SISTEMAS DISTRIBUÍDOS I

Capítulo 9

Segurança em sistemas distribuídos

NOTA PRÉVIA

A estrutura da apresentação é semelhante e utiliza algumas das figuras livro de base do curso

G. Coulouris, J. Dollimore and T. Kindberg,
Distributed Systems - Concepts and Design,
Addison-Wesley, 5th Edition, 2011

Capítulo 11

SEGURANÇA EM SISTEMAS DISTRIBUÍDOS

Este capítulo apresenta uma introdução às técnicas de segurança em sistemas distribuídos, centrada na problemática da segurança das comunicações e da autenticação.

Organização do capítulo

Aula 1

Apresentação do problema

- Diferenças essenciais entre os sistemas centralizados e os sistemas distribuídos
- Canais seguros e seus requisitos

Aula2

Criptografia simétrica

- Utilização da criptografia simétrica na comunicação e autenticação

Criptografia assimétrica

- Utilização da criptografia assimétrica na comunicação e autenticação

Assinaturas digitais

Aula 3

Aplicações seguras mais comuns: OAuth, SSL, PGP, ...

DEPENDABLE SYSTEMS...

Dependable systems é a área da informática que estuda os problemas e as soluções inerentes à realização de sistemas **seguros** e **fiáveis** – "de confiança".

Um sistema é "**do qual se pode depender**" deverá exibir diversas propriedades:

- disponibilidade, fiabilidade perante falhas
- **seguro perante ataques**

DEPENDABLE SYSTEMS…

Dependable systems é a área da informática que estuda os problemas e as soluções inerentes à realização de sistemas **seguros** e **fiáveis** – "de confiança".

Um sistema é "**do qual se pode depender**" deverá exibir diversas propriedades:

- disponibilidade, fiabilidade perante falhas -> redundância, replicação
- **seguro perante ataques -> ?**

DEPENDABLE SYSTEMS...

Dependable systems é a área da informática que estuda os problemas e as soluções inerentes à realização de sistemas **seguros** e **fiáveis** – "de confiança".

Um sistema é "**do qual se pode depender**" deverá exibir diversas propriedades:

- disponibilidade, fiabilidade perante falhas -> redundância, replicação
- **seguro perante ataques -> ?**

Políticas (restrições) de segurança, suportadas por **mecanismos** de segurança,

implementadas sobre uma **base de confiança estabelecida** à partida

DA NECESSIDADE DA SEGURANÇA

O **valor** inerente aos **dados** e às **operações** de um sistema fazem deles alvos de ataques

A proteção dos dados e das operações requer medidas específicas:

Mecanismos de segurança fornecem proteção

Políticas de segurança definem como os mecanismos são usados afim de impor restrições aos acessos (indevidos)

Segurança em Sistemas Distribuídos

Sistemas distribuídos são susceptíveis a novos ataques...

A separação física dos componentes (e a necessidade de comunicarem entre si, através de uma rede) introduz **vetores de ataque** adicionais.

É necessário uma reavaliação da base de confiança -> **trusted computing base.**

Para implementar políticas de segurança equivalentes, são necessários mecanismos de segurança mais sofisticados face a um sistema isolado

MODELO DE SEGURANÇA

Num sistema existem entidades que do ponto de vista da segurança têm identidade própria, direitos e deveres - essas entidades podem ser utilizadores, componentes, processos, etc. e designam-se pelo termo *principal*

A segurança do sistema distribuído passa por:

- autenticar os *principais* (*autenticação*)
- verificar os seus direitos de acesso aos objetos (*controlo de acessos*)

- da distribuição advém a necessidade de utilizar *canais seguros* para impedir o acesso, alteração ou destruição indevida de informação, incluindo, *proteger a privacidade*

Para fornecer segurança é necessário estabelecer que alguns componentes do sistema são seguros (*trusted computing base*) – caso contrário é impossível

Por exemplo, num sistema centralizado é normal assumir que o *kernel* do SO é seguro

ELEMENTOS DO MODELO DE SEGURANÇA

Principal - uma entidade (pessoa, processo, servidor, cliente, …) que é singular do ponto de vista dos direitos no sistema.

Autenticação – processo de verificar que um principal tem a identidade que diz ter – em geral, deve ser capaz de o provar.

Geralmente utiliza-se um método lógico do tipo segredo partilhado entre P e quem o autentica (de que uma palavra chave é o exemplo mais conhecido), mas também se pode basear na verificação de atributos físicos (identificação da voz, impressões digitais ou da retina por exemplo) ou na posse de algo que só P pode possuir (um cartão magnético por exemplo).

Controlo de acessos - dada uma operação *Op* sobre um objecto *O*, é necessário decidir se o principal *P* pode aplicar *Op* a *O*.

Normalmente utilizam-se ACLs (Access Control Lists) ou Capacidades (Tickets).

CAMADAS DE SEGURANÇA

A segurança de um sistema pode ser endereçada por camadas

O papel de cada camada é **estabelecer uma linha de defesa** e **alargar a *trusted computing base*** para as camadas superiores

A *trusted computing base* **inicial** deve ser tão **minimalista** quanto o possível

uma base de confiança reduzida a priori tem um uma superfície de ataque mais reduzida

VETORES DE ATAQUE USUAIS

Adulterar a *trusted computing base*

Explorar insuficiências na *trusted computing base*

Falhas na especificação ou implementação da TCB podem dar ao atacante direitos ou identidades indevidas

Violar os mecanismos de autenticação

Obter segredos indevidos

VETORES DE ATAQUE USUAIS

Os ataques são muito diversos, podem ser muito engenhosos...

Exploram *bugs*, insuficiências das implementações ou de planeamento ou simples fraquezas humanas...

Na prática, é impossível de enunciar todas as formas de ataque e quase sempre estão-se a descobrir novas

A tarefa de manter os sistemas seguros é um esforço contínuo que envolve todos

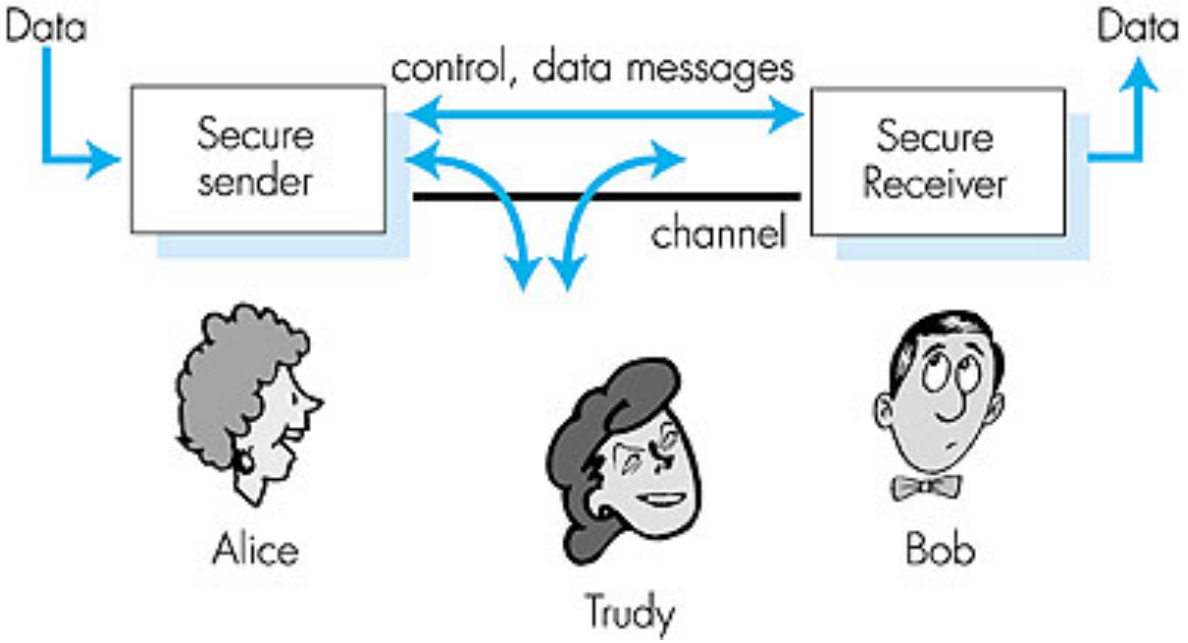
Há programas de recompensam a descoberta de problemas de segurança em produtos de grande consumo

Bug Bounty Hunters: Google Vulnerability Program

Rewards for qualifying bugs range from $100 to $20,000. The following table outlines the usual rewards chosen for the most common classes of bugs:

Category	Examples	Applications that permit taking over a Google account [1]	Other highly sensitive applications [2]	Normal Google applications	Non-integrated acquisitions and other sandboxed or lower priority applications [3]
Vulnerabilities giving direct access to Google servers					
Remote code execution	*Command injection, deserialization bugs, sandbox escapes*	$20,000	$20,000	$20,000	$1,337 - $5,000
Unrestricted file system or database access	*Unsandboxed XXE, SQL injection*	$10,000	$10,000	$10,000	$1,337 - $5,000
Logic flaw bugs leaking or bypassing significant security controls	*Direct object reference, remote user impersonation*	$10,000	$7,500	$5,000	$500
Vulnerabilities giving access to client or authenticated session of the logged-in victim					
Execute code on the client	Web: *Cross-site scripting* Mobile: *Code execution*	$7,500	$5,000	$3,133.7	$100
Other valid security vulnerabilities	Web: *CSRF, Clickjacking* Mobile: *Information leak, privilege escalation*	$500 - $7,500	$500 - $5,000	$500 - $3,133.7	$100

COMUNICAÇÃO NUM SISTEMA SEM SEGURANÇA



Canal está acessível ao atacante

Nomes usados na descrição dos protocolos de segurança:

Alice, **Bob**, **Carol**, **Dave** – participantes que querem comunicar
Eve é usado para um atacante que lê mensagens – *eavesdropper*)
Mallory/**Trudy** – atacante que pode ler, interceptar, modificar, suprimir ou re-introduzir mensagens nos canais ou tentar passar por um dos participantes
Sara – um servidor.

