

Chapter 8

Security

A note on the use of these PowerPoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part.

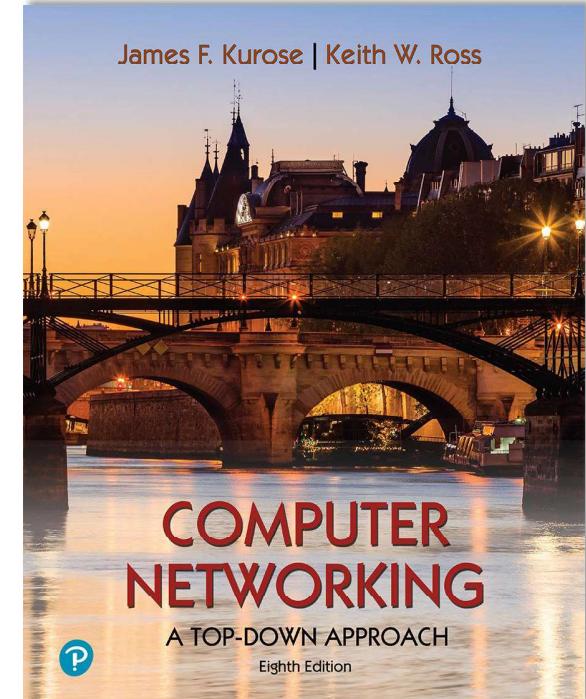
In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

For a revision history, see the slide note for this page.

Thanks and enjoy! JFK/KWR

All material copyright 1996-2020
J.F Kurose and K.W. Ross, All Rights Reserved



*Computer Networking: A
Top-Down Approach*
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

Roteiro do Capítulo 8

- O que é segurança de rede?
- Princípios de criptografia
- Integridade de mensagem, autenticação
- Protegendo o e-mail
- Protegendo conexões TCP: TLS



O que é segurança de rede?

confidencialidade: apenas o remetente e o destinatário pretendido devem “entender” o conteúdo da mensagem

- remetente criptografa mensagem
- destinatário descriptografa mensagem

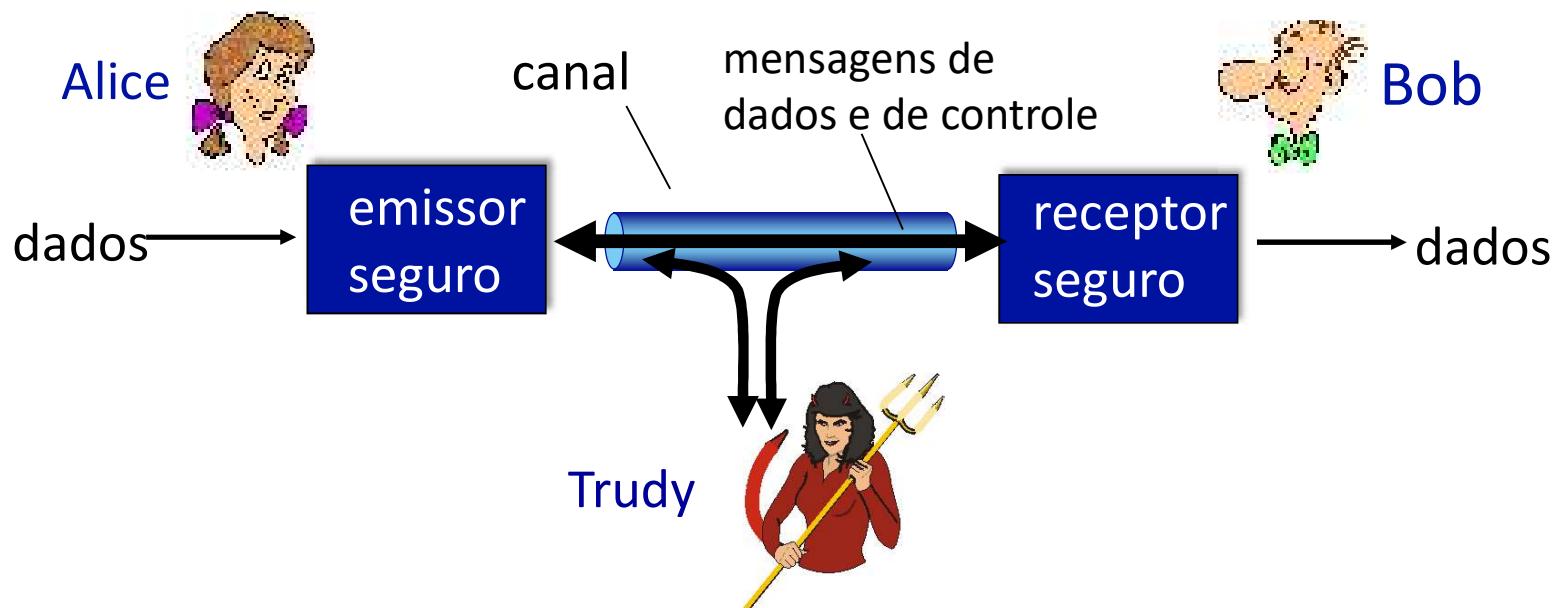
autenticação: remetente e destinatário querem confirmar a identidade um do outro

integridade de mensagem: remetente e destinatário desejam garantir que a mensagem não seja alterada (em trânsito ou posteriormente) sem detecção

acesso e disponibilidade: os serviços devem estar acessíveis e disponíveis para os usuários

Amigos e inimigos: Alice, Bob, Trudy

- bem conhecidos no mundo da segurança de rede
- Bob e Alice (amigos!) querem se comunicar “com segurança”
- Trudy (intrusa) pode interceptar, excluir, adicionar mensagens



Amigos e inimigos: Alice, Bob, Trudy

Quem seriam Bob e Alice?

- ... bem, Bobs e Alices da *vida real*!
- navegador/servidor da Web para transações eletrônicas (por exemplo, compras online)
- cliente/servidor de banco online
- servidores DNS
- roteadores BGP trocando atualizações da tabela de roteamento
- outros exemplos?

Existem bandidos por aí!

Q: O que um “cara mau” pode fazer?

A: Muito! (lembre-se da seção 1.6)

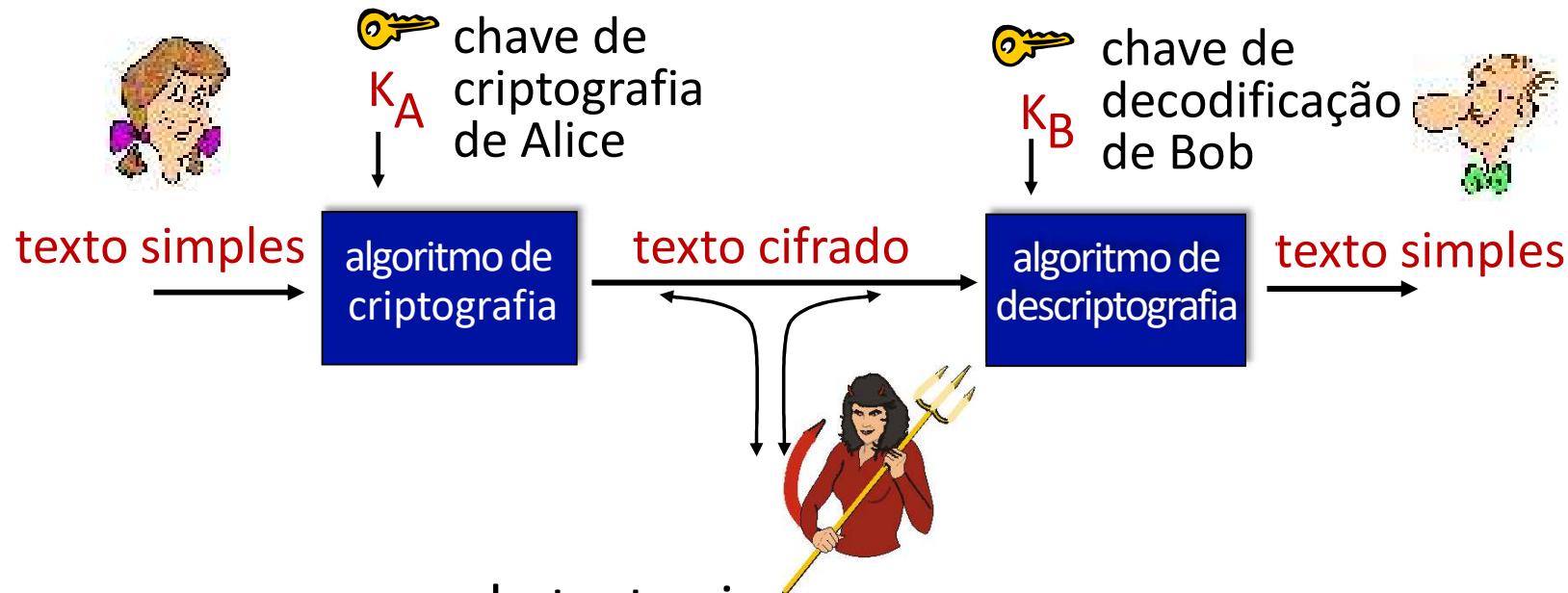
- **bisbilhotar (*eavesdrop*):** interceptar mensagens
- **inserir**ativamente mensagens na conexão
- **personificação:** pode falsificar (*spoof*) o endereço de origem no pacote (ou qualquer campo no pacote)
- **sequestro (*hijacking*):** “assumir” a conexão em andamento, removendo o remetente ou o destinatário, inserindo-se no lugar
- **negação de serviço (*denial of service*):** impedir que o serviço seja usado por outros (por exemplo, sobrecarregando recursos)

Roteiro do Capítulo 8

- O que é segurança de rede?
- Princípios de criptografia
- Integridade de mensagem, autenticação
- Protegendo o e-mail
- Protegendo conexões TCP: TLS



A linguagem da criptografia



m: mensagem de texto simples

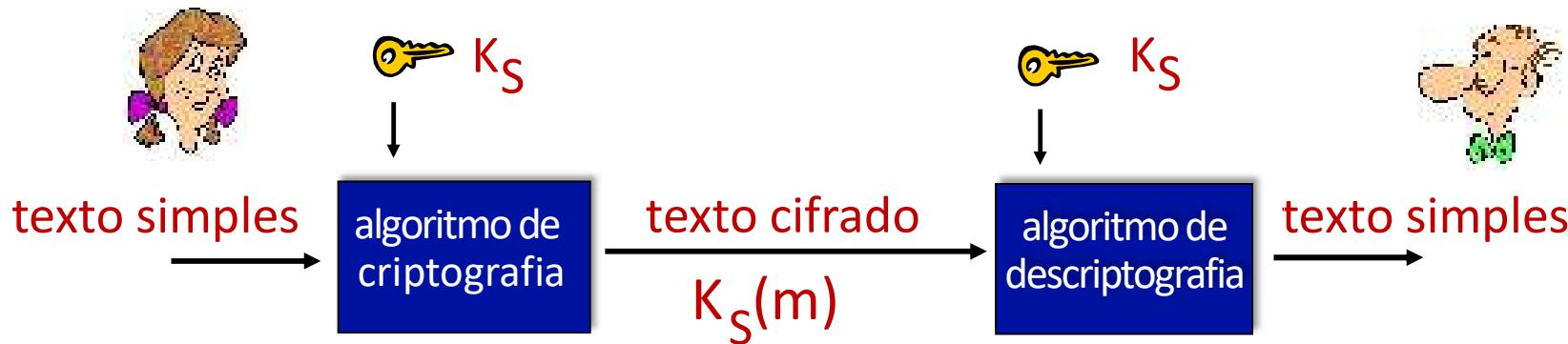
$K_A(m)$: texto cifrado, criptografado com a chave K_A

$m = K_B(K_A(m))$

Quebrando um esquema de criptografia

- ataque apenas de texto cifrado: Trudy tem texto cifrado que ela pode analisar
- duas abordagens:
 - força bruta: pesquisa em todas as chaves
 - análise estatística
- ataque de texto simples conhecido: Trudy tem texto simples correspondente ao texto cifrado
 - *por exemplo*, em cifra monoalfabética, Trudy determina pares para a,l,i,c,e,b,o,
- ataque de texto simples escolhido: Trudy pode obter texto cifrado para texto simples escolhido

Criptografia de chave simétrica



Criptografia de chave simétrica: Bob e Alice compartilham a mesma chave (simétrica): K

- *por exemplo*, a chave é conhecer o padrão de substituição em uma cifra de substituição mono alfabética

Q: como Bob e Alice concordam com o valor da chave?

