensagem cifrada com a chave pública do destinatário
  - Mensagem cifrada com a chave privada do originador
  - Mensagem cifrada com a chave privada do destinatário

Assinatura Digital

Alice recebe uma mensagem do Bob, e quer garantir que:

- a mensagem veio originalmente do Bob
- a mensagem não foi alterada (mantém-se íntegra) desde que foi enviada pelo Bob até ter sido lida pela Alice
- Bob não pode negar (não repúdio) que enviou a mensagem: Alice (recetor) consegue provar a qualquer um que foi o Bob que enviou e que não poderia ter sido mais ninguém, nem mesmo a própria Alice!

"Assinatura Digital"

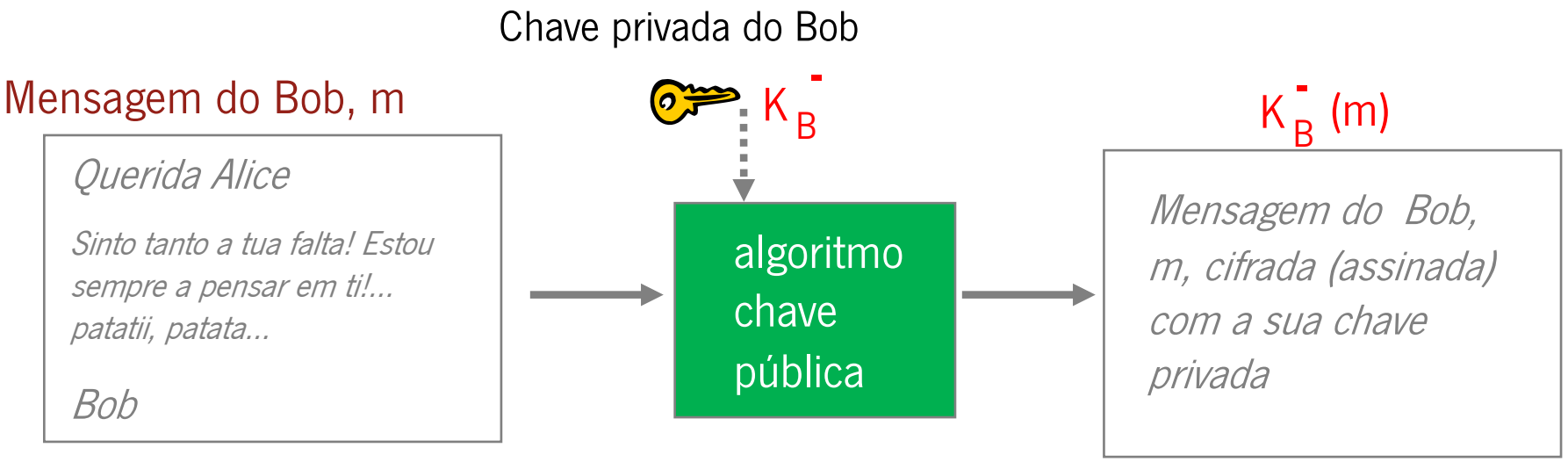
garante integridade, autenticação do originador, não repúdio do originador

- Técnica criptográfica que se assemelha à assinatura manual...

Assinatura Digital (1)

Método 1: usando só criptografia de chave pública

- Podemos usar a *chave privada* para assinar digitalmente a mensagem m
- Bob "assina" m cifrando-a com a sua chave privada, criando assim uma mensagem assinada $K_B^-(m)$

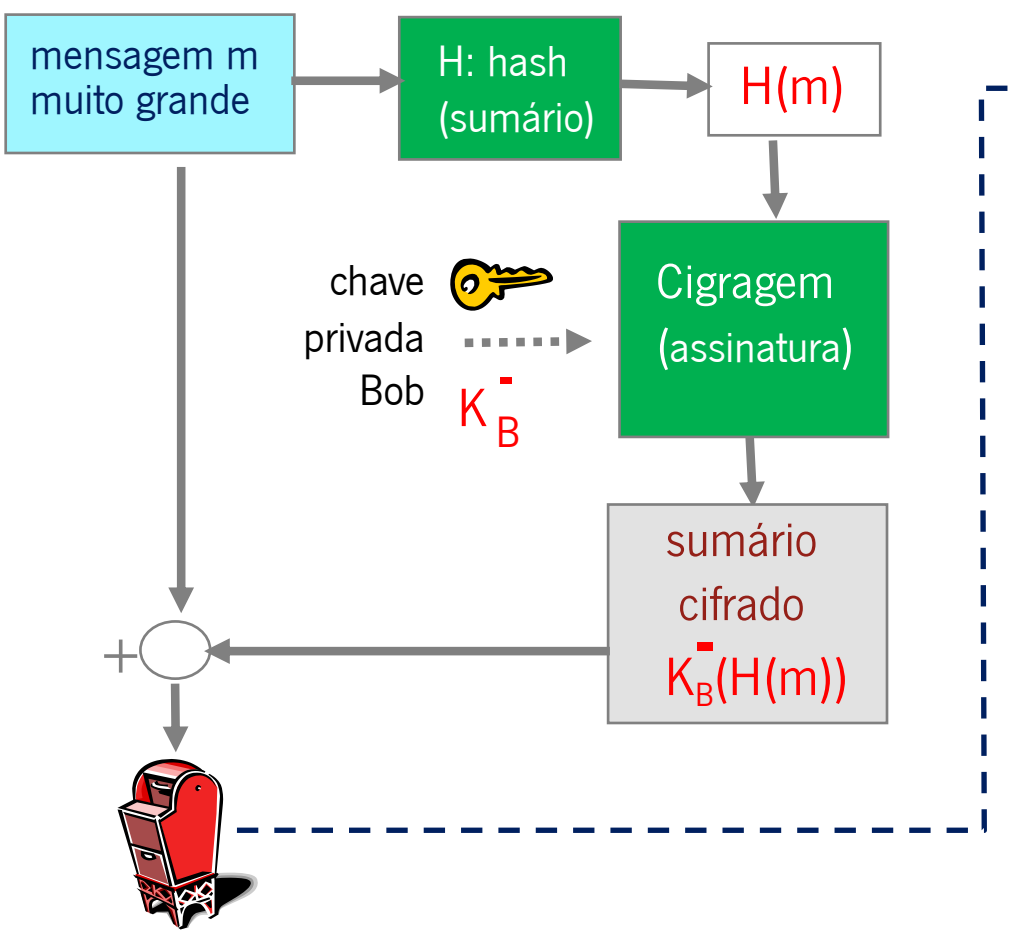


- Garante Autenticação do originador, não repúdio do originador e integridade
- Mas... não usado utilizado na prática por questões de desempenho! Muito lento!