ATAQUES EM SISTEMAS DISTRIBUÍDOS ATRAVÉS DA COMUNICAÇÃO

Indiscrição – obtenção de mensagens sem autorização (*Eavesdropping*)

Mascarar-se ou **pretender** ser outro (*Masquerading*)

Reemissão de mensagens prévias (*Message replaying*)

Adulteração do conteúdo das mensagens (*Message tampering*)

Supressão de mensagens (*Message supression*)

Vandalismo por impedimento de prestação de serviço (*Denial of service attacks*)

Repúdio de mensagens, negar a autoria de mensagens

Análise de tráfego (*Traffic analysis*) procurando padrões que possam indiciar a natureza da comunicação, dos protocolos, etc.

EAVESDROPPING (EXEMPLO): CÓPIA DOS PACOTES QUE TRANSITAM NA REDE

Packet sniffing:

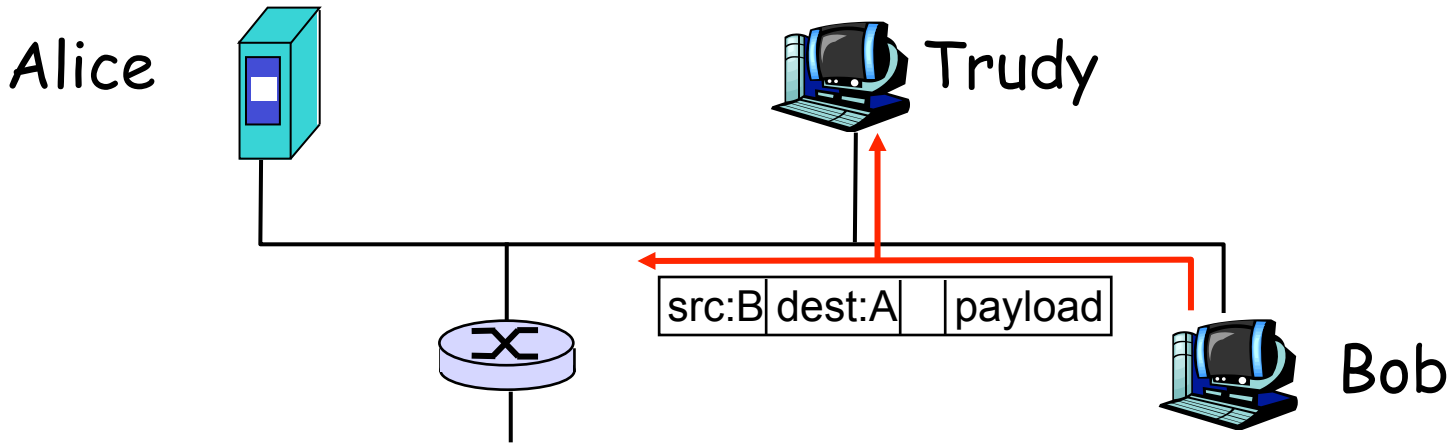
broadcast *media*

modo *promiscuous* da placa ***wireless*** ou de rede permite capturar todos os pacotes

permite a leitura de dados não cifrados

(exemplo: *passwords ftp, telnet, mas não ssh*)

exemplo: *Trudy sniffs Bob's packets*



IMPEDIMENTO DE PRESTAÇÃO DE SERVIÇO (EXEMPLO)

Denial of service (DoS):

um conjunto de pacotes inundam o receptor impedindo-o de fazer qualquer trabalho útil (DoS - Denial Of Service attack)

Distributed DoS (DDoS): ataque distribuído e coordenado de impedimento da prestação de serviço

exemplo, uma botnet (conjunto de *hosts* comprometidos) realizam um ataque coordenado SYN-attack a um website.

"Did Hackers Take Down NASDAQ?"

"CyberBunker Launches "World's Largest" DDoS Attack, Slows Down The Entire Internet"

"Bitcoin Under Attack? Dwolla & Mt. Gox Both Hit With DDoS Attacks Overnight"

DEFESA

Mecanismos de segurança mais comuns

Recurso à criptografia para:

obter **canais seguros**, imunes à repetição e violação da integridade dos dados e garantir confidencialidade

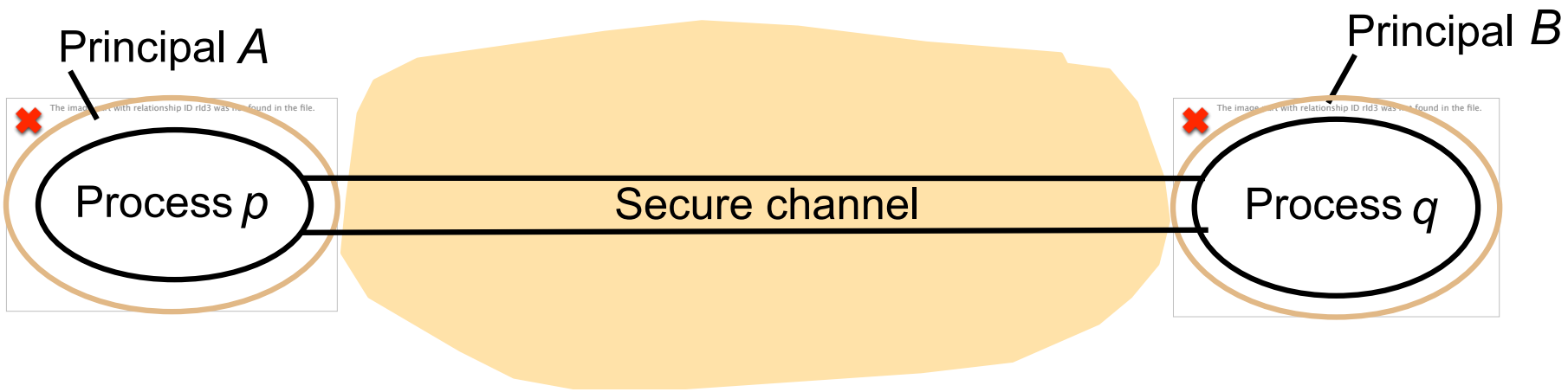
autenticar os principais (utilizadores, servidores, etc)

certificar conteúdos para garantir a sua autenticidade e não repúdio

Evitar/Esconder padrões regulares na comunicação entre principais

(por via de mensagens falsas/inúteis, aleatoriedade)

CANAIS SEGUROS



Objectivo: trocar dados com **confidencialidade**, **integridade** e **autenticidade**

Num canal seguro os interlocutores (A e B) estão autenticados

O atacante **não pode** ler/copiar, alterar ou introduzir mensagens
O atacante **não pode** fazer *replaying* de mensagens (*replaying* = reenvio)
O atacante **não pode** reordenar as mensagens

CANAIS SEGUROS: EM CONCRETO...

SSL/TLS Transport Layer Security / Secure Socket Layer

"The primary **goal** of the TLS protocol is to provide **privacy** and data **integrity** between two communicating computer applications.[1]:3 When secured by TLS, connections between a client (e.g., a web browser) and a server ..."

HTTPS - HTTP over SSL or HTTP Secure

... HTTPS consists of communication over HTTP within a connection encrypted by Transport Layer Security or its predecessor, Secure Sockets Layer. The main motivation for HTTPS is **authentication of the visited website** and protection of the **privacy** and **integrity** of the exchanged data.

--from wikipedia.

AUTENTICAÇÃO/AUTORIZAÇÃO : OAUTH

OAuth um protocolo standard e aberto para a autorização de acesso a recursos partilhados (tipicamente, na web)

Permite delegar a terceiros o acesso a recursos sensíveis, sem necessidade de lhes expor credenciais (*passwords*).

Exemplo:

Fotografias - recursos de um utilizador (o dono) armazenadas num servidor (base da confiança) podem ser acedidas por uma aplicação móvel (o cliente) ou por um serviço de impressão, sem que estes tenham acesso às credenciais geridas pelo servidor.

AGENDA: COMO IMPLEMENTAR...

Canais seguros

- Transmitir informação cifrada
- Autenticar parceiros de comunicação

Autorizar acesso restrito a dados

- Autenticar parceiros
- Criar e trocar testemunhos não forjáveis

CRIPTOGRAFIA

Criptografia é a disciplina que inclui os princípios, meios e métodos de transformação dos dados com a finalidade de:

esconder o seu conteúdo semântico
estabelecer a sua autenticidade
impedir que a sua alteração passe despercebida
evitar o seu repúdio

CRIPTOGRAFIA

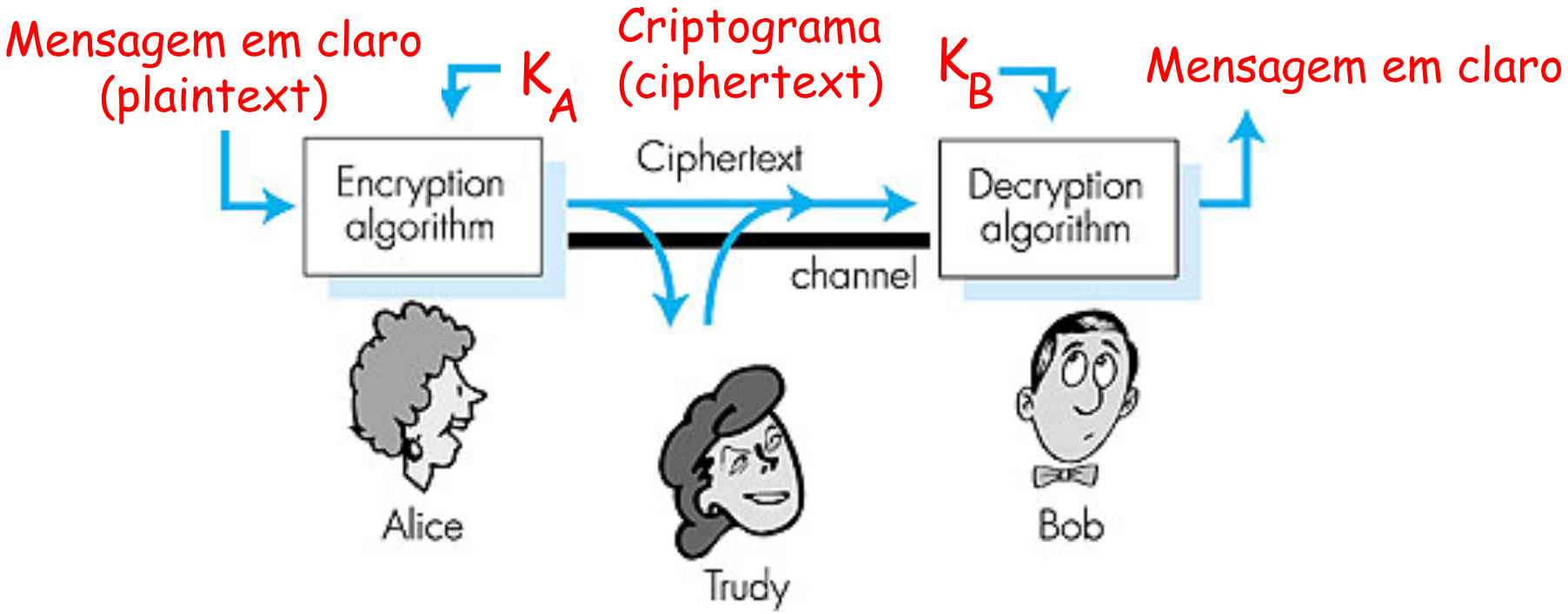
A transformação dos dados por técnicas criptográficas podem ser:

reversível : criptografia simétrica ou assimétrica

irreversível : funções de síntese/*hash* seguras / *message digests*

chave criptográfica é um parâmetro utilizado com um algoritmo criptográfico para transformar, validar, autenticar, cifrar ou decifrar dados

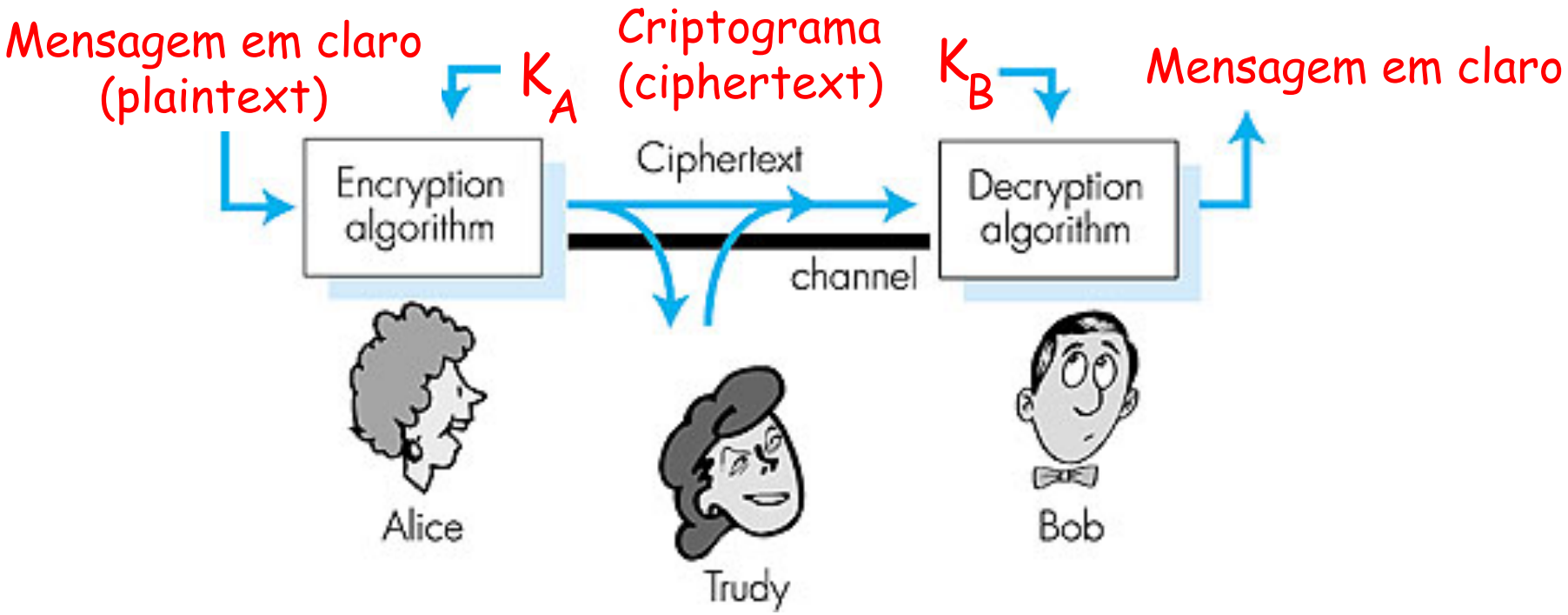
CRIPTOGRAFIA: CHAVE SIMÉTRICA



Criptografia de chave simétrica : a mesma chave cifra e decifra informação

Base mais comum para ofuscar informação e oferecer privacidade de dados (em grande volume)

CRIPTOGRAFIA: CHAVE ASSIMÉTRICA



Criptografia de chaves assimétrica ou de chave pública : cifra-se com a chave pública, decifra-se com a chave privada do receptor, ou vice-versa

Permite autenticação, assinar, comunicação anónima segura.

CRIPTOGRAFIA: FUNÇÕES DE SÍNTESE SEGURAS

As funções de síntese segura são funções de dispersão (hash) não invertíveis e com reduzida probabilidade de colisão.

Os dados de entrada são transformados (digeridos) de modo a obter um número reduzido de *bits (o hash).*

Além de não ser possível reconstruir o texto original,

o algoritmo será seguro quando é computacionalmente caro produzir uma colisão, ou seja alterar o texto original de forma mínima e produzir o mesmo *hash*.

Usam-se para garantir integridade e gerar assinaturas digitais de pequena dimensão

CRIPTOGRAFIA : DERIVADOS

Por combinação das primitivas criptográficas anteriores é possível construir derivados com garantias mais elaboradas:

Assinaturas digitais – atestam a procedência e autenticidade de um documento, evitam o seu repúdio.

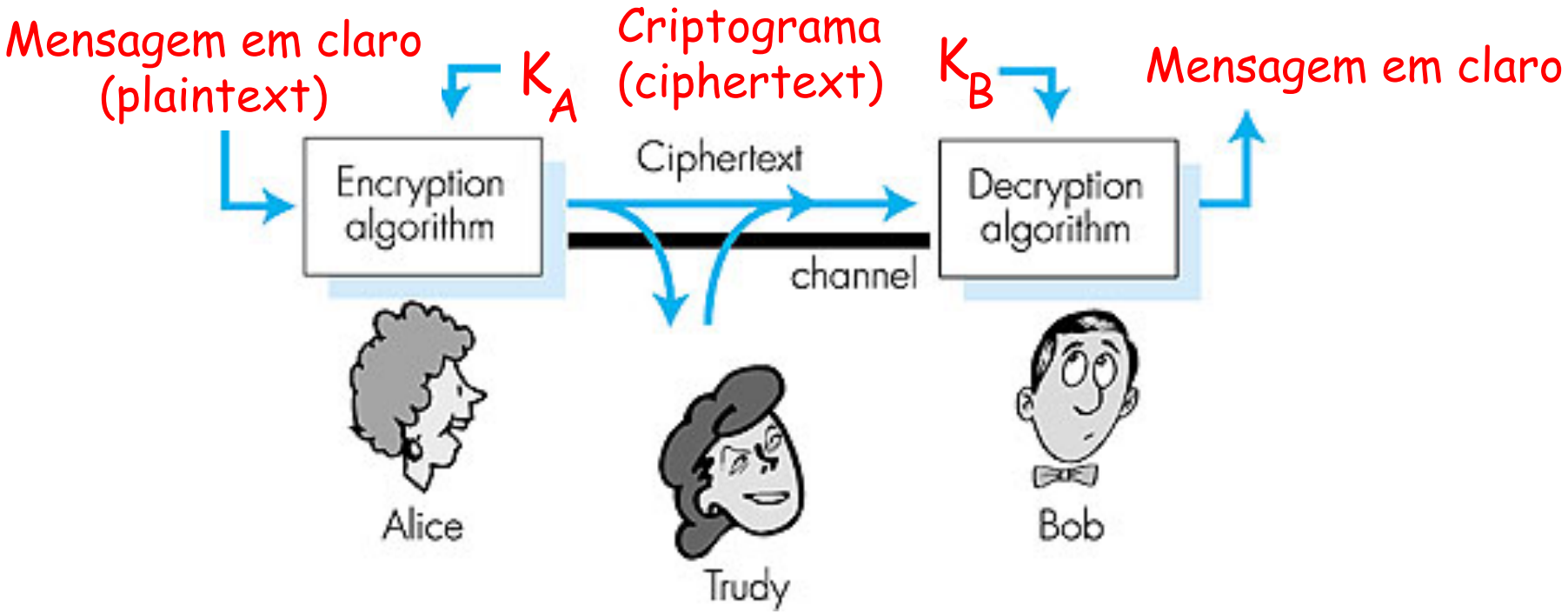
Certificados – permite atestar a identidade de principais com base em informação pública. Utilizada entidades terceiras idóneas (provedor de certificados).