Esquema de criptografia simples

cifra de substituição: substituindo uma coisa por outra

- cifra monoalfabética: substitui uma letra por outra

texto simples: abcdefghijklmnopqrstuvwxyz

texto cifrado: mnbvcxzasdfghjklpoiuytrewq

exemplo:
texto simples: bob. i love you. alice
texto cifrado: nkn. s gktc wky. mgsbc



Chave de encriptação: mapeamento de um conjunto de 26 letras para um conjunto de 26 letras

Uma abordagem de criptografia mais sofisticada

- n cifras de substituição, M_1, M_2, \dots, M_n
 - padrão cíclico:
 - exemplo: n=4: $M_1, M_3, M_4, M_3, M_2; M_1, M_3, M_4, M_3, M_2; \dots$
 - para cada novo símbolo de texto simples, use o padrão de substituição subsequente no padrão cíclico
 - dog: d do M_1 , o do M_3 , g do M_4
-  **Chave de encriptação:** n cifras de substituição e padrão cíclico
- a chave não precisa ser apenas um padrão de n bits

Criptografia de chaves simétricas

KUROSE | ROSS
Redes de computadores e a internet
uma abordagem top-down
6^a edição

- Uma cifra monoalfabética

Letra no texto aberto: a b c d e f g h i j k l m n o p q r s t u v w x y z
Letra no texto cifrado: m n b v c x z a s d f g h j k l p o i u y t r e w q

- Uma cifra polialfabética que utiliza duas cifras de César

Letra do texto aberto: a b c d e f g h i j k l m n o p q r s t u v w x y z
 $C_1(k = 5)$: f g h i j k l m n o p q r s t u v w x y z a b c d e
 $C_2(k = 19)$: t u v w x y z a b c d e f g h i j k l m n o p q r s

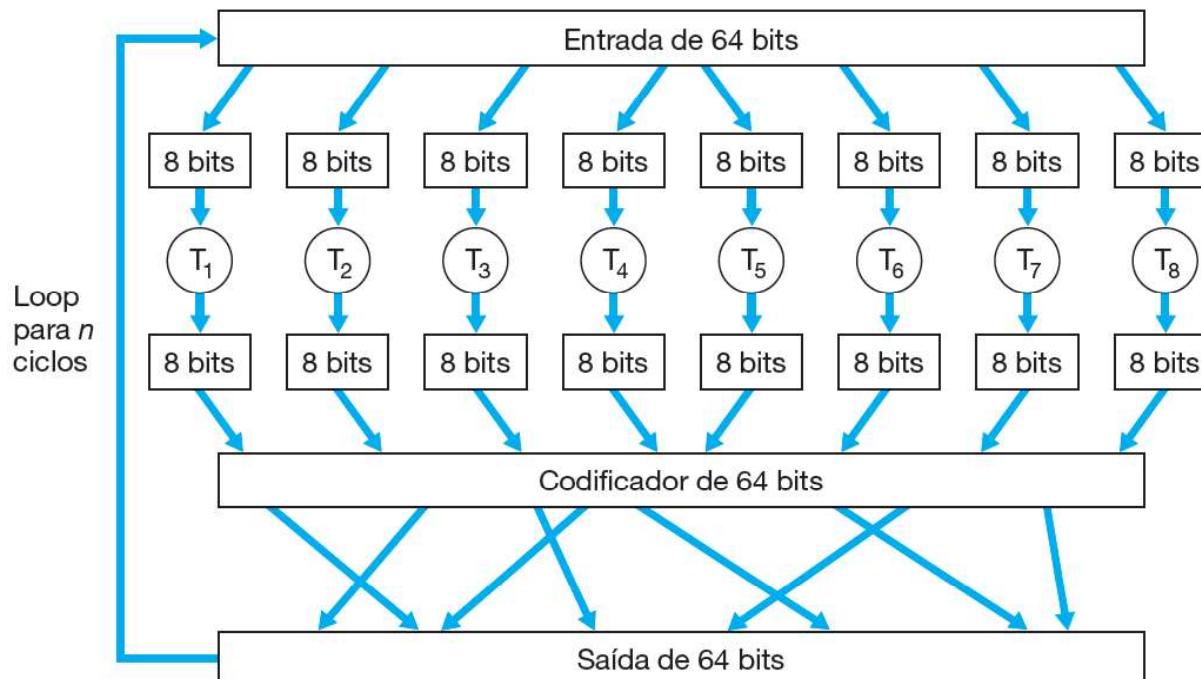
- Uma cifra de bloco de 3 bits específica

Entrada	Saída	Entrada	Saída
000	110	100	011
001	111	101	010
010	101	110	000
011	100	111	001

Criptografia de chaves simétricas

KUROSE | ROSS
Redes de computadores e a internet
uma abordagem top-down
6^a edição

- Exemplo de uma cifra de bloco



Criptografia de chaves simétricas

KUROSE | ROSS
Redes de computadores e a internet
uma abordagem top-down
6^a edição

- As cifras de bloco em geral usam uma técnica chamada **Encadeamento do Bloco de Cifra (CBC)**.
- A ideia básica é enviar somente um valor aleatório junto com a primeira mensagem e, então, fazer o emissor e o receptor usarem blocos codificados em vez do número aleatório subsequente.
- O CBC possui uma consequência importante: é preciso fornecer um mecanismo dentro do protocolo para distribuir o vetor de inicialização do emissor ao receptor.

Criptografia de chave simétrica: DES

DES: Data Encryption Standard

- padrão de criptografia dos EUA [NIST 1993]
- chave simétrica de 56 bits, entrada de texto simples de 64 bits
- cifra de bloco com encadeamento de bloco de cifra
- Quão seguro é o DES?
 - Desafio do DES: frase criptografada com chave de 56 bits descriptografada (força bruta) em menos de um dia
 - nenhum bom ataque analítico conhecido
- tornando o DES mais seguro:
 - 3DES: criptografar 3 vezes com 3 chaves diferentes

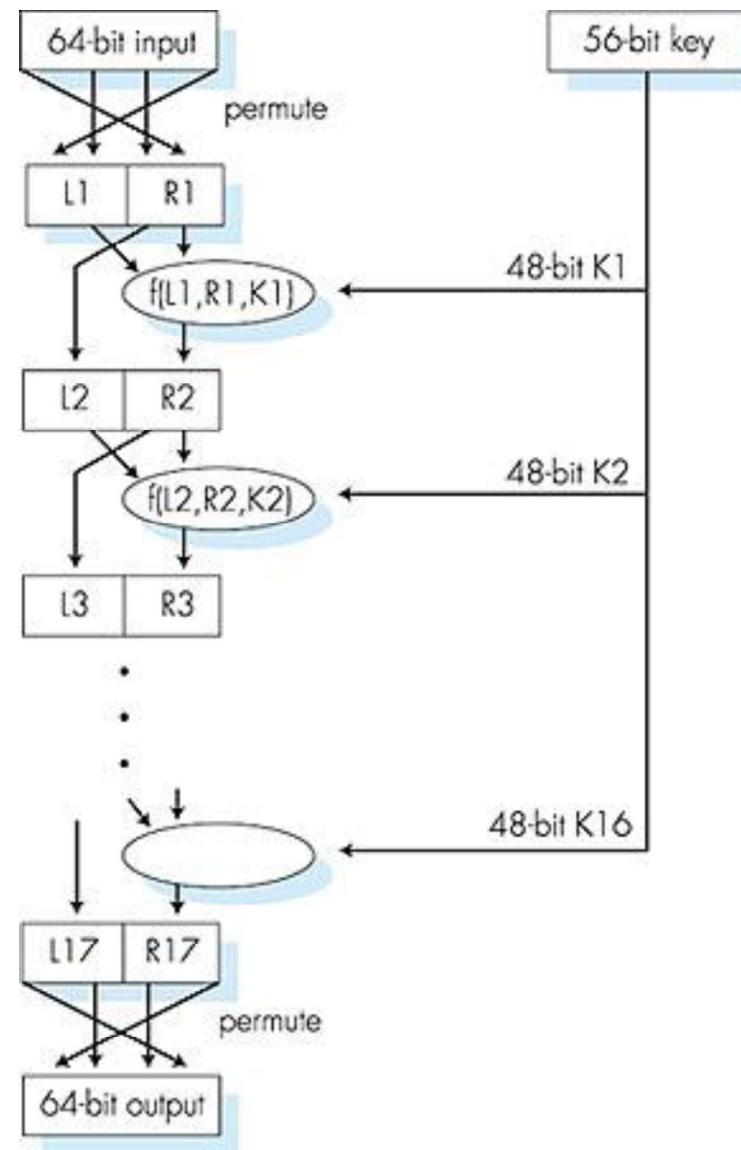
Criptografia de chave simétrica: DES

Operação do DES

permutação inicial

16 “rodadas” idênticas de aplicação de função, cada uma usando uma chave de 48 bits diferente

permutação final



AES: Advanced Encryption Standard

- padrão do NIST de chave simétrica, substituiu o DES (novembro de 2001)
- processa dados em blocos de 128 bits
- chaves de 128, 192 ou 256 bits
- descriptografia de força bruta (tentar cada chave) em uma máquina que levaria 1 segundo com o DES, levaria 149 trilhões de anos com o AES

Criptografia de Chave Pública

criptografia de chave simétrica:

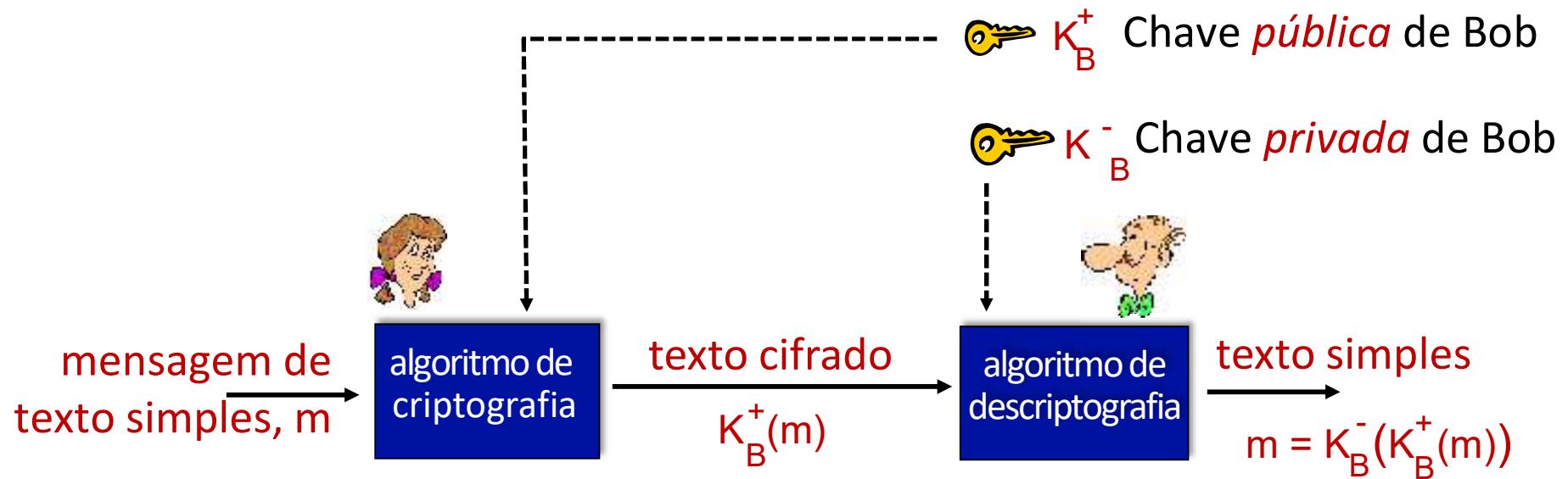
- requer que o remetente e o destinatário conheçam a chave secreta compartilhada
- Q: como chegar a um acordo sobre a chave em primeiro lugar (particularmente se nunca se “encontraram”)?

criptografia de chave pública

- abordagem *radicalmente* diferente [Diffie-Hellman76, RSA78]
- remetente, destinatário *não* compartilham chave secreta
- chave de criptografia *pública* conhecida por *todos*
- chave de descriptografia *privada* conhecida apenas pelo destinatário



Criptografia de Chave Pública



Wow - a criptografia de chave pública revolucionou a criptografia de 2.000 anos (anteriormente apenas de chave simétrica)!

- ideias semelhantes surgiram mais ou menos na mesma época, de forma independente nos EUA e no Reino Unido (confidencial)

Algoritmos de criptografia de chave pública

requisitos:

- 1 precisa de $K_B^+(\cdot)$ e $K_B^-(\cdot)$ tal que:

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

- 2 dada a chave pública K_B^+ , deve ser impossível computar a chave privada K_B^-

RSA: algoritmo de Rivest, Shamir e Adelson

Pré-requisito: aritmética modular

- $x \bmod n =$ resto de x na divisão por n
- fatos:
 - $[(a \bmod n) + (b \bmod n)] \bmod n = (a+b) \bmod n$
 - $[(a \bmod n) - (b \bmod n)] \bmod n = (a-b) \bmod n$
 - $[(a \bmod n) * (b \bmod n)] \bmod n = (a*b) \bmod n$
- portanto
$$(a \bmod n)^d \bmod n = a^d \bmod n$$
- exemplo: $x=14$, $n=10$, $d=2$:
$$(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$$
$$x^d = 14^2 = 196 \quad x^d \bmod 10 = 6$$

RSA: se preparando

- mensagem: apenas um padrão de bit
- um padrão de bits pode ser representado por um número inteiro único
- assim, criptografar uma mensagem é equivalente a criptografar um número

exemplo:

- $m = 10010001$. Esta mensagem é representada exclusivamente pelo número decimal 145.
- para criptografar m , criptografamos o número correspondente, que fornece um novo número (o texto cifrado).