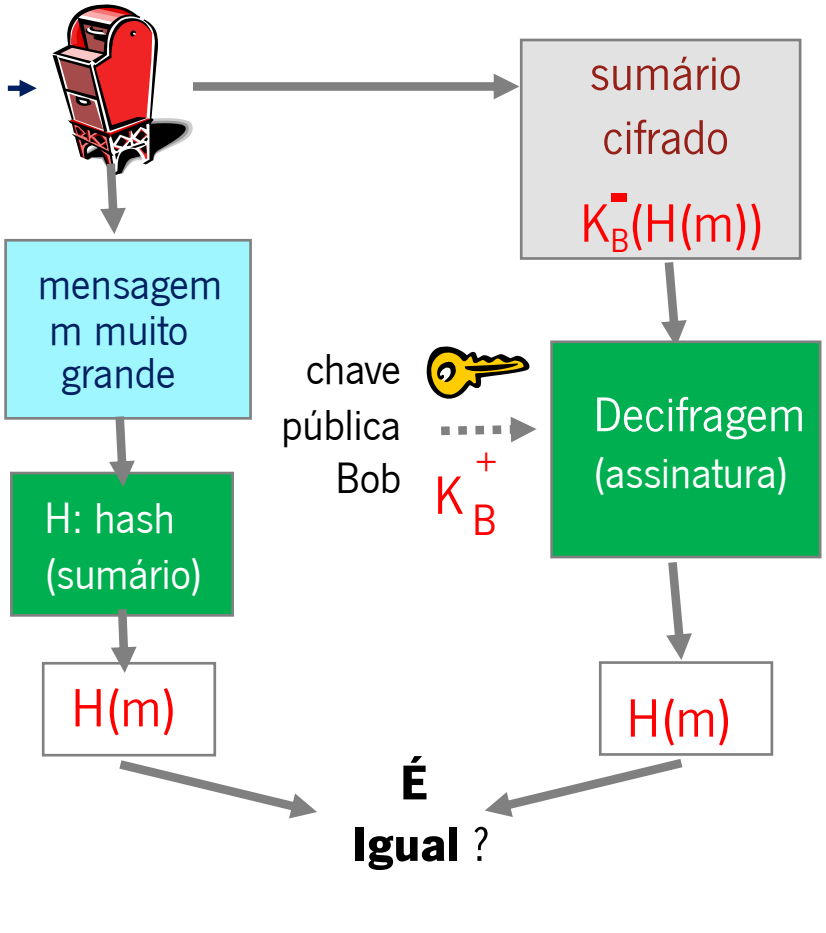
Assinatura Digital (2)

Método 2: criptografia de chave pública e uso de função Hash

- Podemos usar a *chave privada* para assinar digitalmente um Hash(m) em vez de m



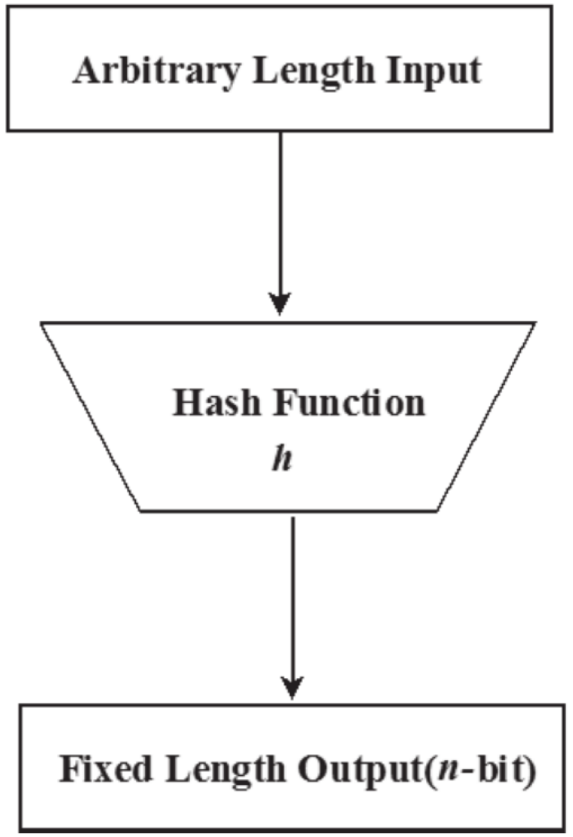
Bob envia mensagem m + assinatura digital



Alice verifica a assinatura

Função hash criptográfica

- **dada uma mensagem de entrada m, produz um sumário de tamanho fixo, H(m)**
  - Nota: neste aspeto é parecido com *checksum*
- **é computacionalmente improvável encontrar duas mensagens x, y diferentes (x != y) e que H(x) = H(y)**
  - de igual modo: dado um m = H(x), (com x desconhecido), não se consegue determinar o x a partir do m
  - Nota: mas isto não é verdade para o *cheksum!*



Algoritmos mais usados

- **MD5 – Message Digest**
  - Calcula sumários de 128 bits em 4 passos
  - ataques ao MD5 (collision attacks) em 2005 mostram que já não é adequado
- **SHA-1 – Secure Hash Algorithm**
  - Calcula sumários de 160 bits
  - Podem detectar-se colisões em 2^{51} tests
- **SHA-2** (sumários de 256 e 512 bit)
- **SHA-3** (publicado em 2015)

Questões

- Comente a seguinte afirmação: ”A assinatura digital associa o assinante ao documento assinado, garantindo integridade e não repúdio.”