CRIPTOGRAFIA

Criptografia é a disciplina que inclui os princípios, meios e métodos de transformação dos dados com a finalidade de:

- esconder o seu conteúdo semântico
- estabelecer a sua autenticidade
- impedir que a sua alteração passe despercebida
- evitar o seu repúdio

A LINGUAGEM CRIPTOGRÁFICA



Criptografia de chave simétrica : as chaves de cifra e de decifra são idênticas

Criptografia de chaves assimétrica ou de chave pública : cifra-se com a chave pública, decifra-se com a chave privada do receptor, ou vice-versa

CRIPTOGRAFIA CLÁSSICA – EX.: CÓDIGO DE CÉSAR

Uma cifras de substituição são um método onde a mensagem é ofuscada pela substituição de um símbolo por outro

Código de César é uma cifra de substituição para texto, considerando A=0, B=1, etc, equivalente a fazer um rotação ao alfabeto:

E(n,x) = (x+n) mod 26
D(n,x) = (x-n) mod 26

Ex.: para n = 3, têm-se as seguintes substituições

```
   ABCDEFGHIJKLMNOPQRSTUVWXYZ
   DEFGHIJKLMNOPQRSTUVWXYZABC
E(3,"SISTEMAS DISTRIBUIDOS")="VLVWHPDV GLVWULEXLGRV"
```

Usado por Júlio César para cifrar mensagens militares

CRIPTOANÁLISE – EXEMPLO ROT13

http://www.rot13.com/ Cifra de César, E(13,x)

NF NEZNF R BF ONEÕRF NFFVANYNQBF,

DHR QN BPVQRAGNY CENVN YHFVGNAN,

CBE ZNERF AHAPN QR NAGRF ANIRTNQBF,

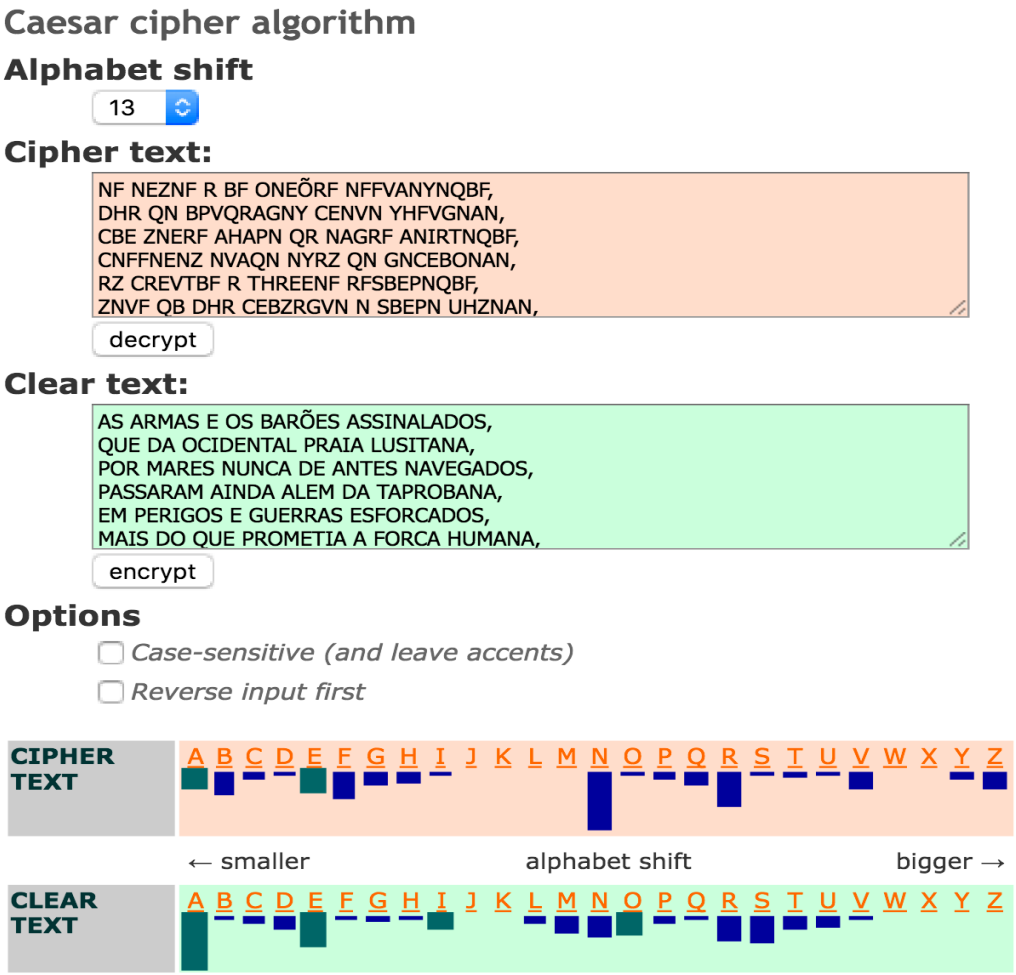
CNFFNENZ NVAQN NYRZ QN GNCEBONAN,

RZ CREVTBF R THREENF RFSBEPNQBF,

ZNVF QB DHR CEBZRGVN N SBEPN UHZNAN,

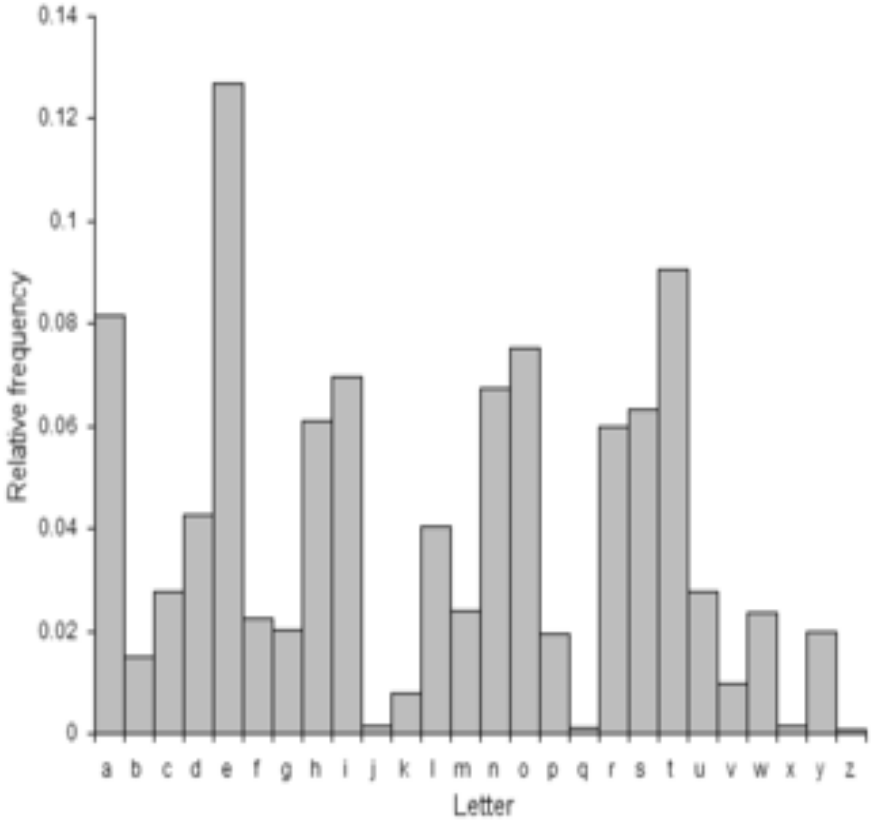
R RAGER TRAGR ERZBGN RQVSVPNENZ

ABIB ERVAB, DHR GNAGB FHOYVZNENZ;



CRIPTOANÁLISE – EXEMPLO

- Análise de frequência: ideia aplicada ao código de César:
  - sabe-se que um símbolo x é substituído sempre por outro y
  - sabe-se a frequência do aparecimento dos diferentes símbolos no texto
  - O símbolo mais frequente no texto cifrado deve corresponder a um dos símbolos mais frequentes usualmente
- Em geral, se se souber algum texto da mensagem, pode-se usar essa informação para inferir parte da chave
  - Ataque à Enigma (WWII)



EFICÁCIA DA CRIPTOGRAFIA

O algoritmo criptográfico será tanto melhor quanto mais difícil for obter o texto original a partir do texto cifrado, sem se conhecer a chave.

A eficácia de um ataque depende de dois factores:

- Algoritmo de cifra
- Cardinalidade do domínio da chave, isto é, número de bits da chave

Métodos de ataque:

- "**Força bruta**" - baseia-se na exploração sistemática de todas as chaves possíveis
- **Criptoanalíticos** - baseia-se em explorar os métodos matemáticos utilizados em criptografia para descobrir como decifrar os dados (ou diminuir o número de possibilidades)

EFICÁCIA DA CRIPTOGRAFIA

Nenhum algoritmo criptográfico é inteiramente seguro se o número de bits da chave tornar um ataque "força bruta" realista no quadro de dois factores:

O "valor" da informação

A capacidade computacional do atacante

ATAQUES FORÇA BRUTA

Supondo que é possível arranjar mil milhões (10^3 x 10^6) de computadores, capazes de testarem mil milhões de chaves por segundo cada um, então é possível testar 10^18 chaves por segundo.

Uma chave de 128 bits tem cerca de 3.4 x 10^38 combinações (10 levantado a 38). Com o poder computacional indicado, seriam necessários 10^13 anos para completar o ataque

Curiosidade: estima-se que a idade do universo é de cerca de 10^10 anos.

Uma chave de 54 bits tem cerca de 6 x 10^16 combinações. Com o poder computacional indicado, seria necessário menos de um segundo para descobrir a chave.

Conclusão: chaves geradas de forma "totalmente" aleatória, e com um número de bits suficiente, são relativamente seguras quando sujeitas a ataques do tipo "força bruta"

CONFIANÇA NOS ALGORITMOS CRIPTOGRÁFICOS

A prova matemática que um algoritmo criptográfico não tem falhas capazes de o tornar frágil perante ataques criptoanalíticos não é viável.

Consegue-se provar exatamente o contrário, através de exemplos.

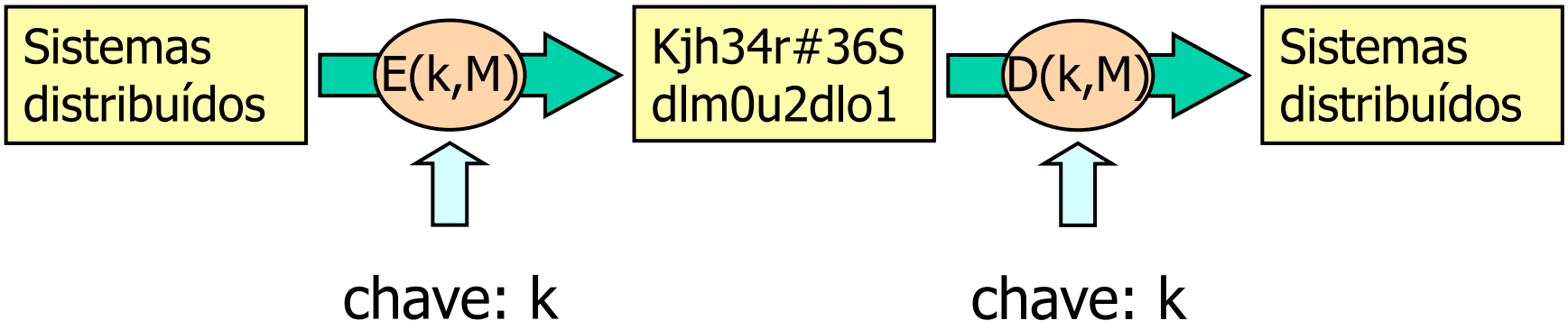
A segurança de um método é pois baseada em o mesmo ser público e ser sujeito ao escrutínio por especialistas.