RSA: Criando par de chaves pública/privada

1. escolha dois grandes números primos p e q . (por exemplo, 1024 bits cada)
2. compute $n = pq$, $z = (p-1)(q-1)$
3. escolha e (com $e < n$) que não tem fatores comuns com z (e e z são “relativamente primos”).
4. escolha d tal que $ed-1$ é exatamente divisível por z . (em outras palavras: $ed \bmod z = 1$).
5. a chave *pública* é $\underbrace{(n,e)}_{K_B^+}$. A chave *privada* é $\underbrace{(n,d)}_{K_B^-}$.

RSA: criptografia, descriptografia

0. dado (n,e) e (n,d) conforme computado anteriormente

1. para criptografar mensagem $m (< n)$, compute

$$c = m^e \text{ mod } n$$

2. para descriptografar o padrão de bits recebido, c , compute

$$m = c^d \text{ mod } n$$

a mágica acontece!

$$m = (\underbrace{m^e \text{ mod } n}_c)^d \text{ mod } n$$

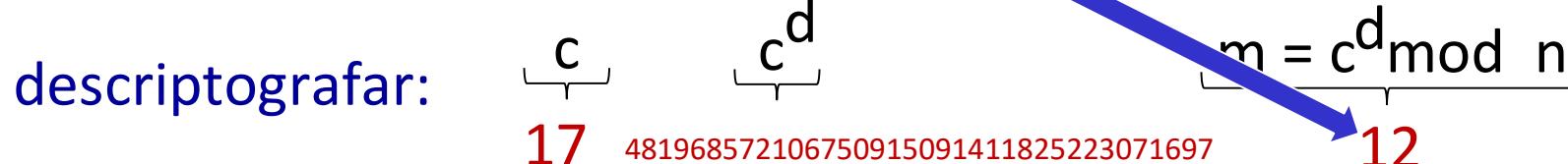
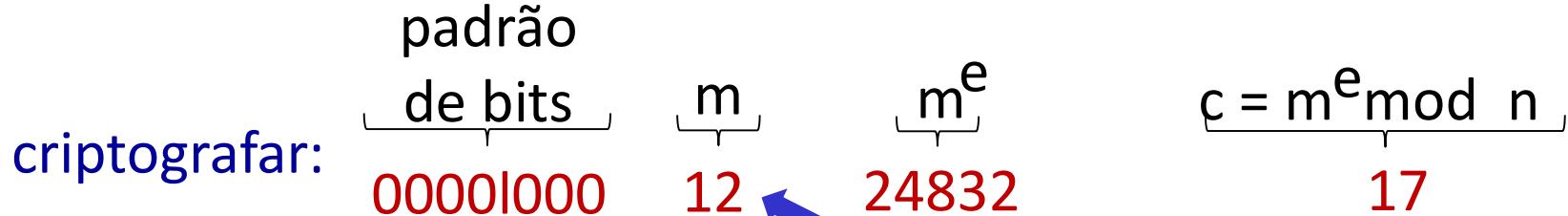
Exemplo de RSA:

Bob escolhe $p=5$, $q=7$. Então $n=35$, $z=24$.

$e=5$ (portanto e e z são relativamente primos).

$d=29$ (portanto $ed-1$ é exatamente divisível por z).

criptografando mensagens de 8 bits.



Por que o RSA funciona?

- Precisamos mostrar que $c^d \text{ mod } n = m$, onde $c = m^e \text{ mod } n$
- fato: para qualquer x e y : $x^y \text{ mod } n = x^{(y \text{ mod } z)} \text{ mod } n$
 - onde $n = pq$ e $z = (p-1)(q-1)$
- portanto,
$$\begin{aligned} c^d \text{ mod } n &= (m^e \text{ mod } n)^d \text{ mod } n \\ &= m^{ed} \text{ mod } n \\ &= m^{(ed \text{ mod } z)} \text{ mod } n \\ &= m^1 \text{ mod } n \\ &= m \end{aligned}$$

RSA: outra propriedade importante

A seguinte propriedade será *muito* útil mais tarde :

$$\underbrace{K_B^-(K_B^+(m))}_{\text{use a chave}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{use a chave}}$$

use a chave
pública primeiro,
seguida pela
chave privada

use a chave
pública primeiro,
seguida pela
chave privada

o resultado é o mesmo!

Por que $K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$?

segue diretamente da aritmética modular:

$$\begin{aligned}(m^e \bmod n)^d \bmod n &= m^{ed} \bmod n \\&= m^{de} \bmod n \\&= (m^d \bmod n)^e \bmod n\end{aligned}$$

Por que o RSA é seguro?

- suponha que você conheça a chave pública de Bob (n,e). Quão difícil é determinar d ?
- essencialmente é necessário encontrar fatores de n sem conhecer os dois fatores p e q
 - fato: fatorar um número grande é difícil

RSA na prática: chaves de sessão

- a exponenciação em RSA é computacionalmente intensiva
- o DES é pelo menos 100 vezes mais rápido que o RSA
- use a criptografia de chave pública para estabelecer uma conexão segura e, em seguida, estabeleça a segunda chave – chave de sessão simétrica – para criptografar dados

chave de sessão, K_s

- Bob e Alice usam RSA para trocar uma chave de sessão simétrica K_s
- uma vez que ambos tenham K_s , eles usam criptografia de chave simétrica

Roteiro do Capítulo 8

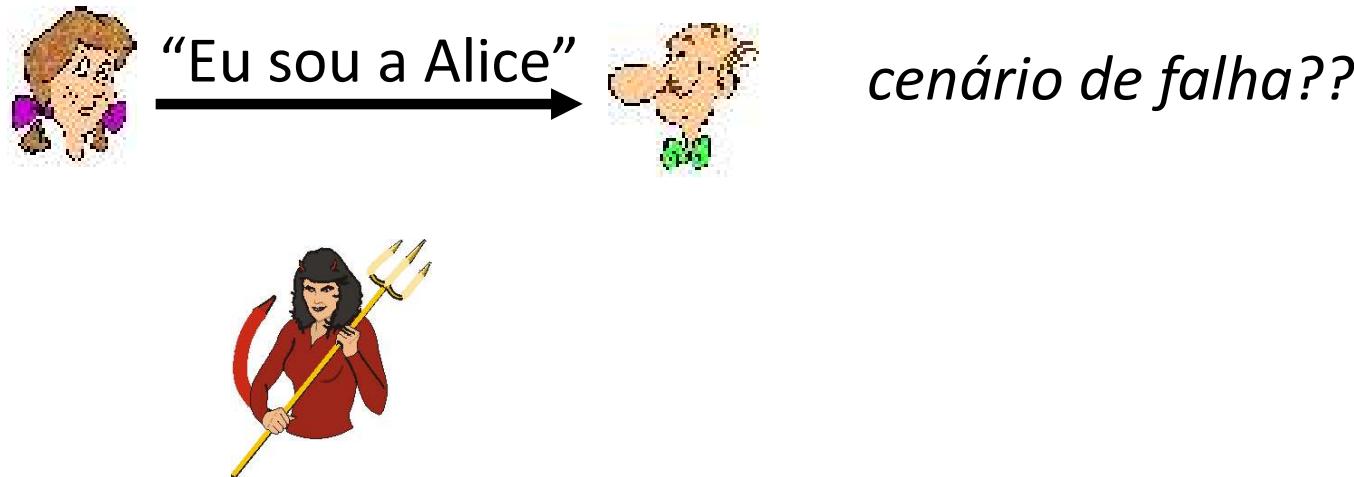
- O que é segurança de rede?
- Princípios de criptografia
- **Autenticação, integridade de mensagem**
- Protegendo o e-mail
- Protegendo conexões TCP: TLS



Autenticação

Objetivo: Bob quer que Alice “prove” sua identidade para ele

Protocolo ap1.0: Alice diz “Eu sou a Alice”



Autenticação

Objetivo: Bob quer que Alice “prove” sua identidade para ele

Protocolo ap1.0: Alice diz “Eu sou a Alice”



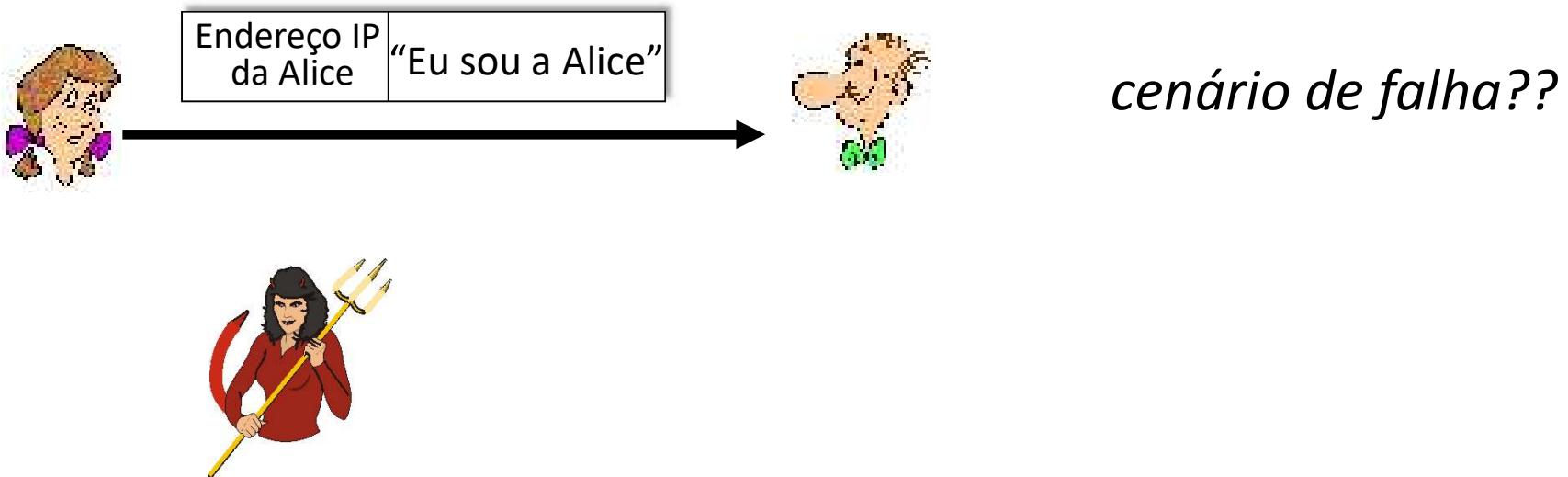
*em uma rede, Bob
não pode “ver”
Alice, então Trudy
simplesmente se
declara Alice*



Autenticação: outra tentativa

Objetivo: Bob quer que Alice “prove” sua identidade para ele

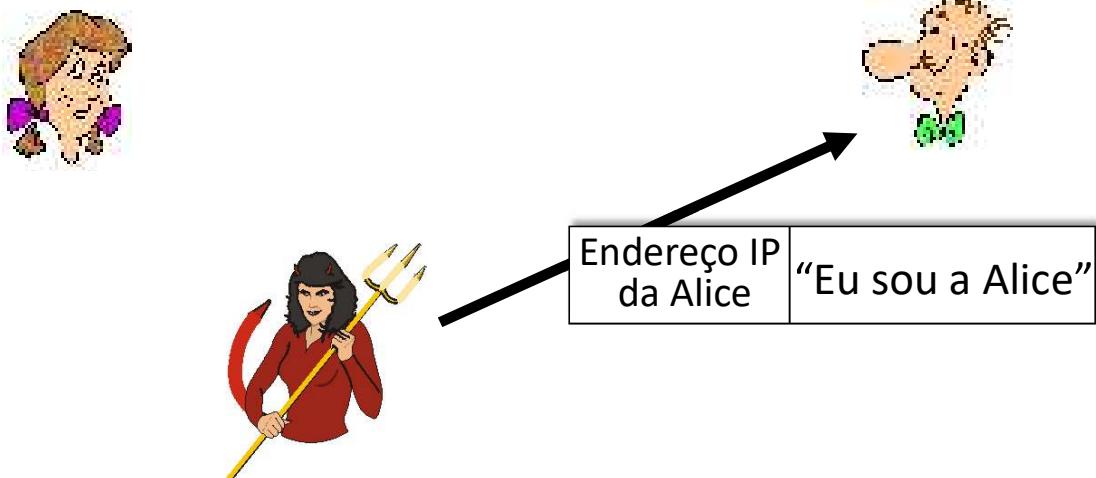
Protocolo ap2.0: Alice diz “Eu sou a Alice” em um pacote IP contendo seu endereço IP de origem