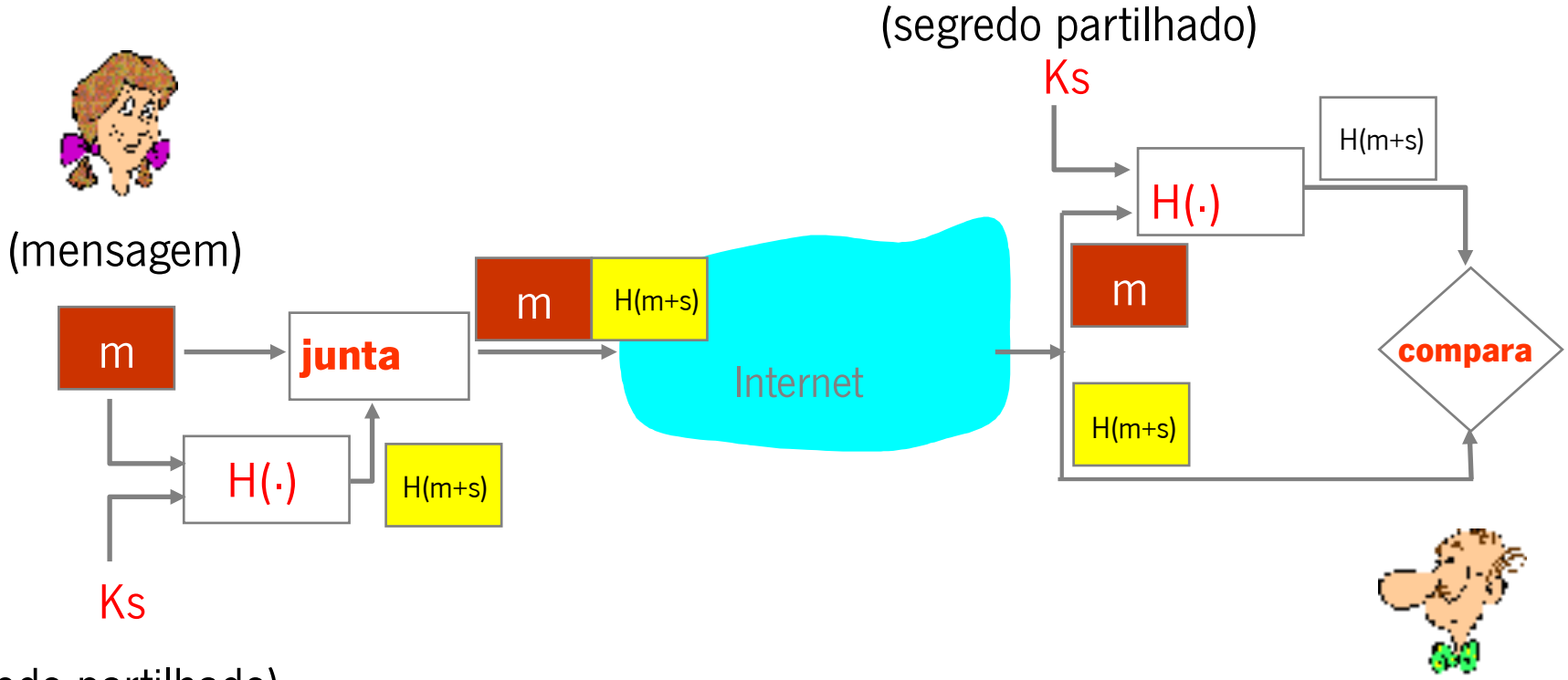
Assinatura Digital

- **Garantias**
  - Só o Bob pode ter assinado m, pois só ele conhece a sua chave privada
  - Mais ninguém poderia ter assinado m
  - A mensagem que foi assinada foi m e não um m' qualquer
  - Qualquer um pode verificar isso: basta pegar na chave pública de Bob e decifrar a assinatura
  - Garante ainda o não repúdio – mesmo em tribunal! – pois Bob não poderá negar ter usado a sua chave privada

Integridade e Autenticação da Origem

MAC – Message Authentication Code

Envia o sumário da mensagem e do segredo juntos (m+s)

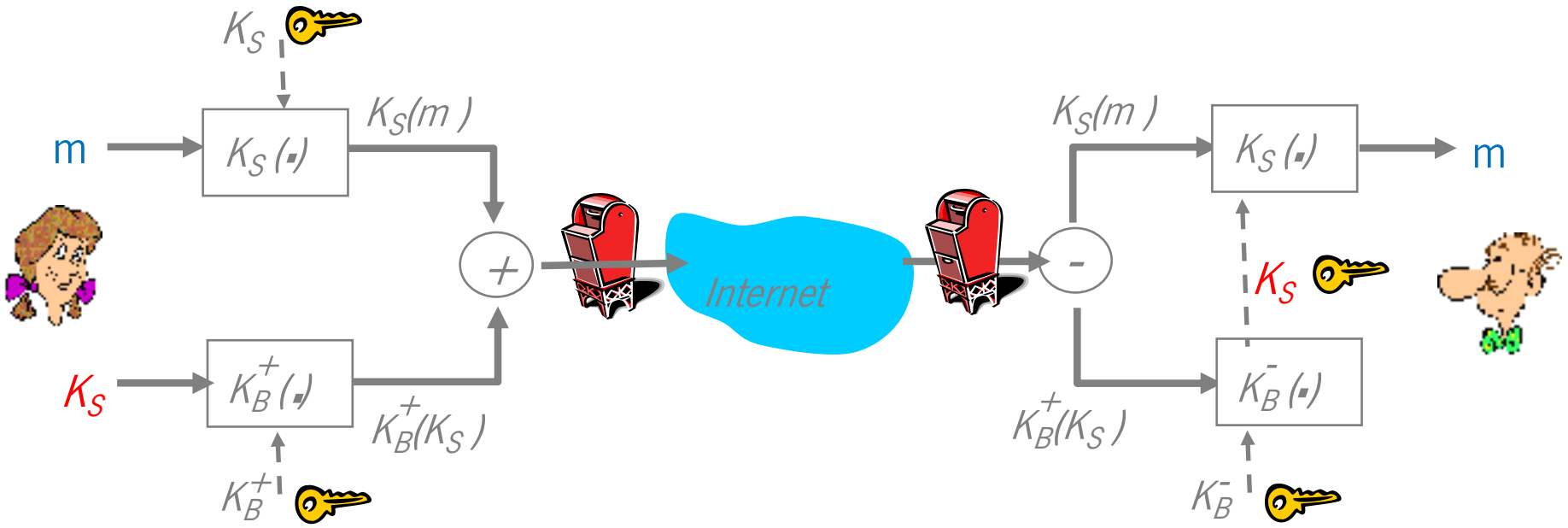


Usa apenas uma função de Hash criptográfica!

Não garante o não repúdio, pois se o segredo é partilhado, Alice e Bob conhecem a chave, e qualquer um deles pode produzir o Hash, não sendo por isso uma Assinatura Digital.

Envelope Digital

Alice envia mensagem confidencial para Bob (”envelope digital” selado)