> **A segurança deve ser resultado do segredo das chaves e não do segredo do algoritmo criptográfico**

A ciência criptográfica é uma disciplina "especializada".

CRIPTOGRAFIA SIMÉTRICA

- Parceiros devem partilhar chave secreta
- Mensagem é cifrada e decifrada com a mesma chave
  - E(k,M) = X
  - D(k,X) = M
- Que garantias dá? (confidencialidade, autenticação?)
- Garante confidencialidade das mensagens
  - Dado X, deve ser computacionalmente impossível obter M sem saber K, mesmo conhecendo E e D



ALGORITMOS DE CRIPTOGRAFIA SIMÉTRICA MAIS COMUNS

DES - Data Encryption Standard - chaves com 56 bits. OBSOLETO (< 1 dia)

Triple DES - DES reforçado – 3 chaves DES : 168 bits. OBSOLETO em 2030

IDEA - *International Data Encryption Algorithm* - chave com 128 bits, com origem na Europa. Após vários anos de utilização e divulgação pública não se conhecem ataques com êxito. (circa 2010)

> 2016 – Atacado! Revelou fraquezas em certos grupos chaves. Patentes expiraram em 2012.

AES/Rijndael - *Advanced Encryption Standard* - Nova norma U.S.A. Definida por concurso público, ganho por uma equipa belga. Admite chaves de 128, 192 ou 256 bits. Atacado!

IDEA & AES foram atacados!!! Perderam o equivalente a 2 bits na dimensão das chaves... (são seguros?)

CRIPTOGRAFIA ASSIMÉTRICA

Cada entidade tem duas chaves

- Chave **privada** (K_{priv}) é conhecida apenas pelo seu dono
- Chave **pública** (K_{pub}) pode ser conhecida por **todos**
- A partir de K_{pub} é impossível derivar K_{priv}

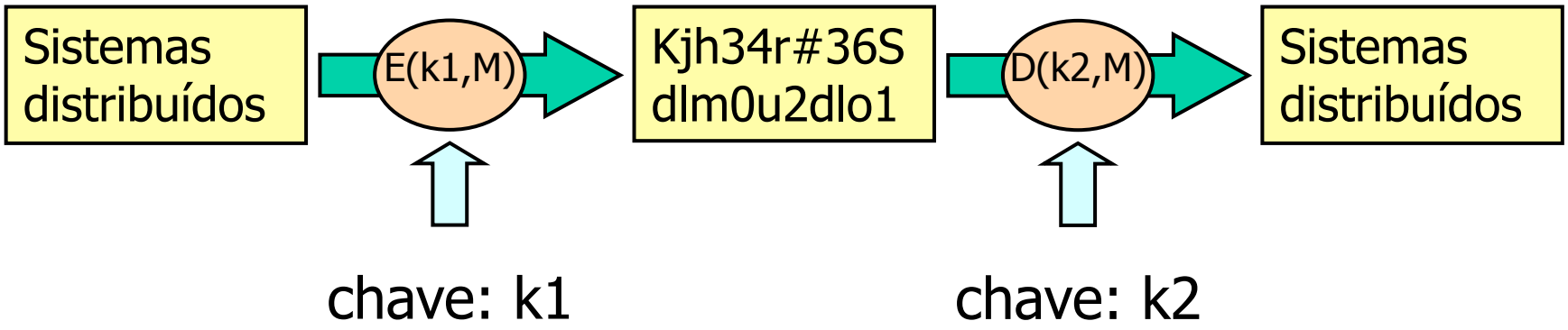
Mensagem é cifrada com uma chave e decifrada com a outra

$E(K_{pub},M) = X$; $D(K_{priv},X) = M$

- Garante o quê? (confidencialidade, autenticação?)

$E(K_{priv},M) = X$; $D(K_{pub},X) = M$

- Garante o quê? (confidencialidade, autenticação?)



CRIPTOGRAFIA ASSIMÉTRICA

Cada entidade tem duas chaves

- Chave privada (K_{priv}) é conhecida apenas pelo seu dono
- Chave pública (K_{pub}) pode ser conhecida por todos
- A partir de K_{pub} é impossível derivar K_{priv}

Mensagem é cifrada com uma chave e decifrada com a outra

$E(K_{pub},M) = X$; $D(K_{priv},X) = M$

- Garante confidencialidade
  - Conhecendo K_{pub} e X deve ser computacionalmente impossível obter M sem saber K_{priv}. Só receptor conhece K_{priv}.

$E(K_{priv},M) = X$; $D(K_{pub},X) = M$

- Garante autenticação do emissor
  - A partir de K_{pub} deve ser computacionalmente impossível obter K_{priv}. Só quem possui K_{priv} pode gerar X.

ALGORITMOS DE CRIPTOGRAFIA ASSIMÉTRICA MAIS COMUNS

RSA – algoritmo mais usado. Chave recomendada: 2048 bits.

- Patente RSA expirou em 2000
- Baseado na fatorização de números primos

Algoritmos de curva elíptica – método para gerar pares de chaves pública/privada baseado nas propriedades das curvas elípticas. Usa chaves de dimensão menor.

- Baseado no cálculo de logaritmos discretos ($b^k = g$)

Em ambos os casos a segurança advém do recurso a problemas considerados matematicamente difíceis.

Curiosidade: Algoritmo RSA (Rivest, Shamir e Adelman)

RSA Challenge (768 bits quebrado)
https://en.wikipedia.org/wiki/RSA_numbers#RSA-768

Para gerar as chaves

Escolhem-se dois números primos *grandes* , P e Q (P e $Q > 10^{100}$)

$N = P*Q$, $Z=(P-1) * (Q-1)$

Escolhe-se *d*, que pode ser qualquer número menor que N que seja primo em relação a Z

Calcula-se *e*, que é um número tal que *e.d-1* é divisível por Z

A chave privada é (N,e); a chave pública é (N,d)

A função de cifra é

$E((e,N), M) = M^e \bmod N = c$

A função de decifra é

$D((d,N), c) = c^d \bmod N = M$

Exemplo:

P=5

Q=11

N=55

Z=40

d=3

e=7

$E(2)= 2^7 \bmod 55 = 18$

$D(18)=18^3 \bmod 55 = 2$

NOTA: Segurança do método depende da dificuldade de factorizar N em P e Q

FUNÇÕES DE SÍNTESE

Objectivo: uma função de síntese segura H (*Message Digest* ou *Secure Hash*) deve produzir um sequência pequena de bits (128, 160, 512,...) que permita identificar uma mensagem de qualquer dimensão

Propriedades:

1. Calcular H(M) é fácil (*computacionalmente*)
2. Dado H(M) é *computacionalmente* impossível calcular M
3. Dado M é *computacionalmente* impossível descobrir M2 tal que H(M) = H(M2)

As funções de síntese com estas propriedades dizem-se "*Secure one-way hash functions*" ou "funções de dispersão unidirecionais seguras".

Porquê unidirecional e seguro?

Para que é que serve?

Usado para garantir integridade dos dados

FUNÇÕES SEGURAS DE SÍNTESE

Função segura de síntese MD5 (abandonada)

- Calcula sínteses de 128 bits num processo em 4 fases
- Uma das funções seguras de síntese computacionalmente mais eficientes
  - Conhecidos ataques que permitem gerar colisões.

A função SHA-1 é também muito usada. (Em abandono)

- Norma USA. Conhecida publicamente
  - Conhecidos ataques que permitem diminuir o espaço de pesquisa para uma colisão para $O(2^{63})$ (em vez de (2^{80}))
  - Está-se a migrar para SHA-2 e SHA-3
- Produz uma síntese de 160 bits

Substitutos: SHA-2 (hashes de 224 a 512 bits)

> 2015 : SHA-3 (hashes de 224 a 512 bits)

O DESEMPENHO DOS DIFERENTES MÉTODOS

CIPHER	Débito (MB/s)
AES	< 100
Twofish	60
Serpent	32
IDEA	35
DES	32
RSA 1024 Cifrar	~3
RSA 1024 Decifrar	~0.2
RSA 2048 Cifrar	~1.5
RSA 2048 Decifrar	~0.04
MD5	255
SHA-1	153
SHA-256	111
SHA-512	99
CRC32	253
Adler32	920

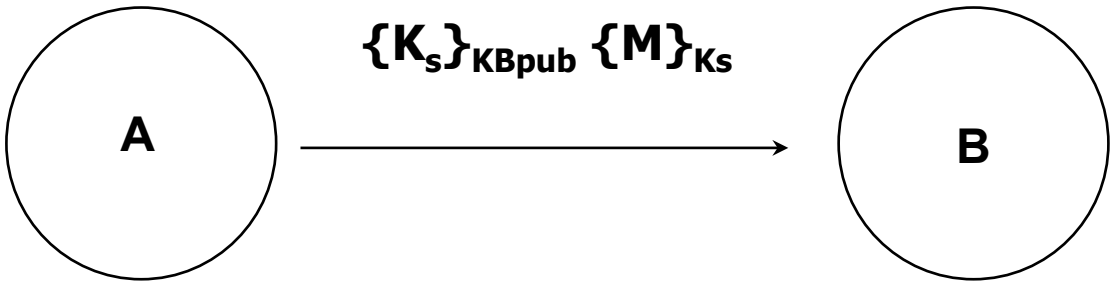
https://www.cryptopp.com/benchmarks.html

UTILIZAÇÃO CONJUNTA DOS MÉTODOS

A criptografia assimétrica é 100 a 1000 vezes mais pesada que a criptografia simétrica. Por esta razão, os dois métodos são utilizados em conjunto. Como?