Autenticação: outra tentativa

Objetivo: Bob quer que Alice “prove” sua identidade para ele

Protocolo ap2.0: Alice diz “Eu sou a Alice” em um pacote IP contendo seu endereço IP de origem

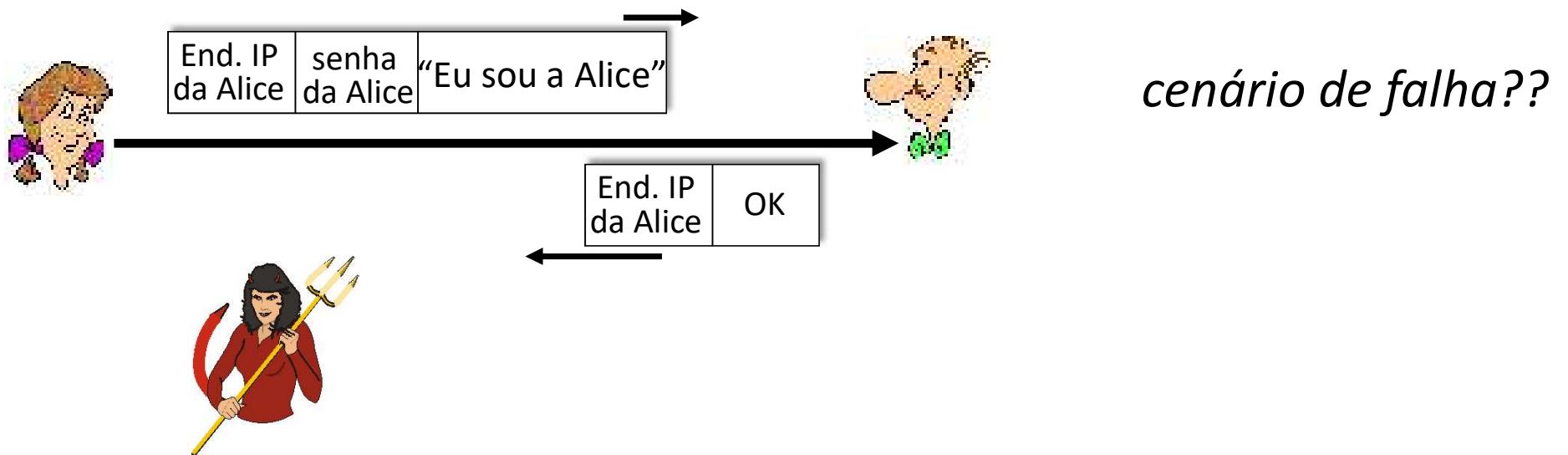


Trudy pode criar um pacote “falsificando” (“spoofing”) o endereço de Alice

Autenticação: uma terceira tentativa

Objetivo: Bob quer que Alice “prove” sua identidade para ele

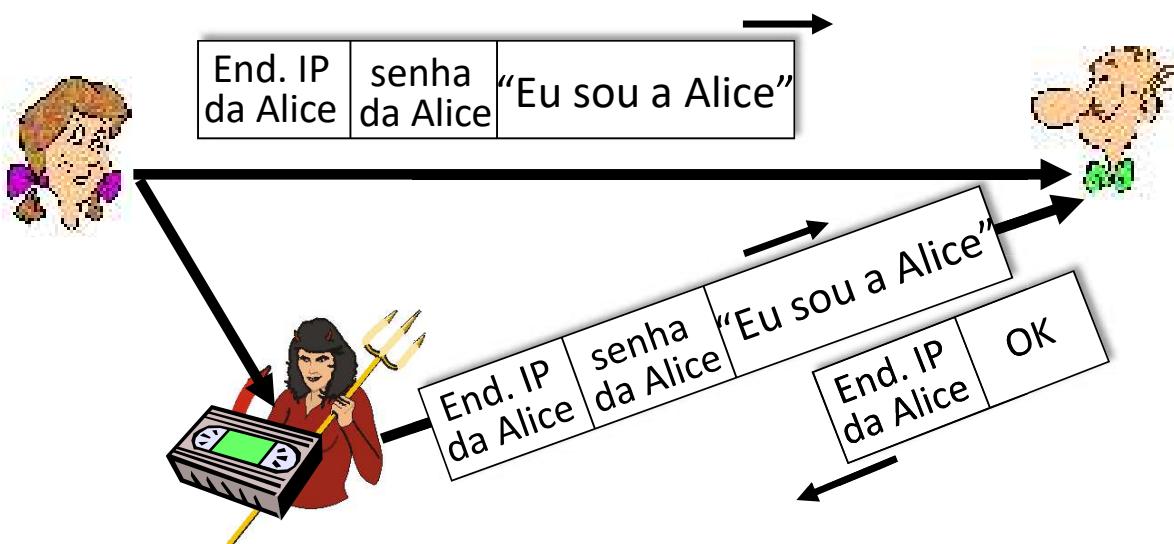
Protocolo ap3.0: Alice diz “eu sou a Alice” e envia sua senha secreta para “provar”.



Autenticação: uma terceira tentativa

Objetivo: Bob quer que Alice “prove” sua identidade para ele

Protocolo ap3.0: Alice diz “eu sou a Alice” e envia sua senha secreta para “provar”.

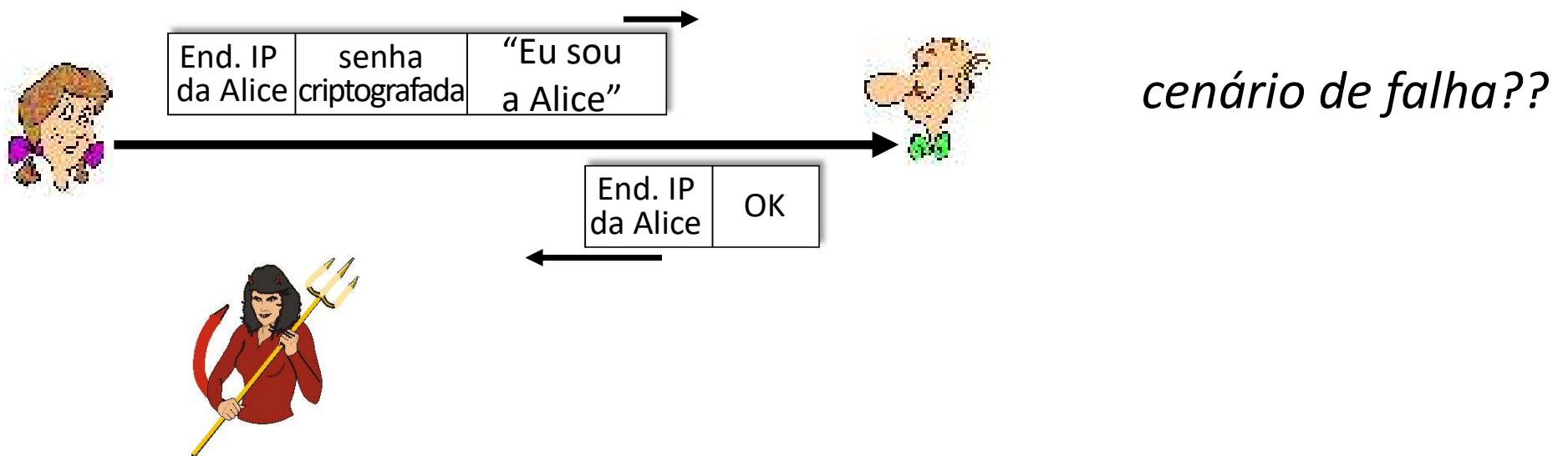


*ataque de
reprodução: Trudy
grava o pacote de
Alice e depois o
reproduz para Bob*

Autenticação: uma terceira tentativa modificada

Objetivo: Bob quer que Alice “prove” sua identidade para ele

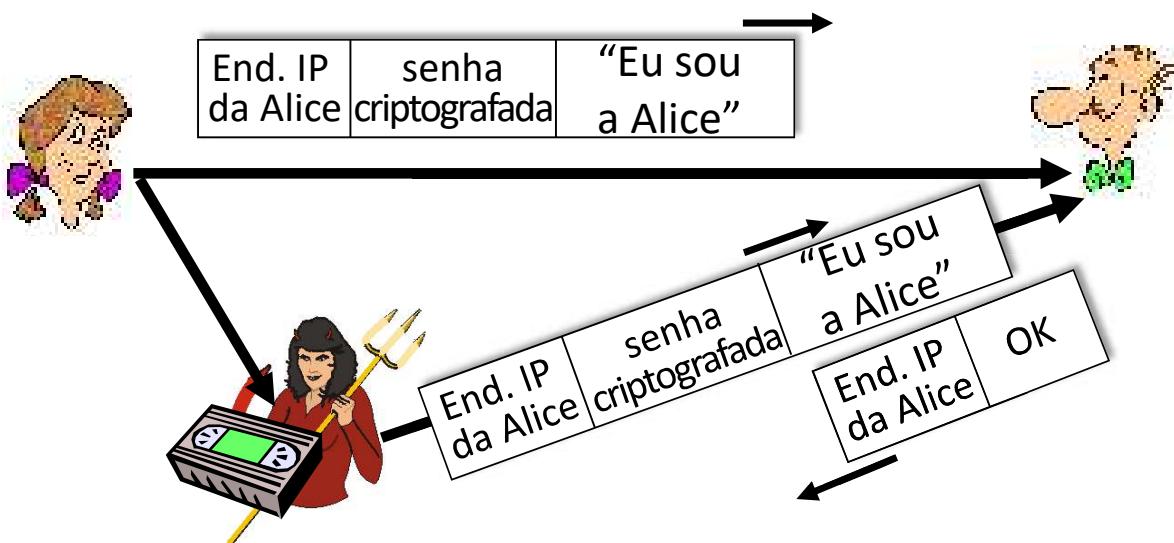
Protocolo ap3.0: Alice diz “Eu sou a Alice” e envia sua senha secreta criptografada para “provar”.



Autenticação: uma terceira tentativa modificada

Objetivo: Bob quer que Alice “prove” sua identidade para ele

Protocolo ap3.0: Alice diz “Eu sou a Alice” e envia sua senha secreta criptografada para “provar”.



*ataque de
reprodução ainda
funciona: Trudy grava
o pacote de Alice e
depois o reproduz
para Bob*

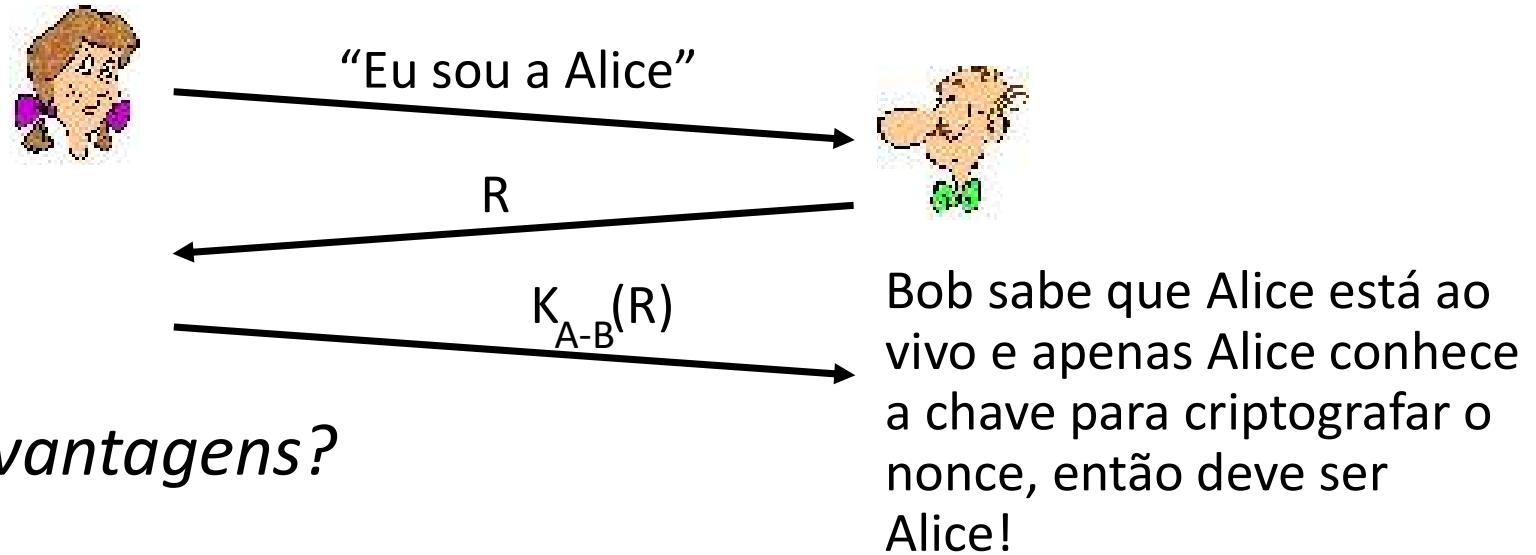
Autenticação: uma quarta tentativa

Objetivo: evitar ataque de reprodução

nonce: número (R) usado apenas **once-in-a-lifetime** (uma vez na vida)

protocolo ap4.0: para provar que Alice está “ao vivo”, Bob envia para ela um nonce R