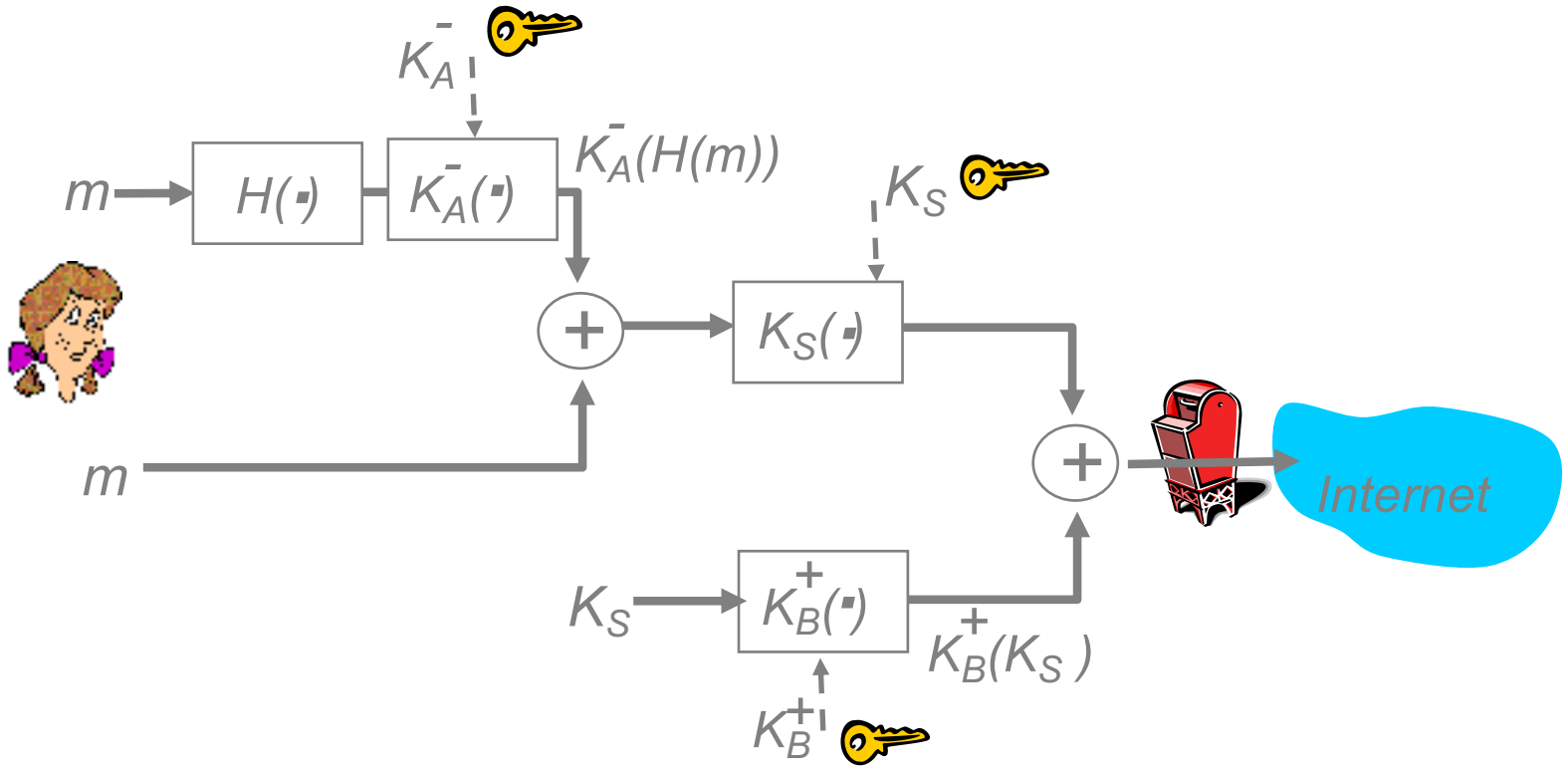


Alice:

- Gera uma chave simétrica secreta, K_S
- Cifra mensagem com K_S (por questões de eficiência)
- Cifra também a chave simétrica secreta K_S com a chave pública de Bob
- Envia ambos: $K_S(m)$ e $K_B(K_S)$ para Bob

Envelope digital (assinado e cifrado)

Para dar todas as garantias: confidencialidade, integridade, autenticação e não repúdio do originador



Alice usa três chaves: a sua chave privada, a chave pública do Bob, uma chave simétrica secreta gerada no momento

Infra-estrutura de chaves públicas (PKI)

Problema Chaves Simétricas:

- Como é que duas entidades estabelecem um segredo (a chave secreta) usando apenas a rede?

Solução:

- Centro de distribuição de chaves que seja de confiança e actua como intermediário entre as entidades

Problema Chaves Públicas

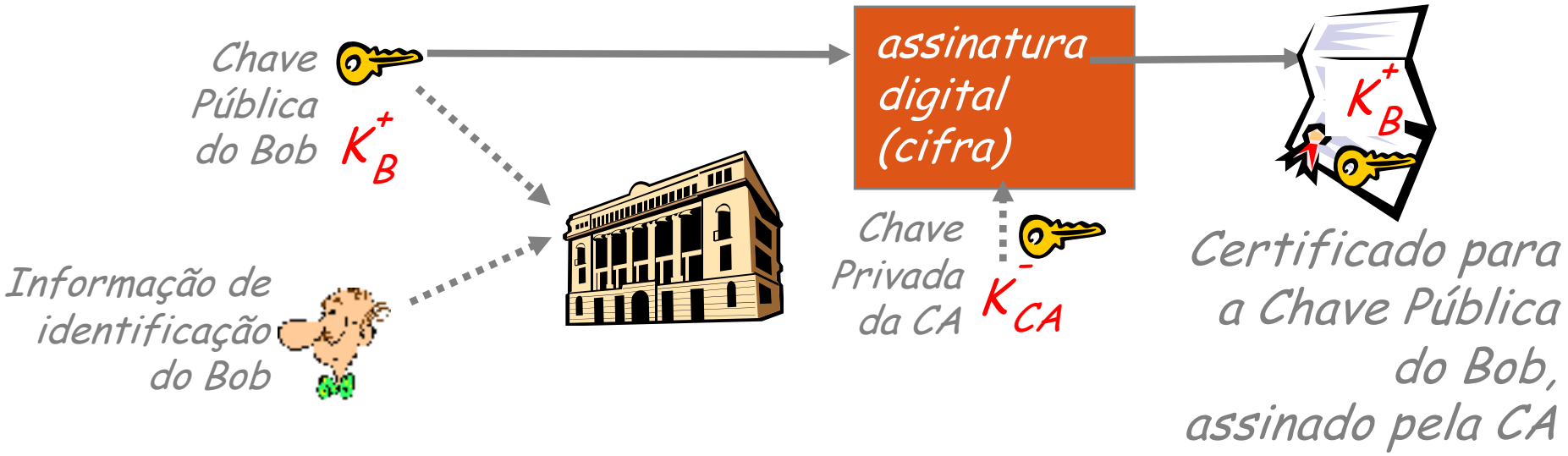
- Quando se obtém a chave pública da Alice ou do Bob na rede (e-mail, web, etc) como sabemos que são mesmo deles e não do intruso?

Solução:

- Autoridade de Certificação (CA) de confiança (*trusted certification authority*)