As chaves públicas são utilizadas para **autenticação** e para **acordar** uma **chave de sessão** para utilização posterior de criptografia simétrica entre os dois principais.

Exemplo:



NOTAÇÕES MAIS FREQUENTES

K_A	Alice's secret key
K_B	Bob's secret key
K_{AB}	Secret key shared between Alice and Bob
K_{privA}	Alice's private key (known only to Alice)
K_{pubA}	Alice's public key (published for all to read)
$\{M\}_K$	Message M encrypted with key K
$[M]_K$	Message M signed with key K

Message M enc

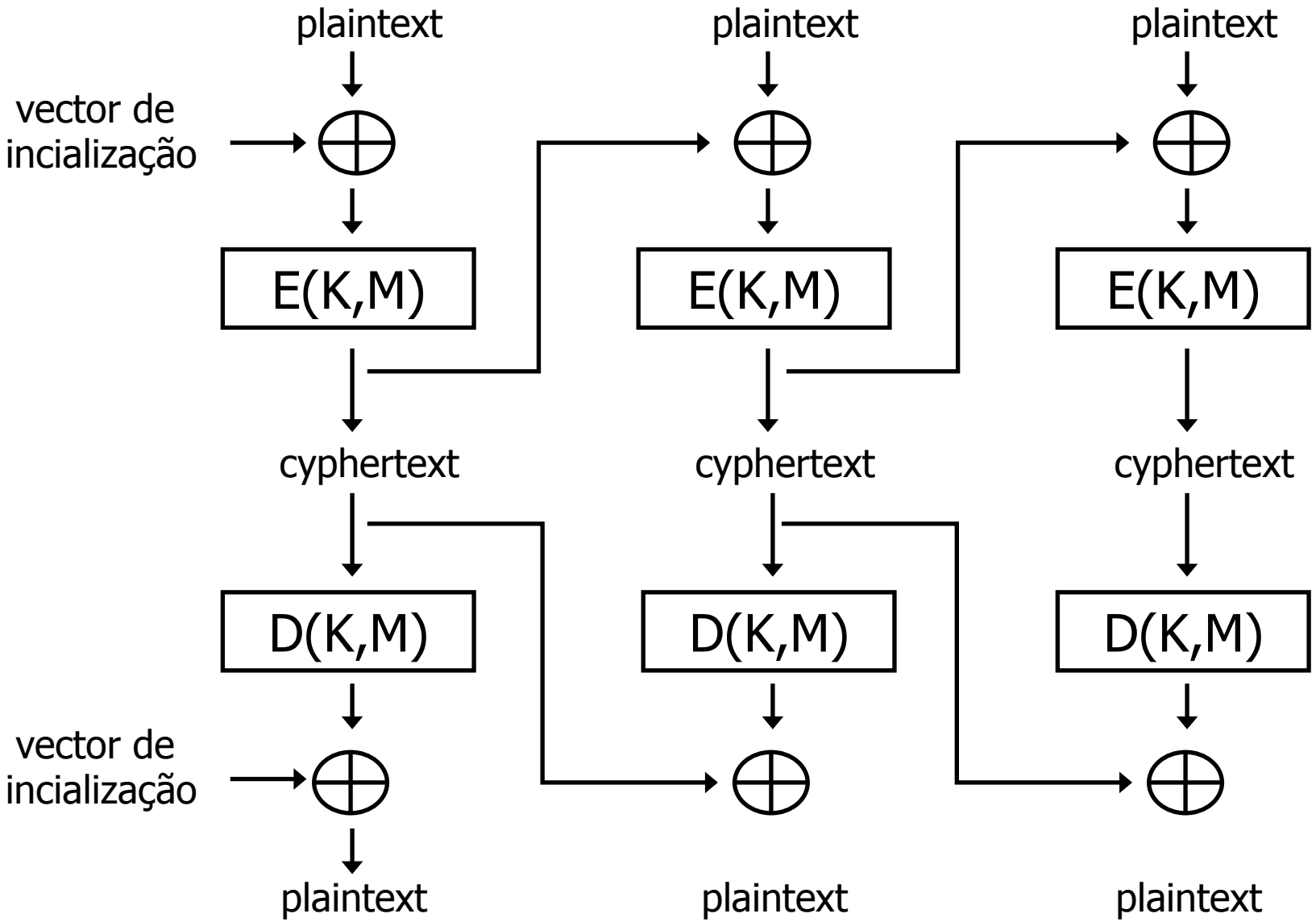
CIFRA POR BLOCOS ENCADEADOS

A cifra por blocos pode revelar-se frágil quando cada bloco é cifrado de forma independente.

> pode revelar padrões repetidos que facilitem a relação do texto cifrado com o texto em claro e facilita o ataque por métodos cripto-analíticos.

Uma técnica possível para melhorar o método é usar cifra por blocos encadeados (CBC - *cipher block chaining*)

CIFRA POR BLOCOS ENCADEADOS



CIFRA POR BLOCOS ENCADEADOS

Na cifra por blocos encadeados:

Antes de cifrar um bloco, faz-se XOR com o resultado da cifra anterior

Após decifrar um bloco faz-se XOR com o bloco anterior cifrado, para obter o texto em claro.

Usa-se um vector de inicialização (por exemplo uma *timestamp* da dimensão de um bloco) para iniciar o processo.

O mesmo texto, cifrado com a mesma chave, mas com vectores iniciais distintos, conduz a resultados distintos.

Esta técnica exige canais de comunicação **fiáveis** pois a perca de uma mensagem impede o processo de decifra.

COMO EVITAR QUE O RECEPTOR FIQUE "BARALHADO"

- Se as mensagens a cifrar só contiverem bits significativos e válidos para as transacções em curso, a sua decifra com uma chave errada conduz, apesar disso, a um padrão de bits que pode ter significado (errado) para o receptor
- Problema?
- Este facto pode ser usado para levar o receptor a executar "disparates" o que pode ser uma fonte de insegurança
  - Pode ser usado para ataques de impedimento de prestação de serviço tentando saturar o receptor a processar dados aleatórios.
- Solução?
- Por este motivo, é necessário introduzir bits redundantes antes de cifrar as mensagens que impeçam o receptor de ficar "baralhado". As soluções mais simples são:
  - colocar um CRC
  - colocar um número de ordem
  - colocar um valor fixo pré-estabelecido, etc.

CONTROLO DE INTEGRIDADE COM FUNÇÕES DE SÍNTESE SEGURAS

É possível usar as funções de síntese para controlar a integridade num canal de dados não seguro. O método consiste em usar MACs ("*Message Authentication Codes*") que são assinaturas, computacionalmente fáceis de calcular, baseadas em chaves secretas.

O método funciona assim:

1. A e B estabelecem uma chave secreta K só conhecida por ambos. K pode ser trocada aquando da autenticação, por exemplo.
2. Para A enviar a mensagem M a B:
   Calcula h = H(M+K)
   Envia M,h
3. Ao receber "M,h", B calcula h'=H(M+K) e verifica integridade da mensagem se h=h'
   Porque é que garante autenticação e integridade?
   Só A conhece K, logo só A consegue gerar H(M+K) – autentica emissor de M e garante que a mensagem não foi modificada

NOTA: Não se pretende garantir confidencialidade

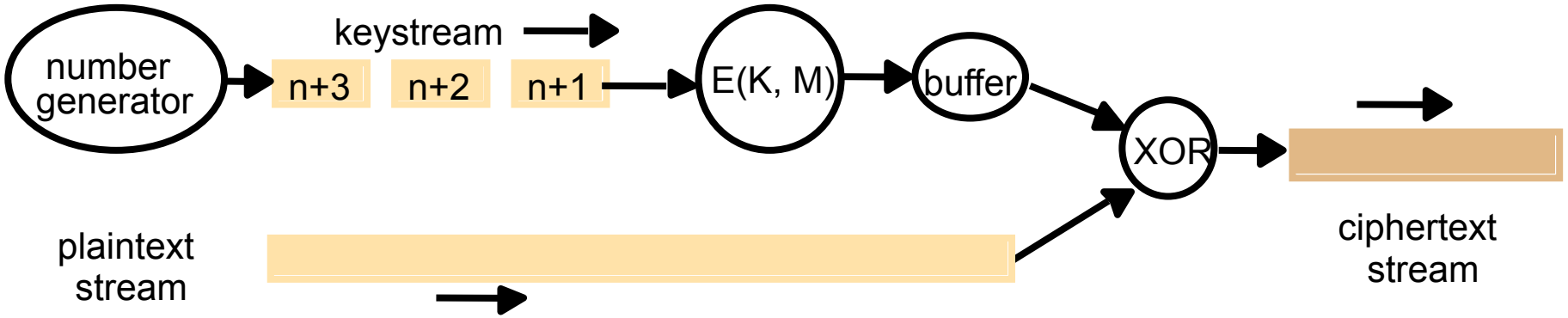
CIFRA DE STREAMS

Um algoritmo de cifra por blocos apenas pode cifrar bloco a bloco

Quando se pretende cifrar dados de dimensão inferior a um bloco é necessário preparar um bloco que contenha os dados a enviar (padding) e cifrá-lo [impacto nas sessões interativas???]