- Alice deve retornar R, criptografado com chave secreta compartilhada

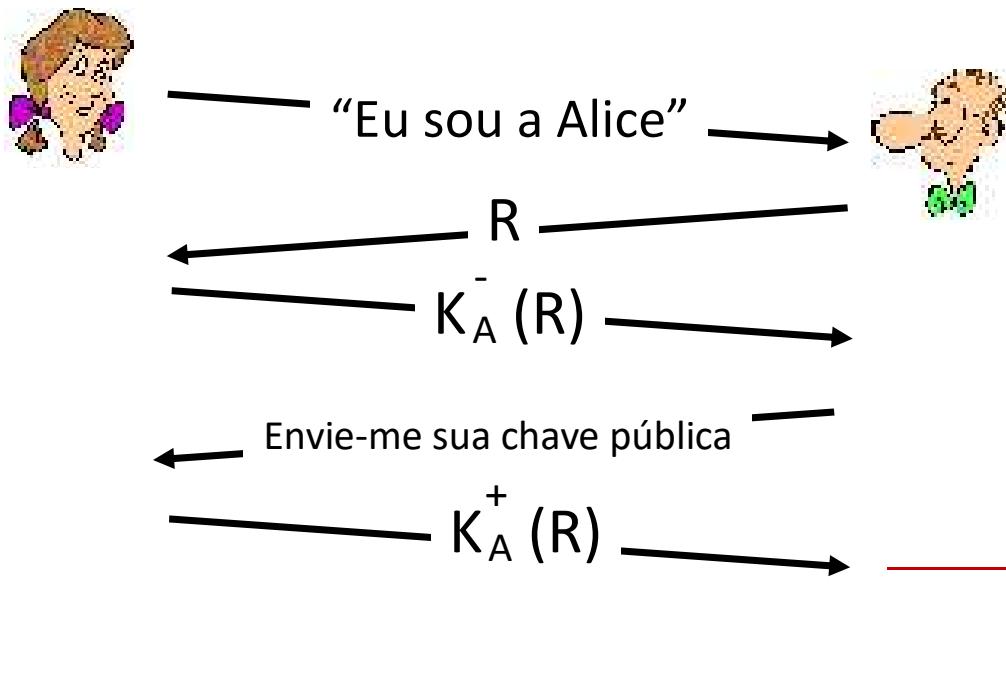


Falhas, desvantagens?

Autenticação: ap5.0

ap4.0 requer chave simétrica compartilhada - podemos autenticar usando técnicas de chave pública?

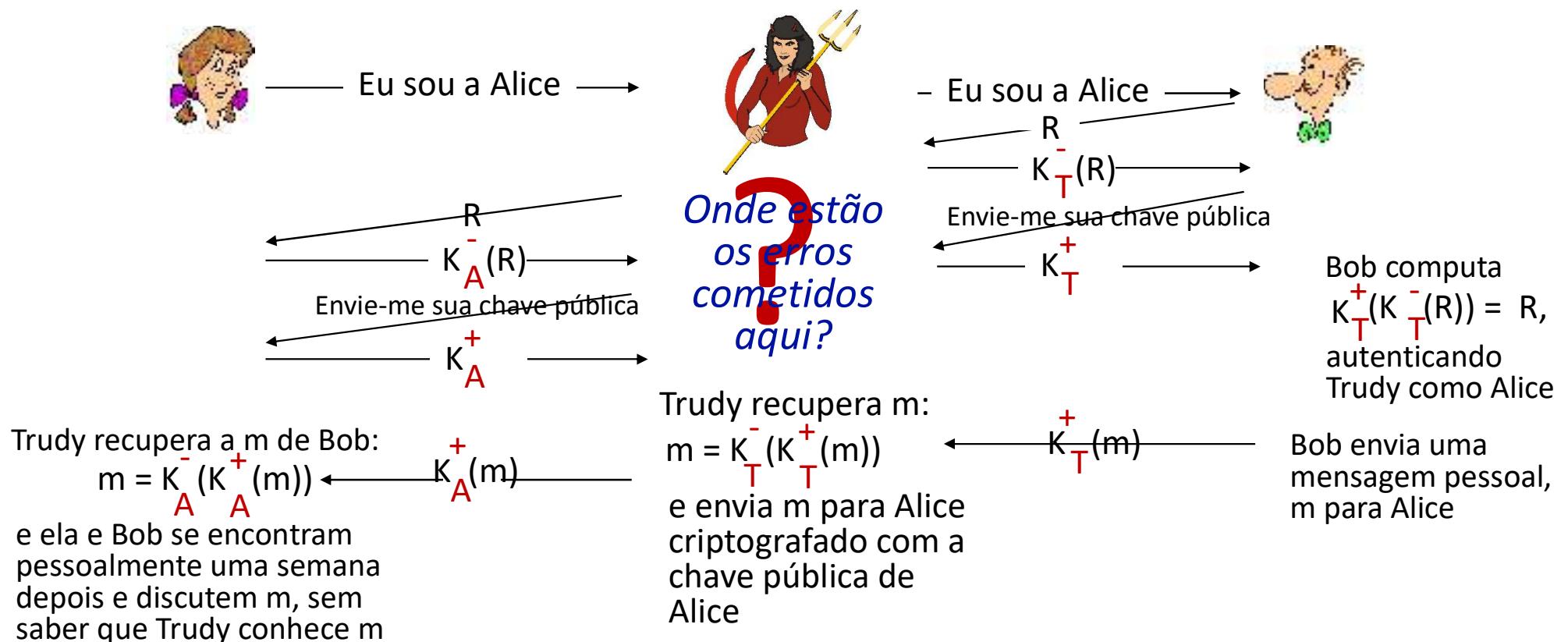
ap5.0: usa nonce e criptografia de chave pública



Bob computa
 $K_A^+ (K_A^- (R)) = R$
e sabe que apenas Alice
poderia ter a chave privada
que criptografou R de
forma que
 $K_A^+ (K_A^- (R)) = R$

Autenticação: ap5.0 – ainda há uma falha!

ataque do homem (ou mulher) no meio: Trudy se apresenta como Alice (para Bob) e como Bob (para Alice)



Roteiro do Capítulo 8

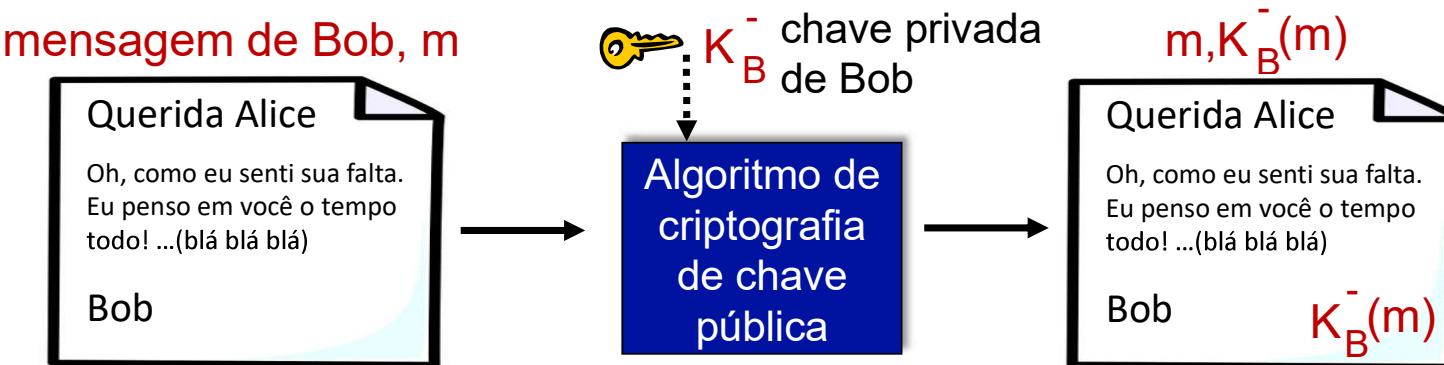
- O que é segurança de rede?
- Princípios de criptografia
- Autenticação, **integridade de mensagem**
- Protegendo o e-mail
- Protegendo conexões TCP: TLS



Assinaturas digitais

técnica criptográfica análoga às assinaturas manuscritas:

- remetente (Bob) assina digitalmente o documento: ele é o proprietário/criador do documento.
- *verificável, não falsificável*: destinatário (Alice) pode provar a alguém que Bob, e mais ninguém (incluindo Alice), deve ter assinado o documento
- **assinatura digital simples para mensagem m:**
 - Bob assina m criptografando com sua chave privada K_B^- , criando mensagem “assinada”, $K_B^-(m)$



Assinaturas digitais

- suponha que Alice receba a msg m , com assinatura: $m, K_B^-(m)$
- Alice verifica que m foi assinado por Bob aplicando a chave pública K_B^+ em $K_B^-(m)$ e então checando que $K_B^+(K_B^-(m)) = m$.
- Se $K_B^+(K_B^-(m)) = m$, quem assinou m deve ter usado a chave privada de Bob

Alice assim verifica que:

- Bob assinou m
- ninguém mais assinou m
- Bob assinou m e não m'

não repúdio:

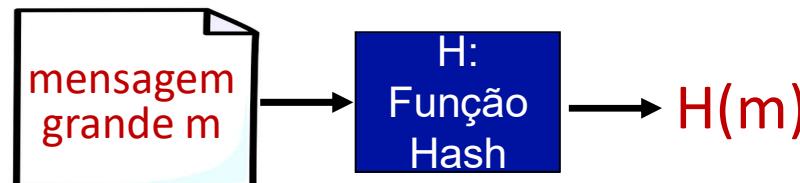
- ✓ Alice pode levar m e a assinatura $K_B^-(m)$ ao tribunal e provar que Bob assinou m

Resumos de mensagens

é computacionalmente custoso criptografar mensagens longas com chave pública

objetivo: “impressão digital” de comprimento fixo e fácil de calcular

- aplicar função hash H a m , obter resumo de mensagem de tamanho fixo, $H(m)$



Propriedades da função Hash :

- muitos-para-1
- produz resumo de mensagem de tamanho fixo (impressão digital)
- dado resumo da mensagem x , é computacionalmente inviável encontrar m tal que $x = H(m)$

Soma de verificação da Internet: função hash fraca para criptografia

A soma de verificação da Internet tem algumas propriedades de uma função hash:

- produz resumo de tamanho fixo (soma de 16 bits) da mensagem
- é muitos-para-um

mas dada mensagem com determinado valor de hash, é fácil encontrar outra mensagem com o mesmo valor de hash :

mensagem formato ASCII

I O U 1	49 4F 55 31
0 0 . 9	30 30 2E 39
9 B O B	39 42 D2 42

B2 C1 D2 AC

mensagem formato ASCII

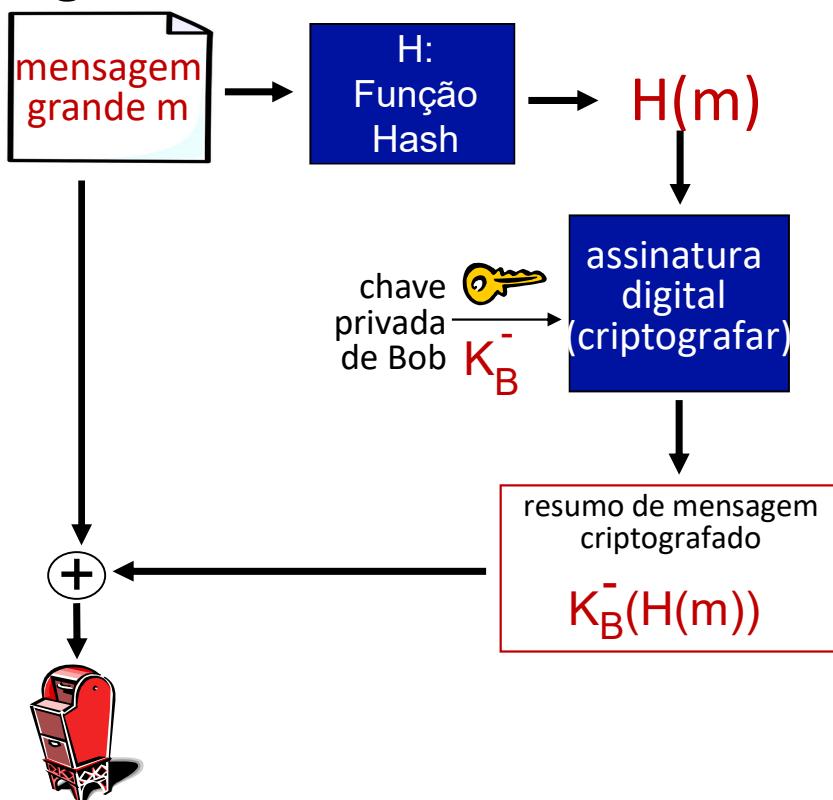
I O U 9	49 4F 55 <u>39</u>
0 0 . 1	30 30 2E <u>31</u>
9 B O B	39 42 D2 42