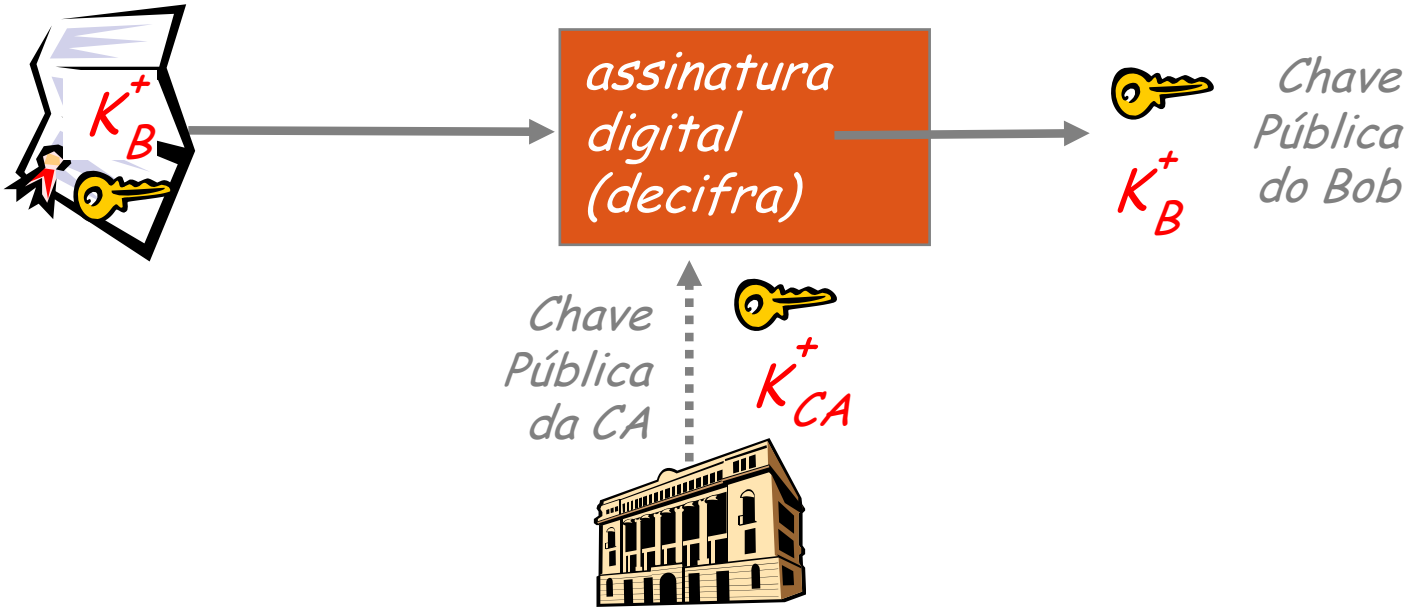
Autoridades de Certificação

- *Autoridade de Certificação (CA): associa a chave pública a uma determinada entidade, E*
- *E (pessoa, máquina,..) regista a sua chave pública na CA*
  - *E tem de fornecer uma provada de identidade a CA*
  - *CA cria um certificado digital associando E à sua chave pública*
  - *certificado contém a chave pública de E assinada digitalmente pela CA que assim assegura que "esta é a chave pública de E"*



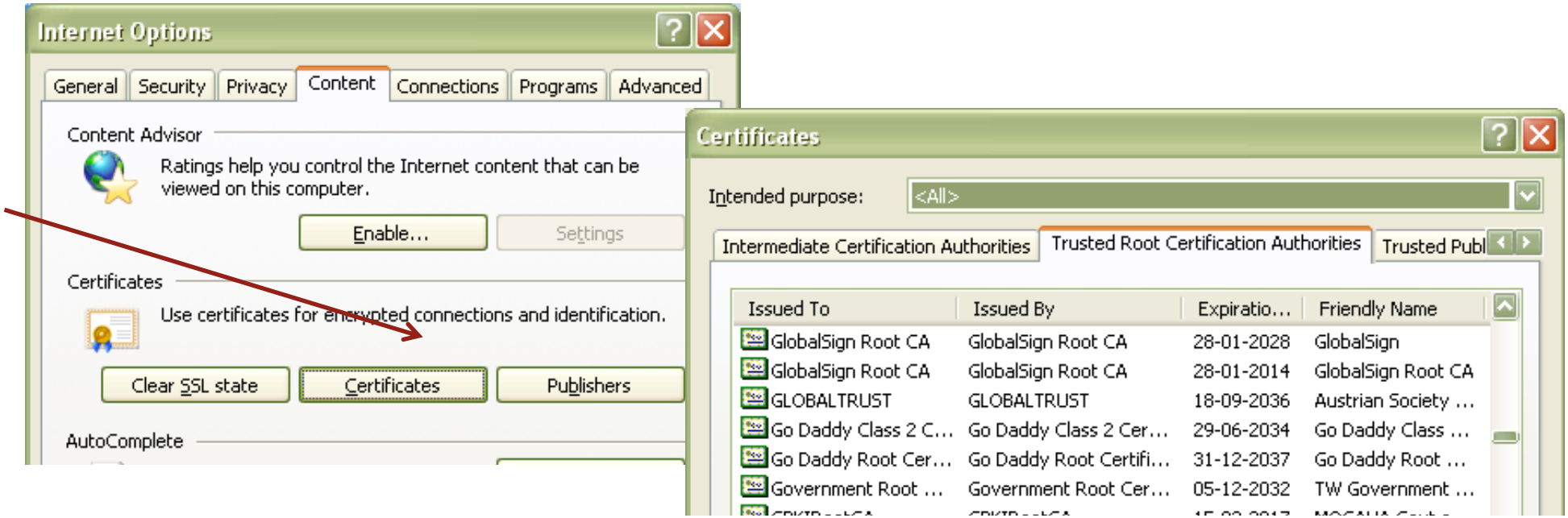
Autoridades de Certificação

- *Quando a Alice quer obter a chave pública do Bob:*
  - *obtem o certificado do Bob (dele mesmo ou doutros sítios).*
  - *aplica a chave pública da CA ao certificado para verificar a validade do certificado e extrair de lá a chave pública do Bob*