Alternativamente, pode-se obfuscar a mensagem a transmitir usando um stream (keystream) criado a partir de um gerador de números aleatórios (fazendo XOR)

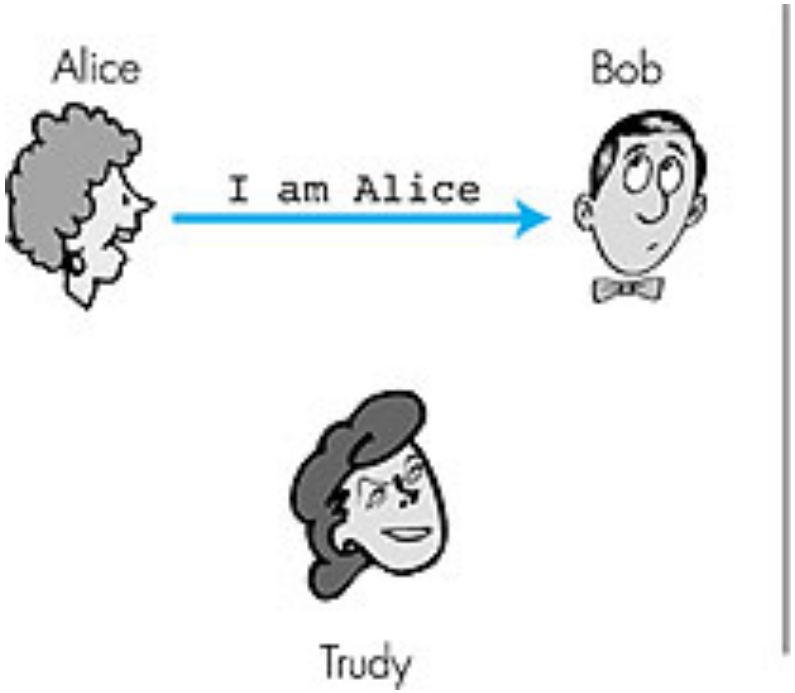
Como decifrar a mensagem?



AUTENTICAÇÃO

Objectivo: Bob quer que a Alice lhe "demonstre" a sua identidade

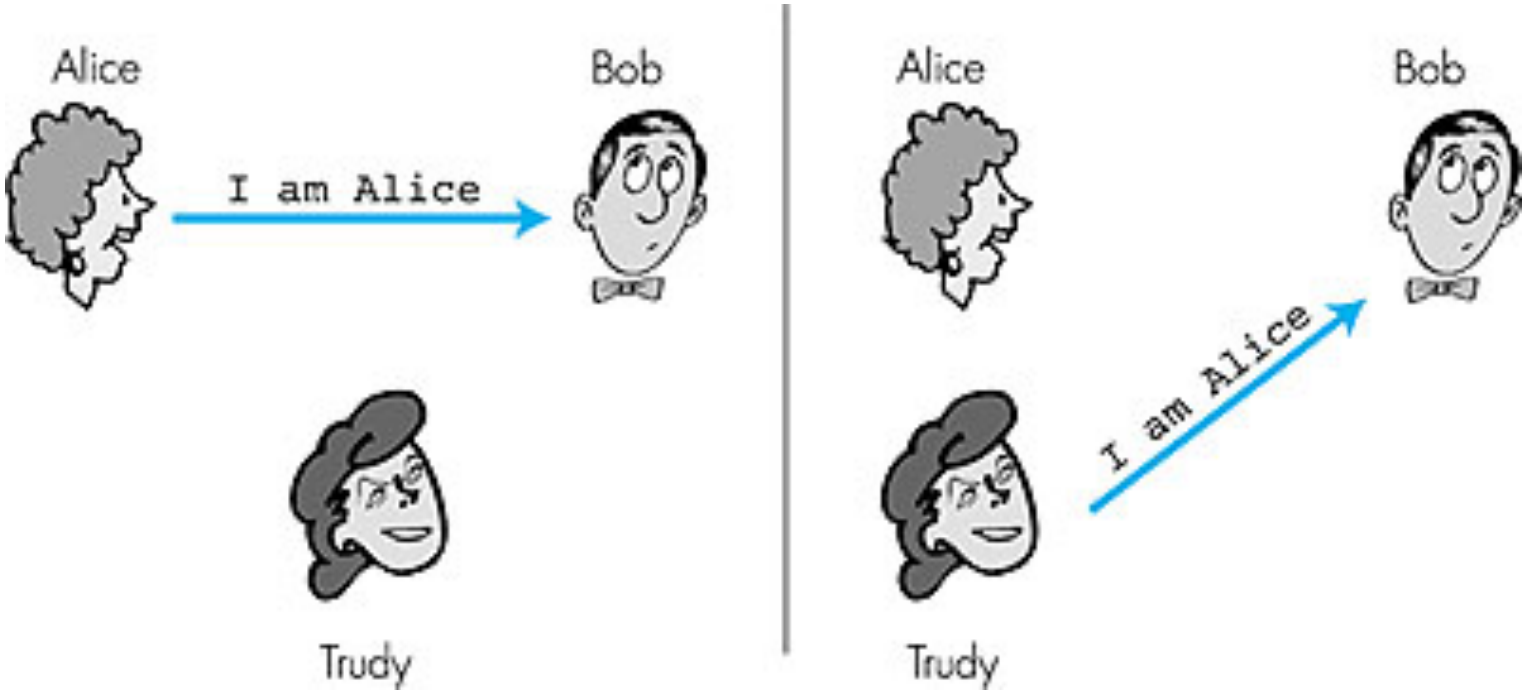
Protocolo 1: Alice diz simplesmente "I am Alice"



AUTENTICAÇÃO

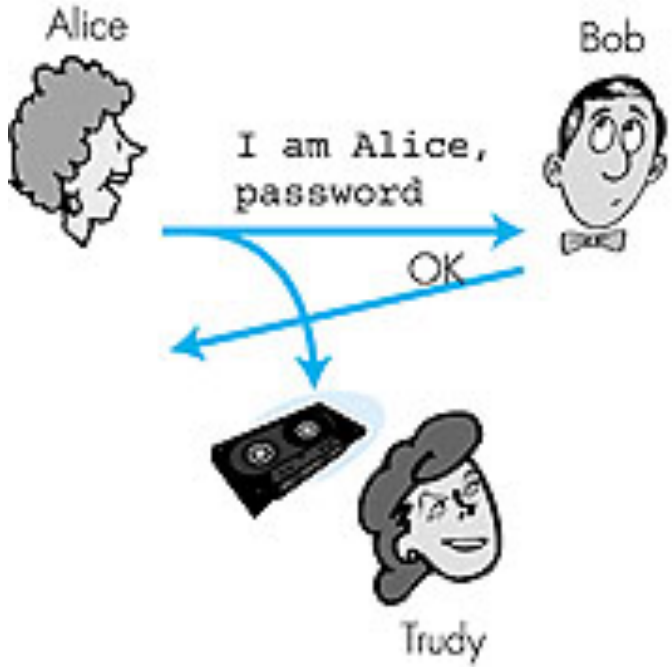
Objectivo: Bob quer que a Alice lhe "demonstre" a sua identidade

Protocolo 1: Alice diz simplesmente "I am Alice"



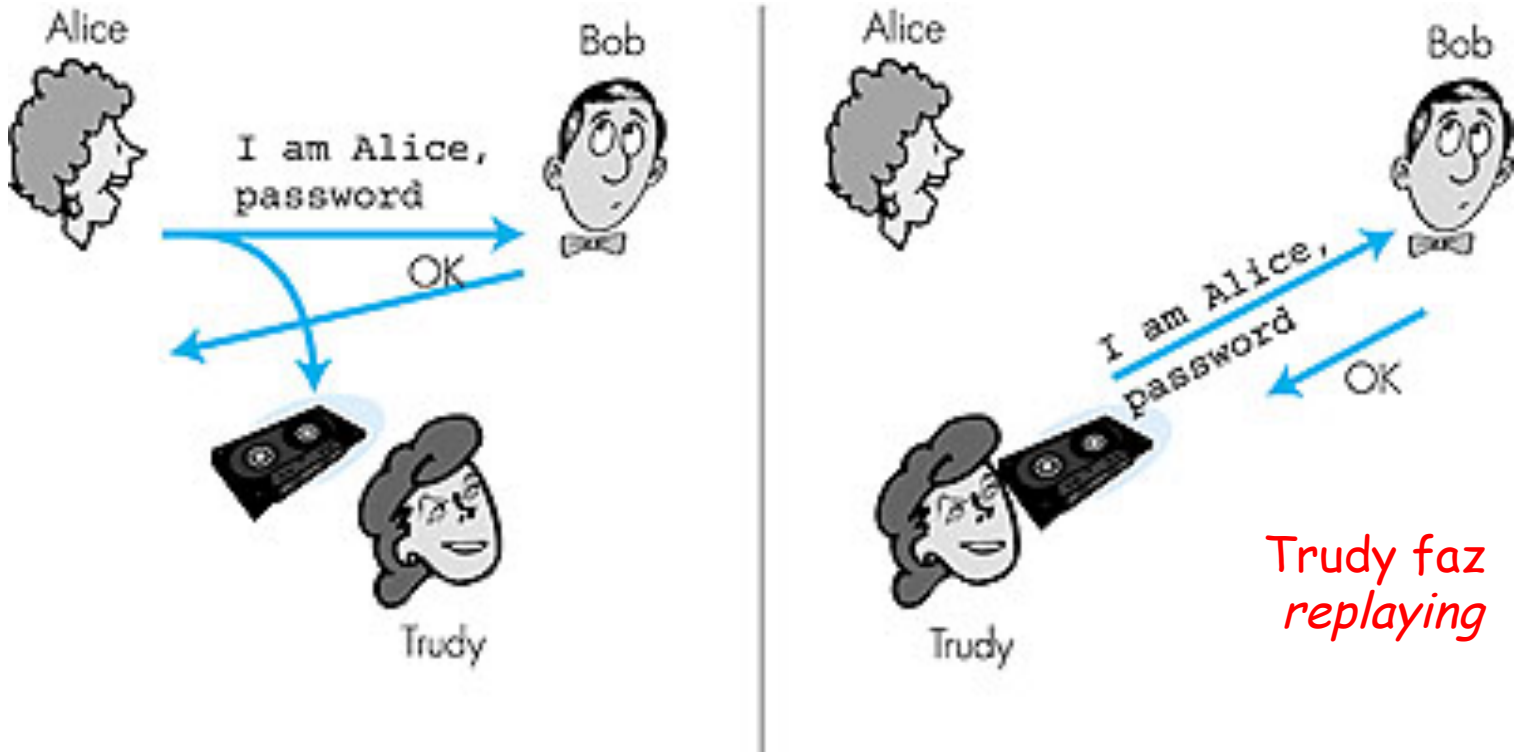
SEGUNDA TENTATIVA

Protocolo 2: Alice diz "I am Alice" e envia também a sua "password"