B2 C1 D2 AC

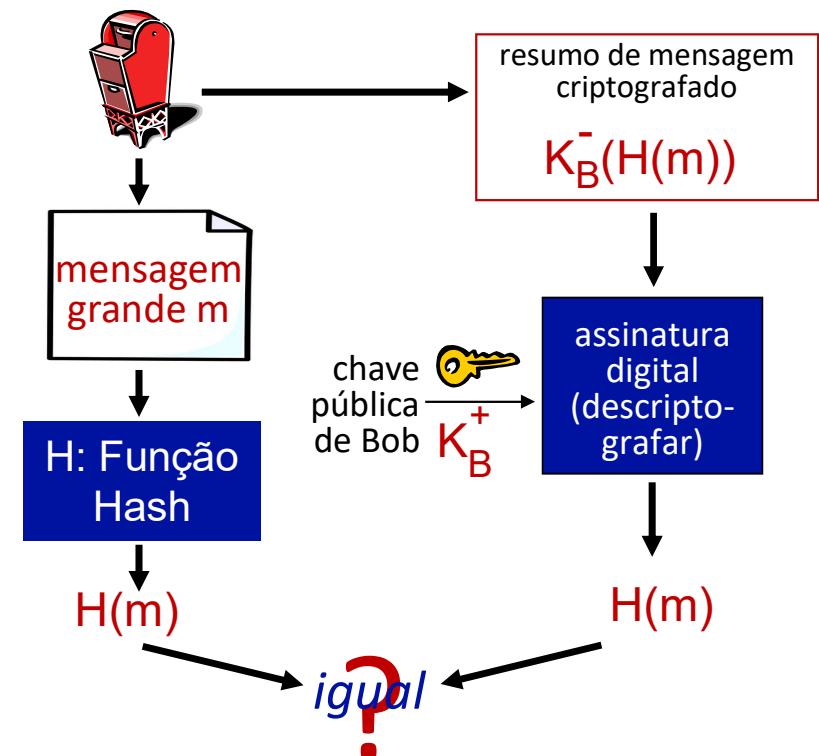
*mensagens diferentes, mas
somas de verificação idênticas!*

Assinatura digital = resumo da mensagem assinado

Bob envia mensagem assinada digitalmente:



Alice verifica a assinatura e a integridade da mensagem assinada digitalmente:



Algoritmos de função Hash

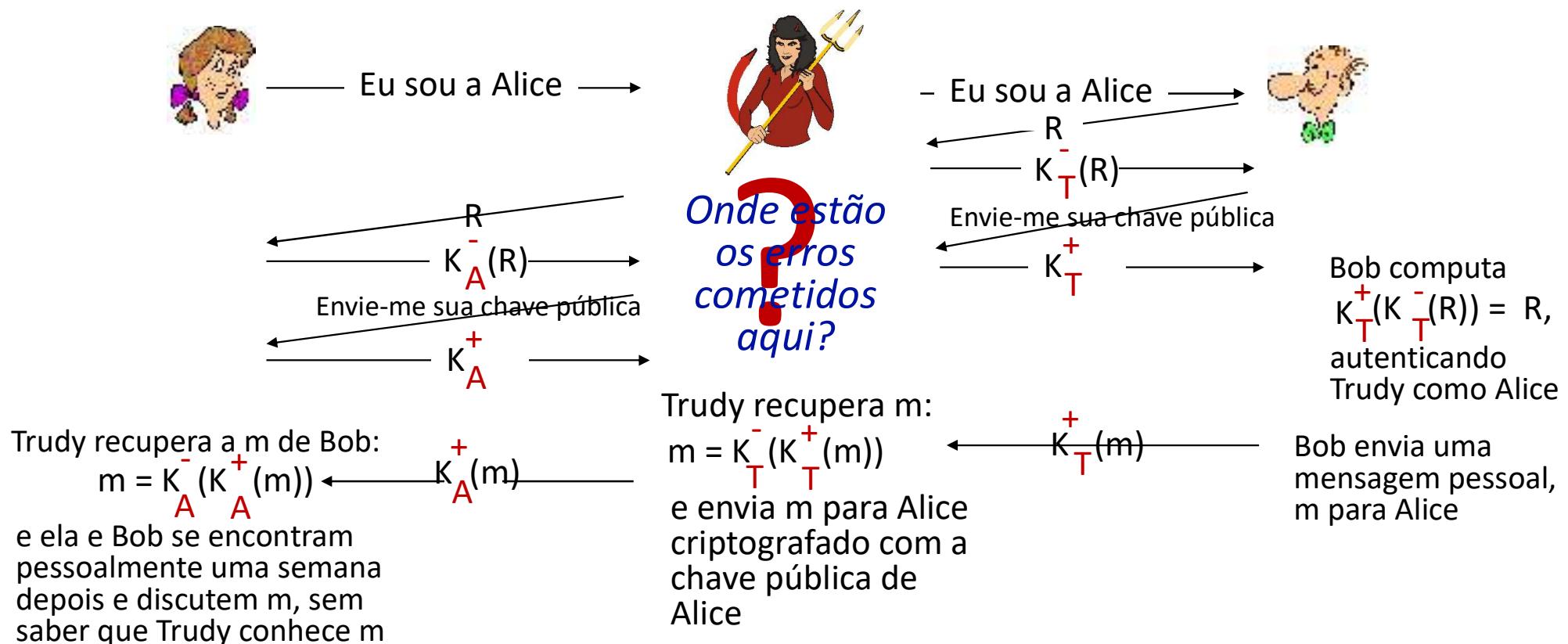
- A função de Hash MD5 é amplamente utilizada (RFC 1321)
 - calcula o resumo da mensagem de 128 bits em um processo de 4 etapas.
 - dada uma string arbitrária de 128 bits x , parece difícil de construir uma mensagem m cujo hash MD5 é igual a x
- SHA-1 também é usado
 - padrão dos EUA [NIST, FIPS PUB 180-1]
 - resumo de mensagem de 160 bits

Algoritmos de função Hash

- MD5 e SHA-1 não são mais considerados seguros.
 - Tem falhas que podem ser exploradas.
- SHA-2 e SHA-3 são utilizados atualmente.
 - SHA-2 desenvolvido pela NSA.
 - SHA-3 escolhido em uma competição organizada pelo NIST.
 - Hash de 224 à 512 bits.
 - SHA-3 tem versão que gera hash de tamanho arbitrário.

Autenticação: ap5.0 – vamos consertar!!

Lembre-se do problema: Trudy se apresenta como Alice (para Bob) e como Bob (para Alice)



Necessidade de chaves públicas certificadas

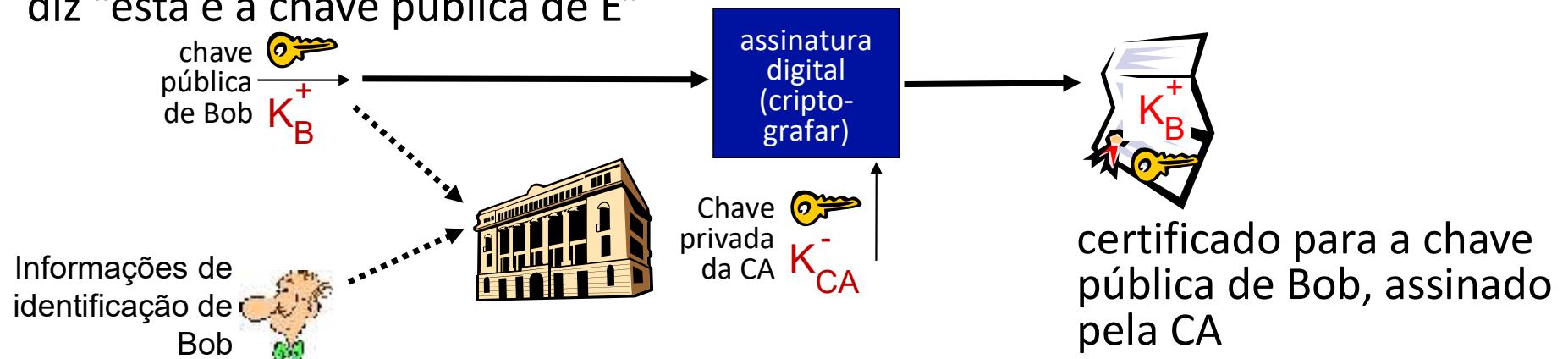
- motivação: Trudy aplica o trote da pizza em Bob

- Trudy cria um pedido por e-mail:
Prezada Pizzaria, Por favor, entregue-me quatro pizzas de pepperoni. Obrigado, Bob
- Trudy assina o pedido com sua chave privada
- Trudy envia o pedido para a Pizzaria
- Trudy envia sua chave pública para a Pizzaria, mas diz que é a chave pública de Bob
- Pizzaria verifica assinatura; então entrega quatro pizzas de pepperoni para Bob
- Bob nem gosta de pepperoni



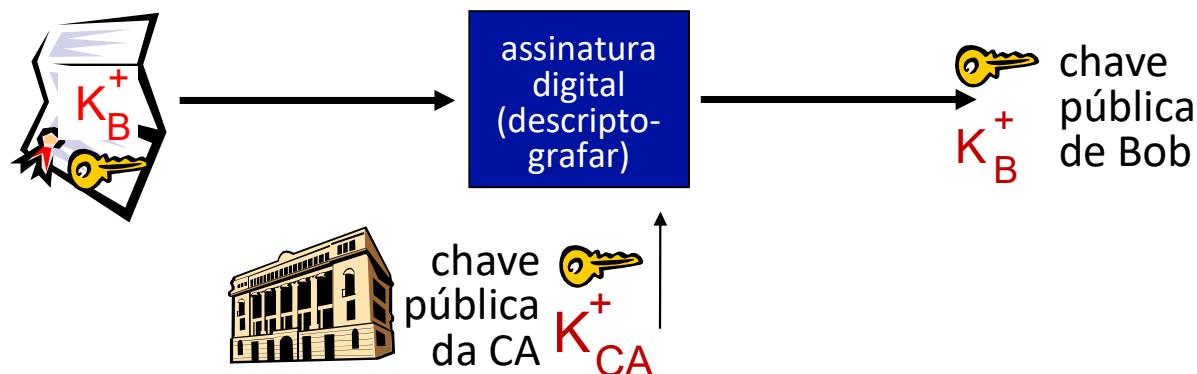
Autoridades de Certificação (CA - Certification Authorities) de chave pública

- **autoridade de certificação (CA):** vincula a chave pública a uma entidade específica, E
- entidade (pessoa, site, roteador) registra sua chave pública com uma CA e fornece “prova de identidade” a ela
 - A CA cria um certificado vinculando a identidade de E à sua chave pública
 - certificado contendo a chave pública de E é assinado digitalmente por CA: CA diz “esta é a chave pública de E”



Autoridades de Certificação (CA - Certification Authorities) de chave pública

- quando Alice quer a chave pública de Bob:
 - obtém o certificado de Bob (de Bob ou outro lugar)
 - aplica a chave pública da CA ao certificado de Bob, obtém a chave pública de Bob