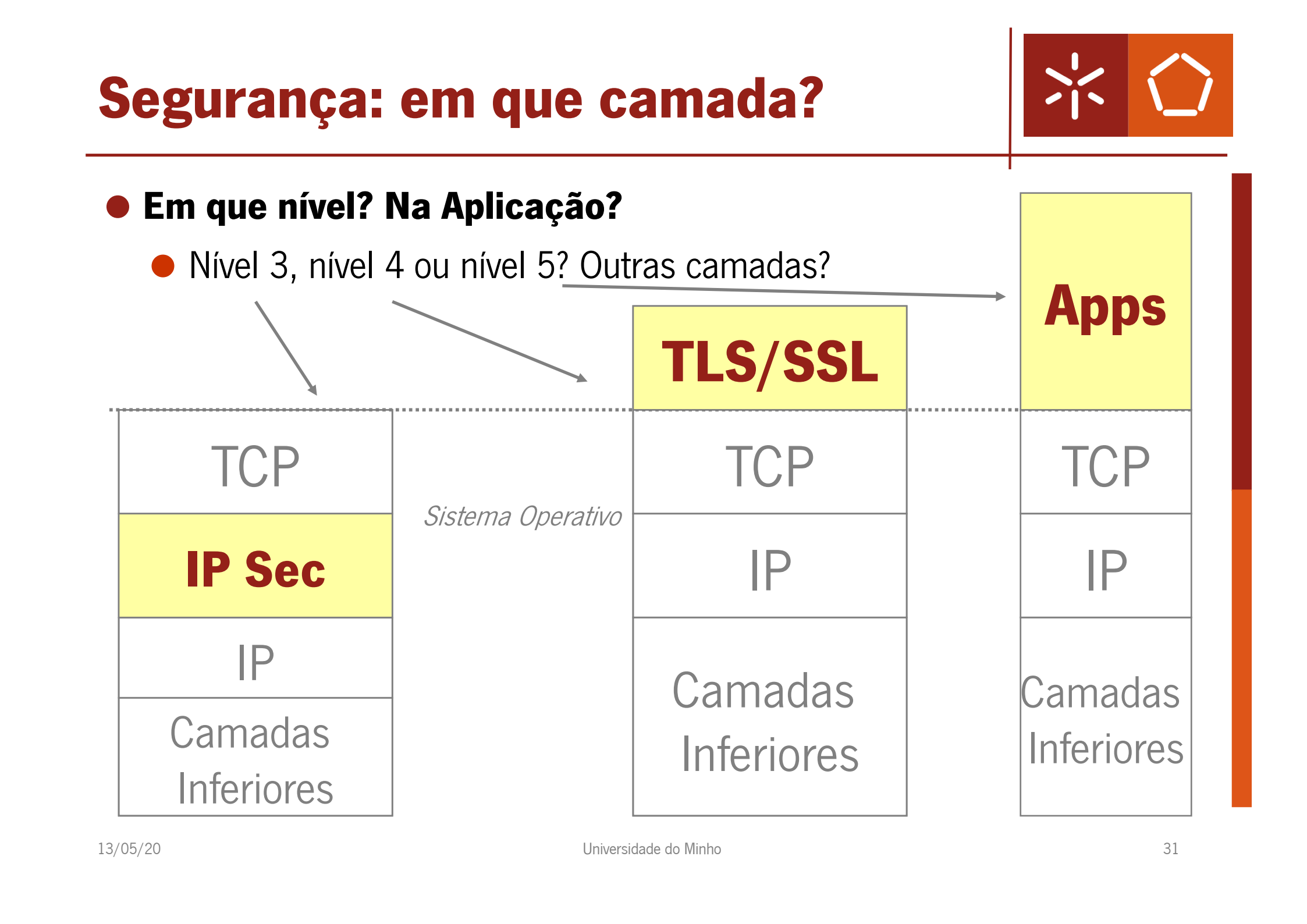


Autoridades de Certificação

- *Problema: como confiar no CA?*
  - *A mesma coisa?... mas?... problema!*
  - *Certificados de raíz (root certificates) instalados com as máquinas (Windows, Linux, ou seja lá o que for)*
    - *Esse é o momento decisivo para a criptografia de chave pública*



Segurança: em que camada?



- **Em que nível? Na Aplicação?**
  - Nível 3, nível 4 ou nível 5? Outras camadas?

Segurança: em que camada?

- **A nível 4 (TLS) as aplicações fazem interface com TLS e não com o TCP:**
  - Como o TCP não participa em nada, não consegue discernir pacotes inseridos maliciosamente na *stream*, desde que estejam corretos (*checksum*) e passa-os ao TLS...
- **A nível 3, as aplicações continuam a interagir com o TCP:**
  - As aplicações não precisam ser modificadas
  - Mas o nível IP só sabe com que IP está a trocar dados e não com que utilizador...
- **Também é possível a nível 5 (aplicação), mantendo compatibilidade com aplicações existentes:**
  - Exemplo: **S/MIME** ou PGP (compatível MAIL) sobre SMTP
  - Soluções especificas para uma dada aplicação...

Segurança: TLS

- **As diferentes versões SSL e TLS ao longo do tempo**

Protocol	Published	Standard	Observations
SSL 1.0	Never	–	Never published (developped by Netscape)
SSL 2.0	1995	–	Deprecated in 2011 (RFC 6176). Prohibited.
SSL 3.0	1996	RFC 6101 (Historic)	Historic RFC published in 2011. Deprecated in 2015 (RFC 7568)
TLS 1.0	1999	RFC 2246	Deprecated in 2020
TLS 1.1	2006	RFC 4346	Deprecated in 2020
TLS 1.2	2008	RFC 5246	
TLS 1.3	2018	RFC 8446	finalized in 2018 after 11 years and nearly 30 IETF drafts

"The differences between this protocol and SSL 3.0 are not dramatic, but they are significant enough that TLS 1.0 and SSL 3.0 do not interoperate."

The creators of TLS, RFC2246, Jan 1999

Segurança: TLS

- **Nível 4: TLS (Transport Layer Security) – muitas vezes ainda designado pelo nome inicial: Secure Sockets Layer (SSL)**
  - Segurança ao nível de transporte para qualquer aplicação TCP
  - Usado, por exemplo, no acesso a servidores HTTP, IMAP, SMTP
  - Serviços de segurança:
    - Autenticação do servidor e, opcionalmente, do cliente
    - Confidencialidade dos dados
  - **Autenticação do servidor**:
    - Cliente conhece chaves públicas de autoridades de certificação de sua confiança (CA)
    - Obtem certificado do servidor emitido por uma CA sua conhecida
    - Extrai chave pública do certificado depois de verificada validade

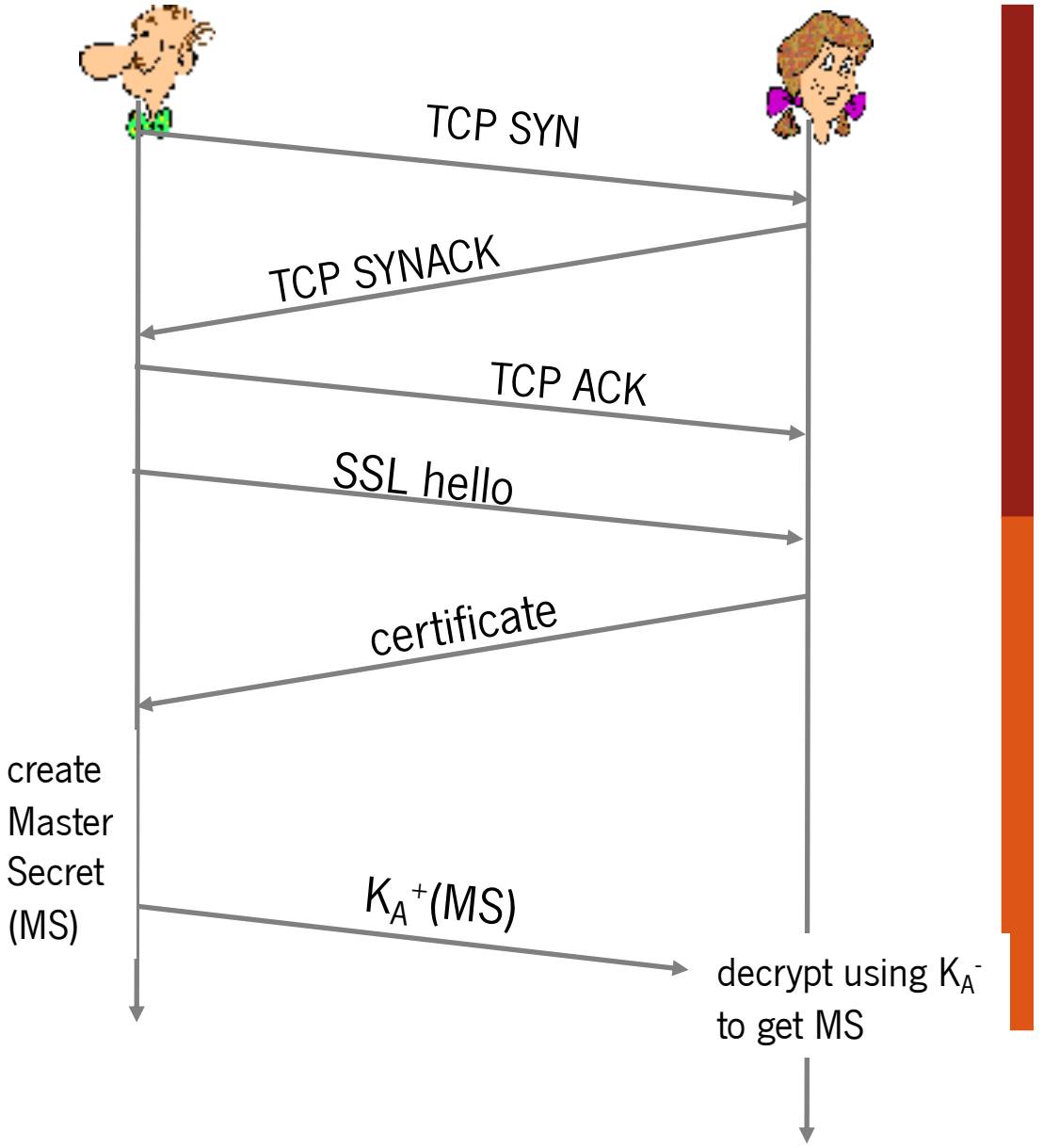
Segurança: TLS

- **Nível 4: TLS (Transport Layer Security)**
  - **Confidencialidade (cifragem dos dados da sessão)**
    - Cliente gera chave de sessão, cifra-a com a chave pública do servidor e envia-a ao servidor
    - Servidor decifra a chave de sessão usando a sua chave privada
    - Ambos – cliente e servidor – na posse da chave de sessão, podem cifrar todos os dados trocados...
  - **Autenticação do cliente** pode ser feita com base em certificados do cliente
    - Opcional, porque a maioria dos clientes não possui chave pública e chave privada com certificado verificável

Exemplo TLS1.0-1.2/SSL3.0: 3 fases

1. *Handshake* inicial:

- Bob estabelece conexão TCP com Alice
- Autentica Alice usando o certificado assinado por uma CA
- Cria chave mestra, encripta-a (usando a chave pública da Alice), e envia-a à Alice
  - Incompleto: a troca de um "nonce" não está ilustrada



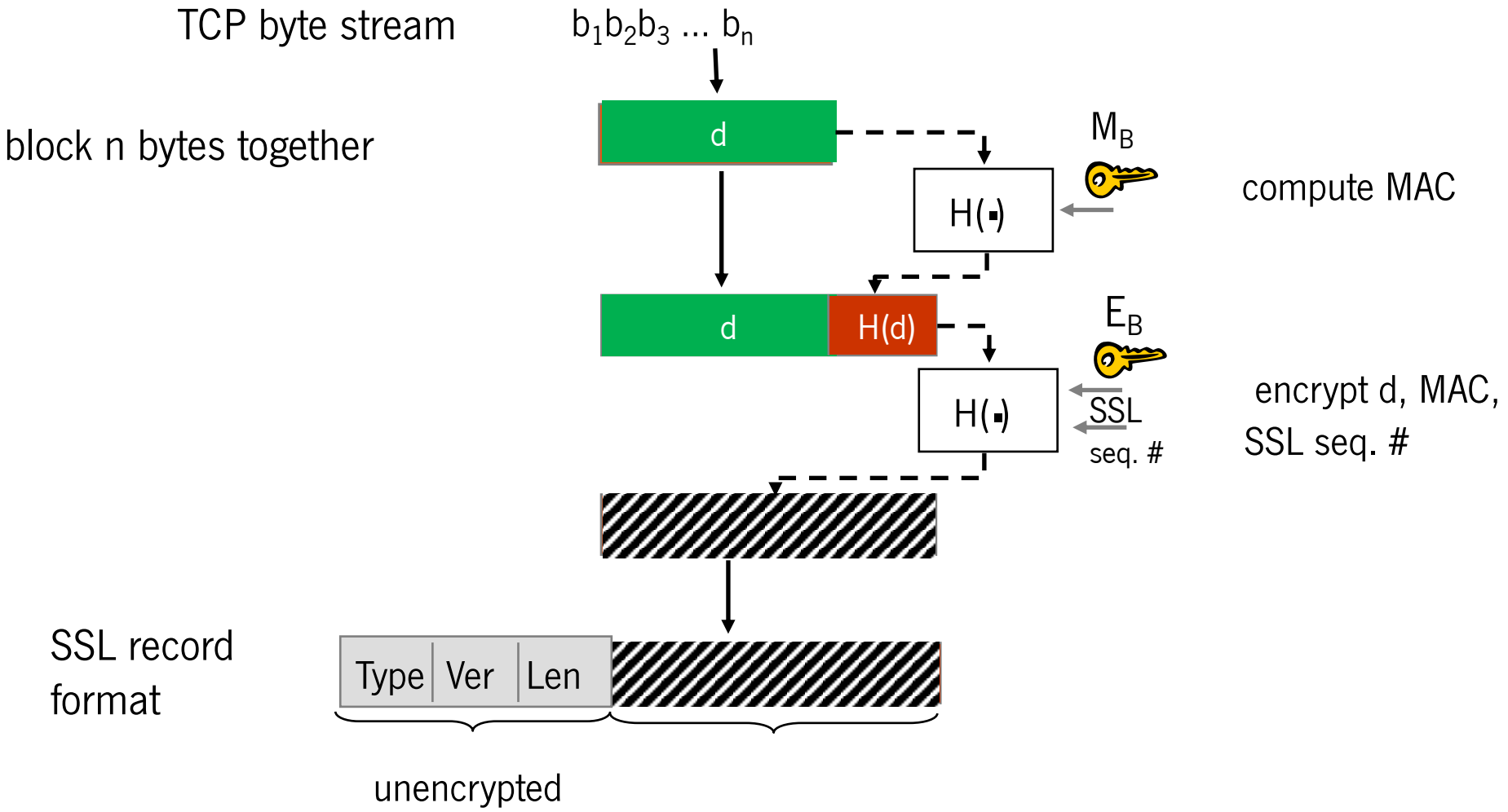
Exemplo TLS1.0-1.2/SSL3.0: 3 fases

2. Cálculo das chaves:

- Alice e Bob usam a chave mestra (MS) para gerar 4 chaves:
  - **E_B:** *Bob->Alice* chave de cifragem de dados
  - **E_A:** *Alice->Bob* chave de cifragem de dados
  - **M_B:** *Bob->Alice* chave MAC (*Message Authentication Code*)
  - **M_A:** *Alice->Bob* chave MAC (*Message Authentication Code*)
- Os algoritmos (cifragem e MAC) são negociados entre a Alice e o Bob
- Porquê 4 chaves?