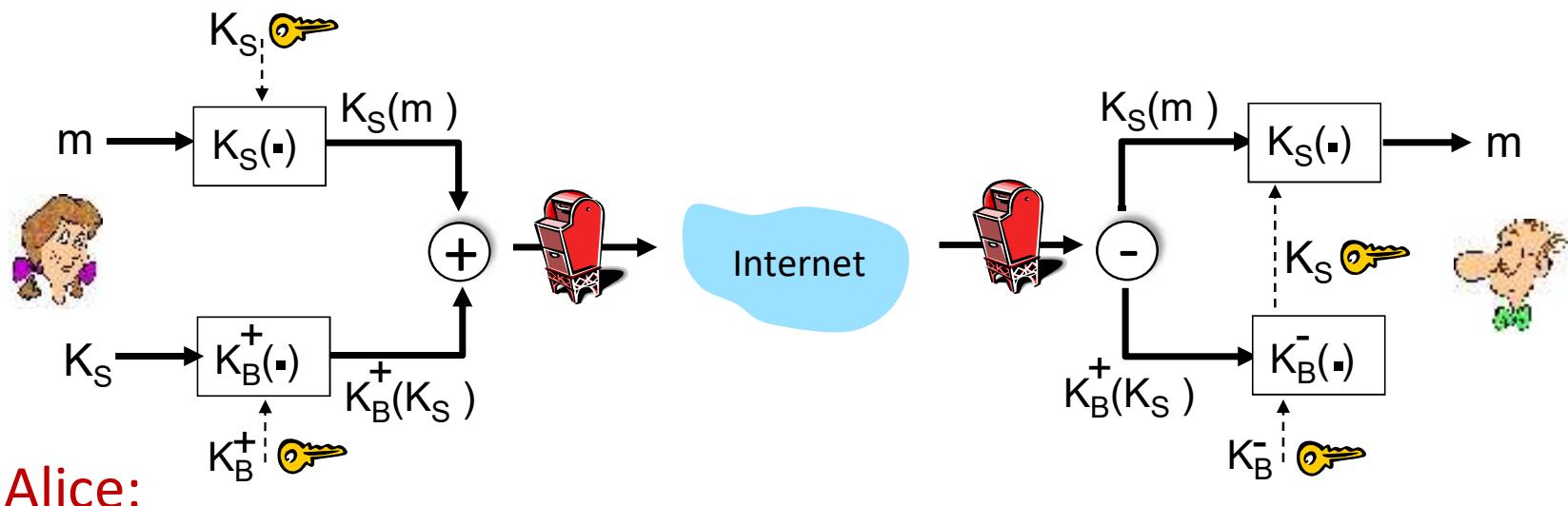
Roteiro do Capítulo 8

- O que é segurança de rede?
- Princípios de criptografia
- Autenticação, integridade de mensagem
- Protegendo o e-mail**
- Protegendo conexões TCP: TLS



E-mail seguro: confidencialidade

Alice quer enviar um e-mail *confidencial*, m , para o Bob.

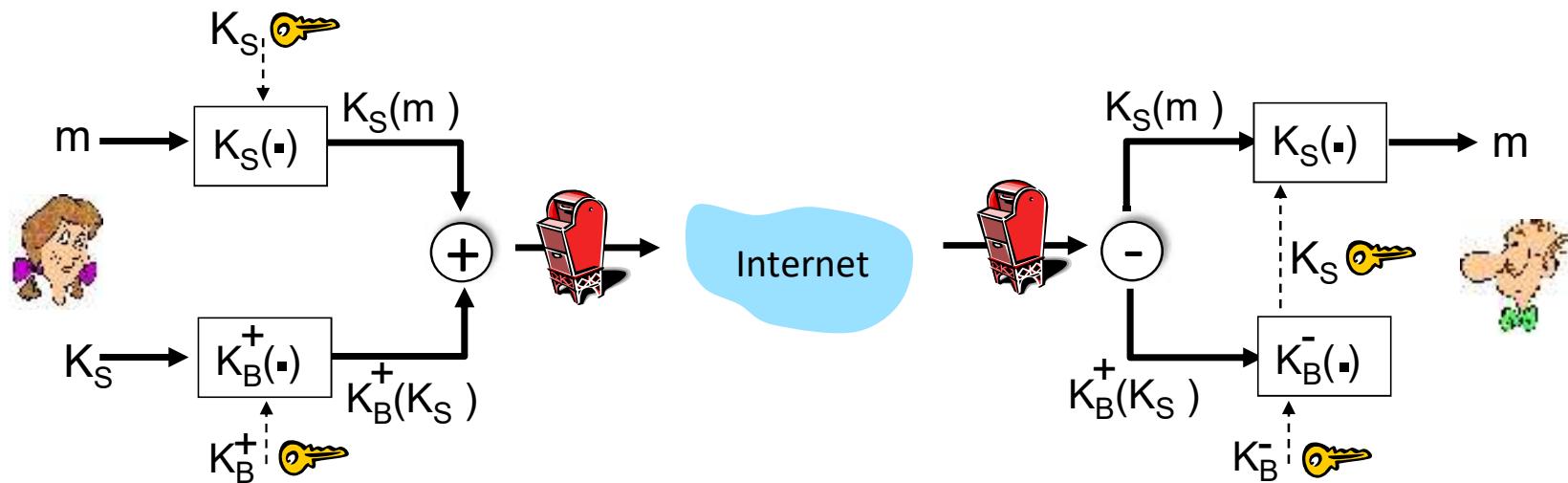


Alice:

- gera chave privada *simétrica* aleatória, K_S
- criptografa a mensagem com K_S (por eficiência)
- também criptografa K_S com a chave pública de Bob
- envia ambos $K_S(m)$ e $K_B^+(K_S)$ para o Bob

E-mail seguro: confidencialidade (mais)

Alice quer enviar um e-mail *confidencial*, m , para o Bob.

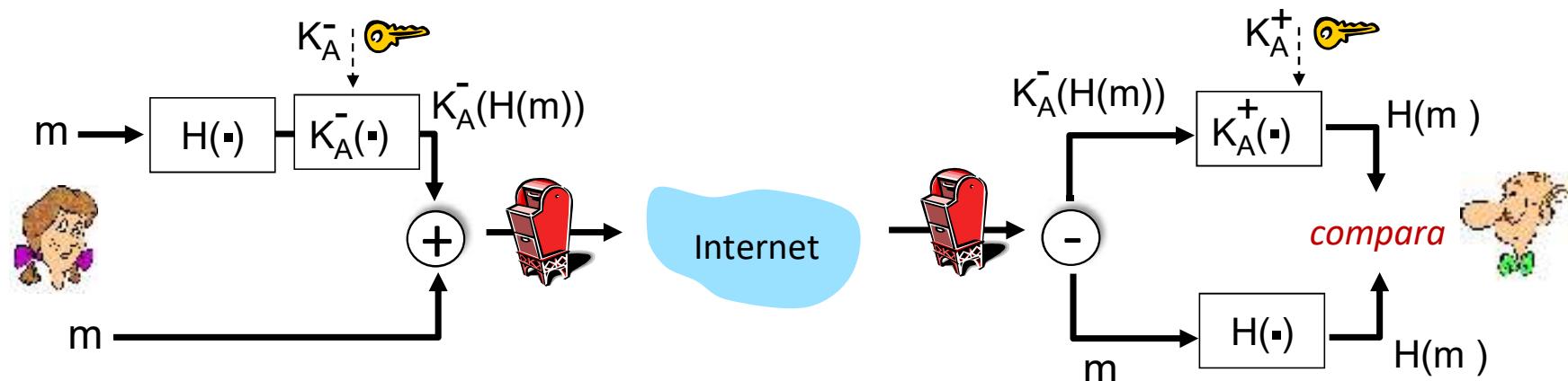


Bob:

- usa sua chave privada para descriptografar e recuperar K_S
- usa K_S para descriptografar $K_S(m)$ e recuperar m

E-mail seguro: integridade, autenticação

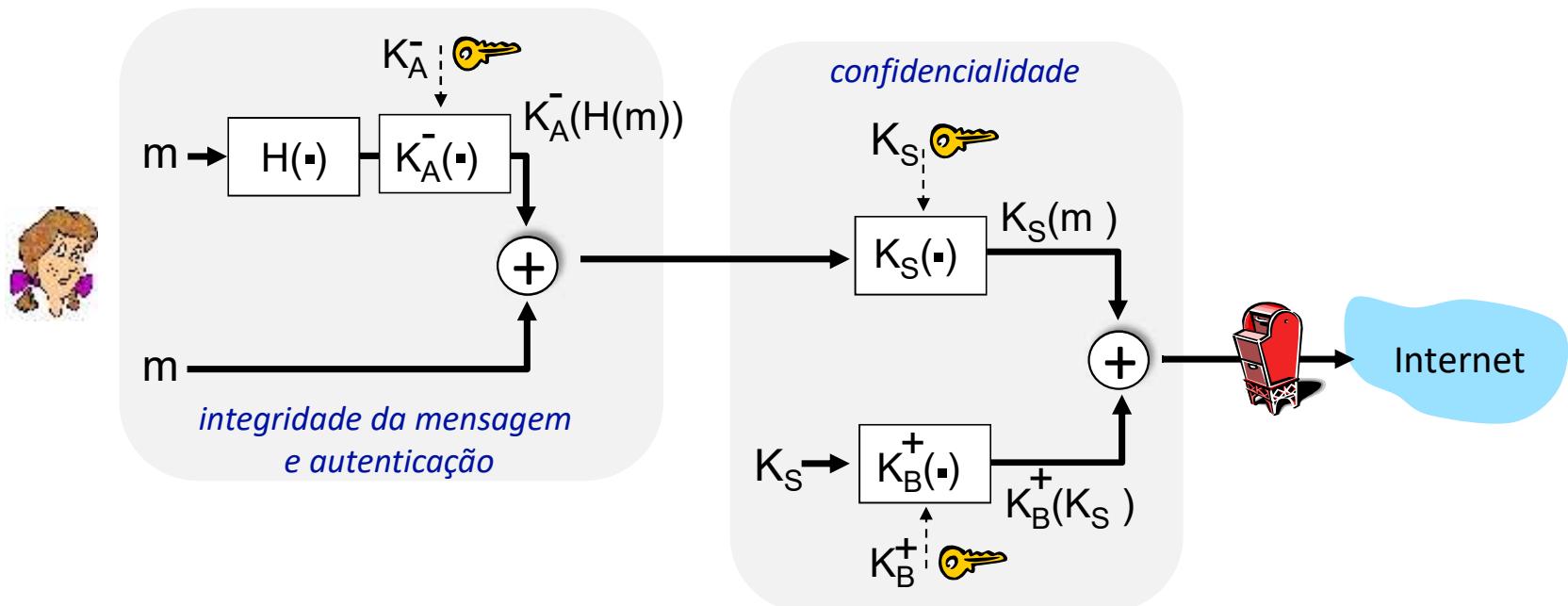
Alice quer enviar m para Bob, com *integridade de mensagem* e *autenticação*



- Alice assina digitalmente o hash de sua mensagem com sua chave privada, fornecendo integridade e autenticação
- envia ambas a mensagem (aberta) e a assinatura digital

E-mail seguro: integridade, autenticação

Alice envia m para Bob, com *confidencialidade, integridade da mensagem, e autenticação*



Alice usa três chaves: chave privada dela, chave pública de Bob, nova chave simétrica

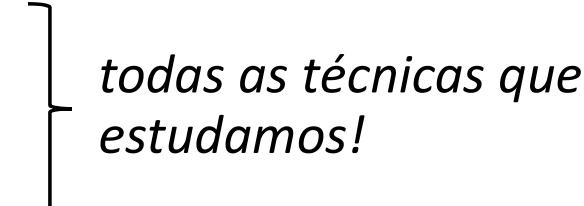
Quais são as ações complementares de Bob?

Roteiro do Capítulo 8

- O que é segurança de rede?
- Princípios de criptografia
- Autenticação, integridade de mensagem
- Protegendo o e-mail
- **Protegendo conexões TCP: TLS**



Transport-layer security (TLS)

- protocolo de segurança amplamente implantado acima da camada de transporte
 - suportado por quase todos os navegadores, servidores web: https (porta 443)
 - fornece:
 - **confidencialidade:** via *criptografia simétrica*
 - **integridade:** via *hash criptográfico*
 - **autenticação:** via *criptografia de chave pública*
 - história:
 - pesquisa e implementação inicial: programação de rede segura, soquetes seguros
 - secure socket layer (SSL) descontinuado [2015]
 - TLS 1.3: RFC 8846 [2018]
- 
- todas as técnicas que estudamos!*

Transport-layer security (TLS)

- SSL é predecessor do TLS.
 - SSL v3 tem vulnerabilidades e não é mais considerado seguro.
- TLS v1.3 (publicada em 2018) é a versão mais recente e mais usada atualmente.

Network

Client HTTP Version Used

HTTP/1.1	1.5M	
HTTP/2	421.49k	
HTTP/3	379.63k	
HTTP/1.0	7.63k	

Traffic Served Over SSL

none	188.07k	
TLSv1.3	1.89M	
TLSv1	2.55k	
TLSv1.1	144	
TLSv1.2	228.43k	

Top Content Types

html	1.15M	
jpeg	286.07k	
empty	244.63k	
js	168.82k	
css	141.51k	

Transport-layer security: o que é necessário?