Exemplo TLS1.0-1.2/SSL3.0: 3 fases

3. Transferência dados

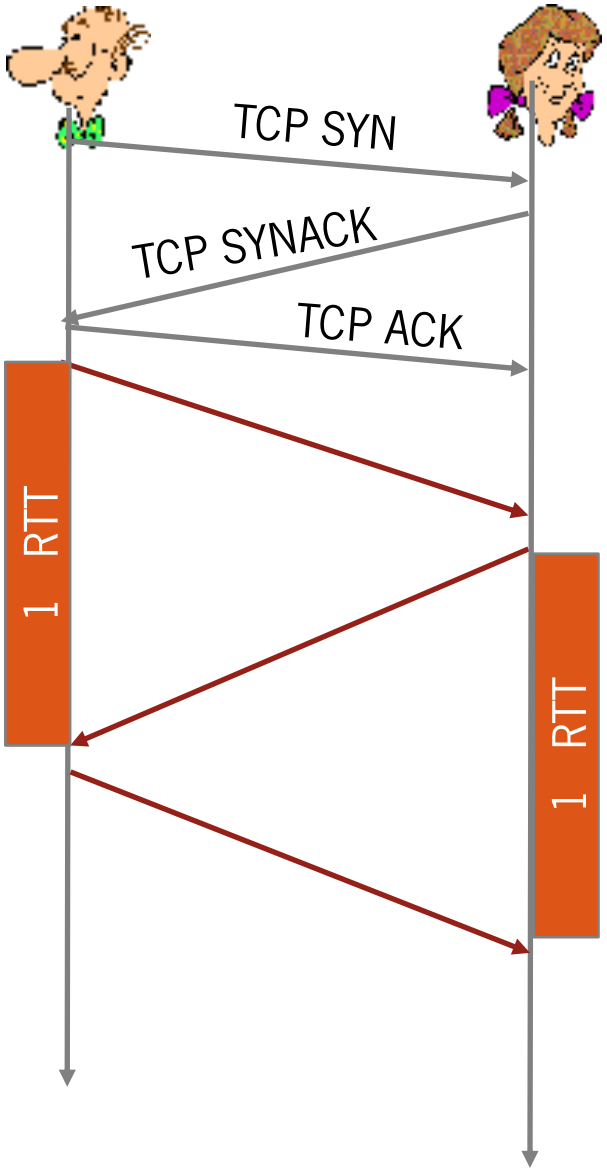


Segurança: TLS 1.3 (2020)

- **Objetivos principais**
  - Segurança melhorada → falhas detetadas, cifras obsoletas, etc.
  - Melhoria de desempenho → Reduzir o *handshake* inicial a 1-RTT ou 0-RTT
- **Principais diferenças para o TLS 1.2**
  - Remover mecanismos e algoritmos obsoletos (Hash: MD5, SHA-1; Simétricos: DES, 3DES, AES-CBC, RC4; chave pública: RSA estático, Diffie-Helman estático...)
  - Todas as mensagens de handshake a seguir ao ServerHello são cifradas
  - Novas funções para derivar chaves
  - Reduzir o handshake inicial (evitar a negociação do primeiro RTT)
  - Retoma de uma sessão com ou sem manutenção de estado do lado do servidor
  - Permitir um modo especial com 0 – RTT !!

Segurança: TLS 1.3 (2020)

- **Handshake com 1-RTT**



Key Exch
- ClientHello
- + key_share*
- + signature_algorithms*
- + psk_key_exchange_modes*
- + pre_shared_key*

Auth
- {Certificate*}
- {CertificateVerify*}
- {Finished}
- **[Application Data]**

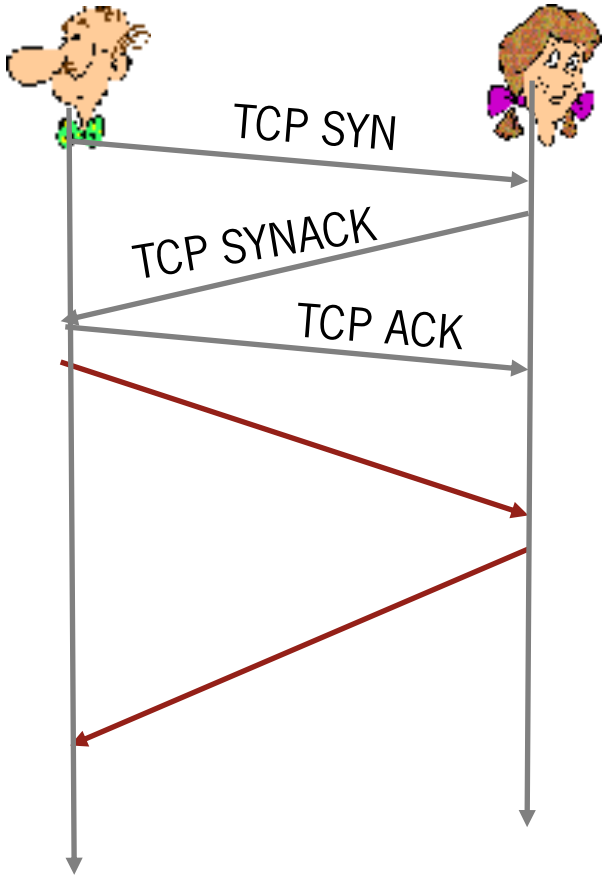
Key Exch
- ServerHello
- + key_share*
- + pre_shared_key*

Server Parems
- {EncryptedExtensions}
- {CertificateRequest*}

Auth
- {Certificate*}
- {CertificateVerify*}
- {Finished}
- **[Application Data*]**

Segurança: TLS 1.3 (2020)

- **Handshake com 0-RTT**



ClientHello
+ early_data
+ key_share*
+ psk_key_exchange_modes
+ pre_shared_key*
(Application Data*)

(EndOfEarlyData)
{Finished}
[Application Data]

ServerHello
+ pre_shared_key
+ key_share*
{EncryptedExtensions}
early_data
{Finished}
[Application Data*]

Quando cliente e servidor partilham uma Pre-SharedKey, obtida externamente ou num handshake anterior, os clients podem enviar dados no primeiro segmento (early_data), usando o PSK para autenticar o servidor e encriptar os dados. Semelhante à retomada de uma sessão

Exemplo de uso – E-Mail

S/MIME (Ex: OpenSSL)
→ Faz uso de PKI

PGP (Ex: OpenPGP) Pretty Good Privacy
→ Não faz uso de PKI

- **Fazer demo com E-mail**
  1. Adicionar um certificado válido ao cliente E-mail
  2. Consultar certificados de terceiros...
  3. Assinar e/ou "cifrar" uma mensagem...
  4. Verificar a assinatura