- vamos *construir* um protocolo TLS de brinquedo, t-tls, para ver o que é necessário!
- já vimos as "peças" :
 - **handshake**: Alice e Bob usam seus certificados e chaves privadas para autenticar um ao outro e trocar ou criar segredo compartilhado
 - **derivação de chaves**: Alice e Bob usam segredo compartilhado para derivar conjunto de chaves
 - **transferência de dados**: transferência de fluxo de dados: dados como uma série de registros
 - não apenas transações únicas
 - **fechamento de conexão**: mensagens especiais para fechar a conexão com segurança

t-tls: handshake inicial



fase do handshake do t-tls:

- Bob estabelece conexão TCP com Alice
- Bob verifica se Alice é realmente Alice
- Bob envia a Alice uma chave mestra secreta (MS), usada para gerar todas as outras chaves para a sessão TLS
- problemas potenciais:
 - 3 RTT antes que o cliente possa começar a receber dados (incluindo o handshake do TCP)

t-tls: chaves criptográficas

- é considerado ruim usar a mesma chave para mais de uma função criptográfica
 - usa-se chaves diferentes para código hash de autenticação de mensagem (HMAC – *hashed message authentication code*) e criptografia
- quatro chaves:
 - 🔑 K_c : chave de criptografia para dados enviados do cliente para o servidor
 - 🔑 M_c : chave HMAC para dados enviados do cliente para o servidor
 - 🔑 K_s : chave de criptografia para dados enviados do servidor para o cliente
 - 🔑 M_s : chave HMAC para dados enviados do servidor para o cliente
- chaves derivadas da função de derivação de chave (KDF - *key derivation function*)
 - pega o segredo mestre e (possivelmente) alguns dados aleatórios adicionais para criar novas chaves

t-tls: criptografando dados

- lembre-se: TCP fornece abstração de *fluxo de bytes* de dados
- Q: podemos criptografar dados *no fluxo* conforme gravados no soquete TCP?
 - R: para onde iria o HMAC? Se estiver no final, nenhuma verificação da integridade da mensagem seria realizada até que todos os dados sejam recebidos e a conexão seja fechada!
 - solução: quebrar fluxo em série de “registros”
 - cada registro cliente-servidor carrega um HMAC, criado usando M_c
 - receptor pode agir em cada registro que chega
- registro t-tls criptografado usando chave simétrica, K_c , e passado para o TCP:

$$K_c \left(\begin{array}{|c|c|c|} \hline tamanho & dados & HMAC \\ \hline \end{array} \right)$$

t-tls: criptografando dados (mais)

- possíveis ataques no fluxo de dados?
 - *reordenação*: intermediário intercepta segmentos TCP e reordena (manipula números de sequência no cabeçalho TCP não criptografado)
 - *reprodução*
- soluções:
 - usar números de sequência TLS (dados e número de sequência TLS incorporados ao HMAC)
 - usar nonce

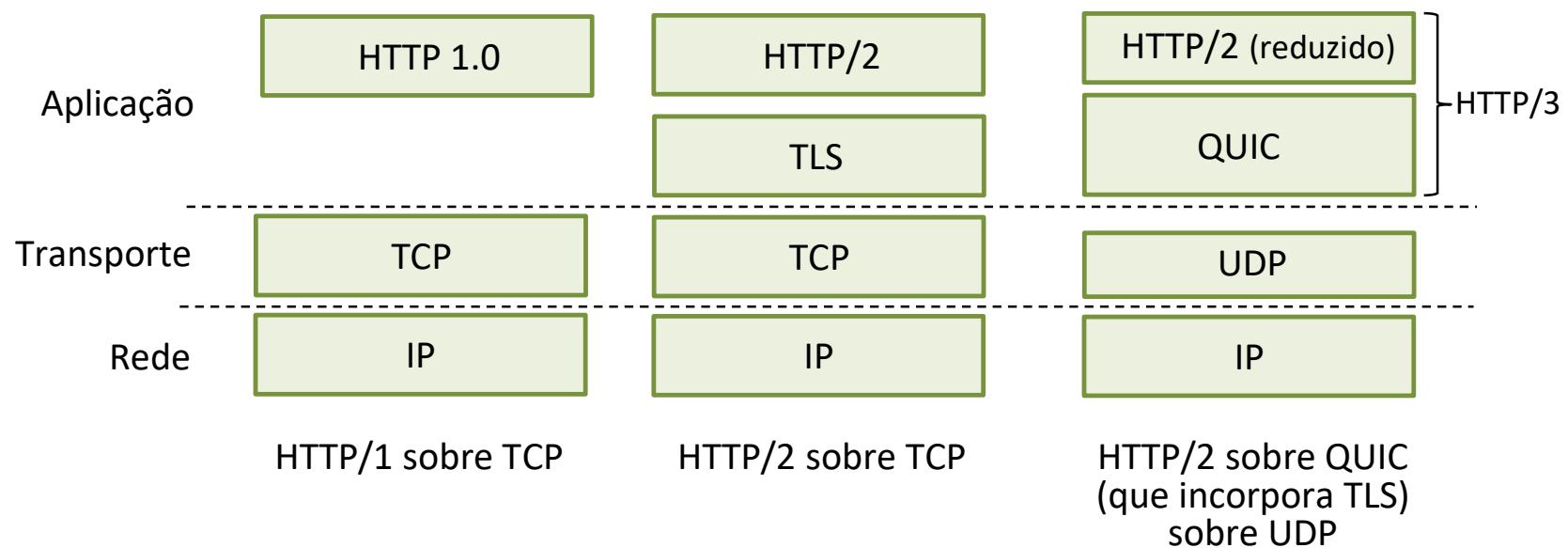
t-tls: fechamento de conexão

- ataque de truncamento:
 - atacante forja segmento de fechamento de conexão TCP
 - um ou ambos os lados pensam que há menos dados do que realmente há
- solução: tipos de registro, com um tipo para fechamento
 - tipo 0 para dados; tipo 1 para fechar
- HMAC agora computado usando dados, tipo, e número de sequência

$$K_C \left(\begin{array}{|c|c|c|c|} \hline tamanho & tipo & dados & HMAC \\ \hline \end{array} \right)$$

Transport-layer security (TLS)

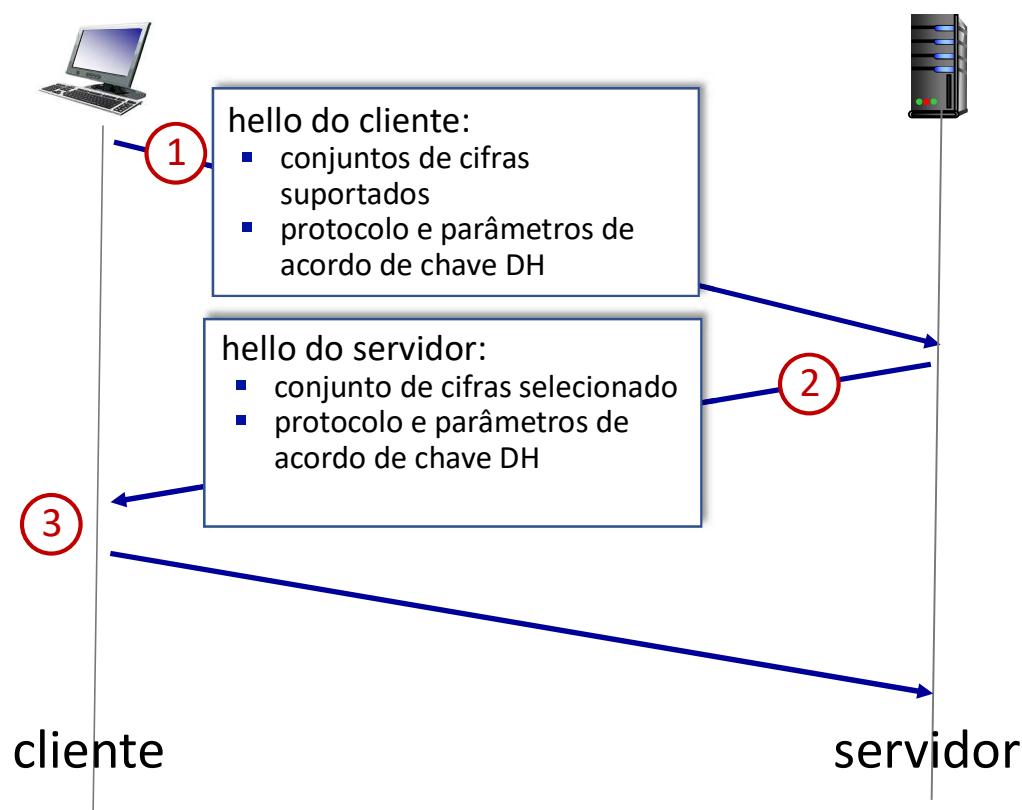
- O TLS fornece uma API que *qualquer* aplicativo pode usar
- uma visualização do HTTP usando TLS:



TLS: conjunto de cifras da versão 1.3

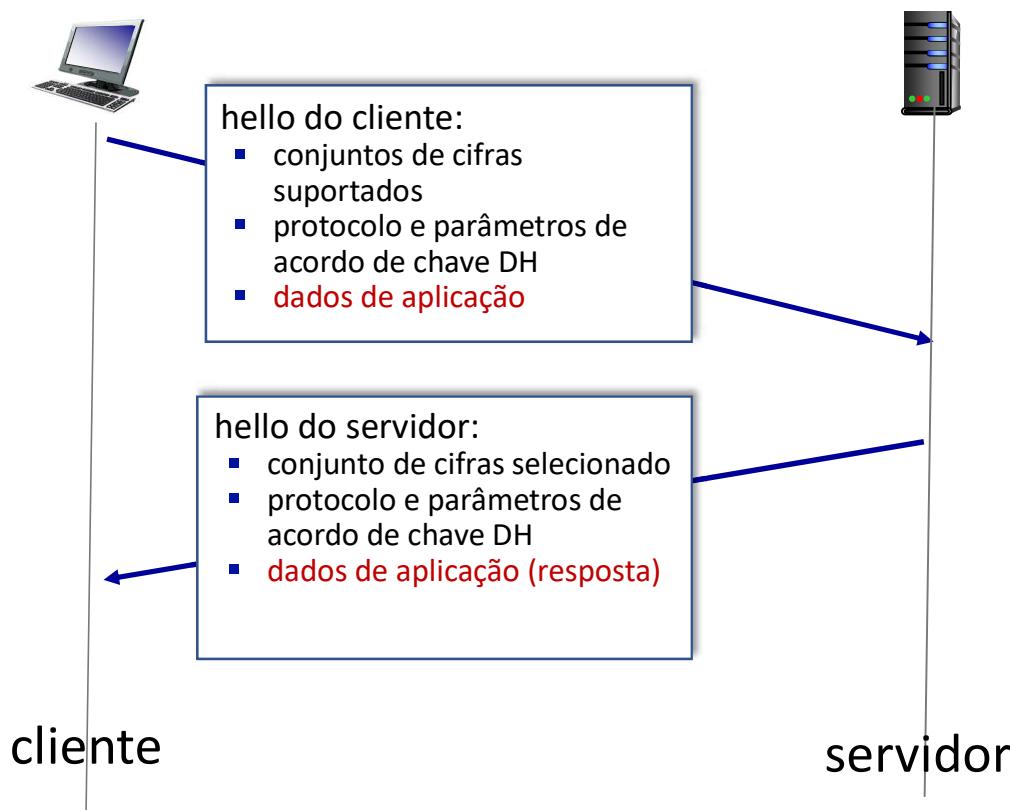
- “suíte de cifras”: algoritmos que podem ser usados para geração de chaves, criptografia, HMAC, e assinatura digital
- TLS 1.3 (2018): escolha de conjunto de cifras mais limitada do que no TLS 1.2 (2008)
 - apenas 5 opções, em vez de 37 opções
 - 4 das opções são baseadas no AES
 - *requer* Diffie-Hellman (DH) para troca de chaves, em vez de DH ou RSA
 - algoritmo combinado de criptografia e autenticação (“criptografia autenticada”) para dados em vez de criptografia e autenticação em série
 - HMAC usa função de criptografia de hash SHA (256 ou 284)

TLS 1.3 handshake: 1 RTT



- ① mensagem de saudação TLS do cliente:
 - protocolo de acordo de chave, parâmetros
 - indica conjuntos de cifras que ele suporta
- ② mensagem de saudação TLS do servidor escolhe:
 - protocolo e parâmetros de acordo de chave
 - conjunto de cifras
 - certificado assinado do servidor
- ③ cliente:
 - verifica o certificado do servidor
 - gera chave
 - agora pode fazer solicitação de aplicação (por exemplo, HTTPS GET)

TLS 1.3 handshake: 0 RTT



- a mensagem de saudação inicial contém dados criptografados de aplicação!
 - “retomando” (*resume*) conexão anterior entre cliente e servidor
 - dados de aplicação criptografados usando “segredo mestre de retomada” (“*resumption master secret*”) da conexão anterior
- vulnerável a ataques de repetição!
 - talvez OK para HTTP GET ou solicitações de clientes que não modificam o estado do servidor