SEGUNDA TENTATIVA

Protocolo 2: Alice diz "I am Alice" e envia também a sua "password"

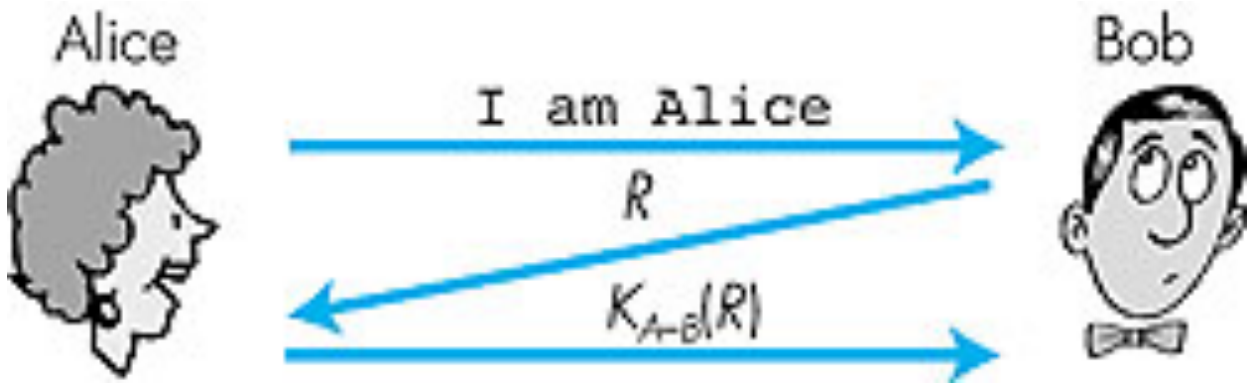


MÉTODO DESAFIO / RESPOSTA

Objectivo: evitar o ataque por *replaying*

Nonce: número usado uma única vez

Protocolo 3: para garantir a "frescura" da transacção, Bob envia à Alice o *nonce*, R. Alice deve devolver R cifrado com a chave secreta que ambos partilham.



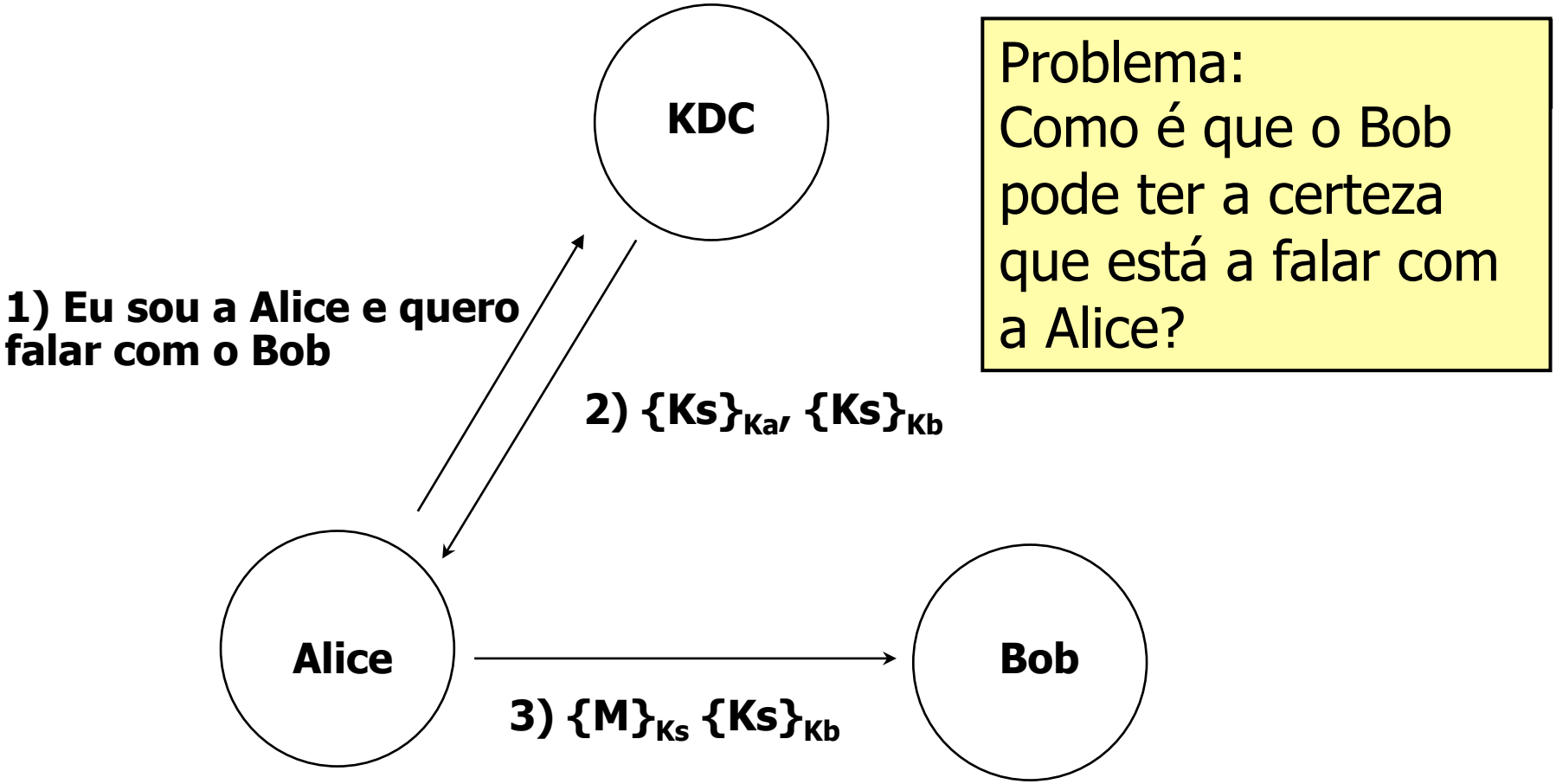
Problemas e possíveis soluções

Apesar de a chave secreta não necessitar de passar na rede durante a autenticação, Alice e Bob têm de arranjar alguma forma de se porem previamente de acordo sobre a chave secreta que vão usar.

Dado que tal chave não pode ser passada na rede, terão de se encontrar ou usar uma terceira pessoa para trocarem a chave.
Se isso se revelar um método complexo, vão ter tendência a manter a mesma chave durante muito tempo, o que é perigoso.

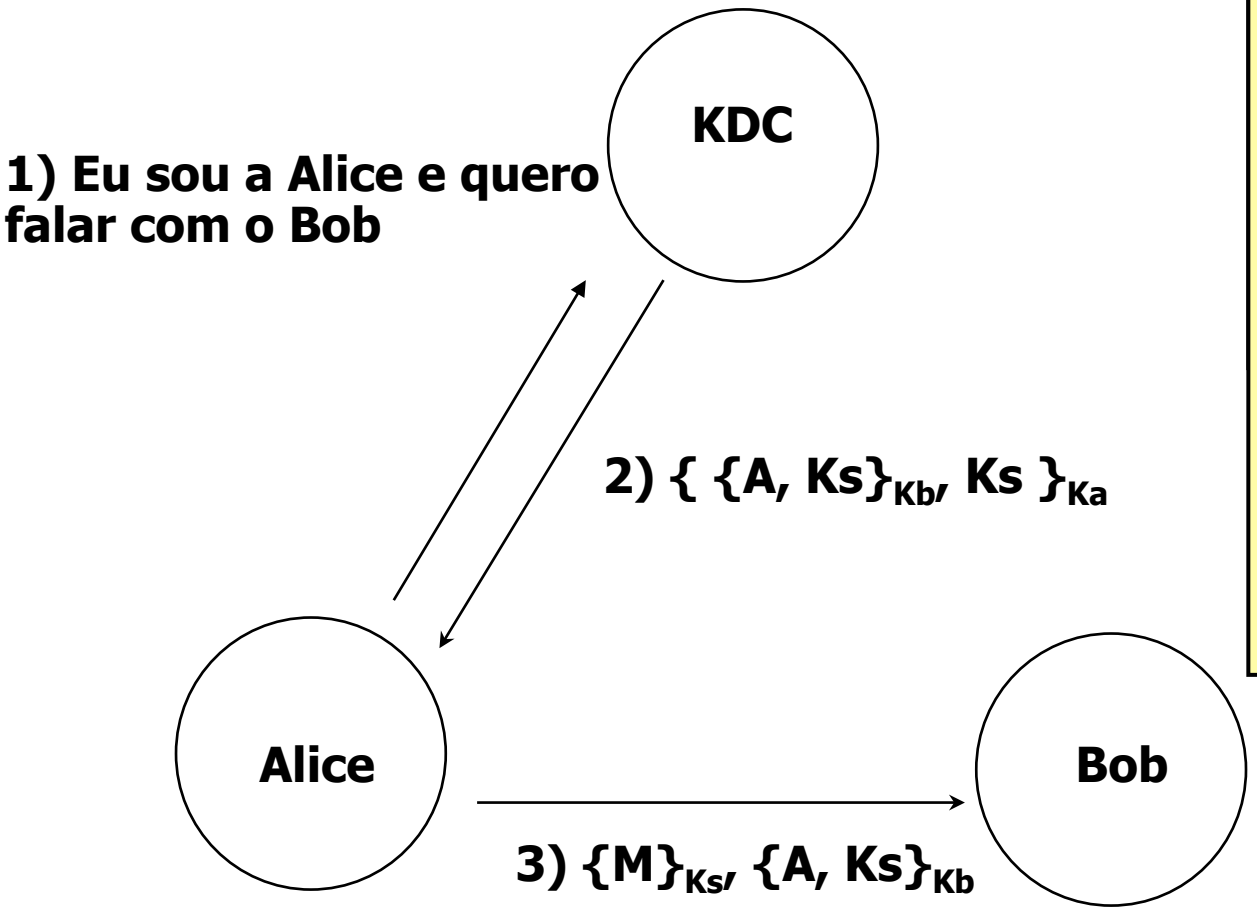
Solução: usar uma chave diferente em cada sessão e usar um **centro de distribuição de chaves** no qual Alice e Bob **confiam**

DISTRIBUIÇÃO DE CHAVES DE SESSÃO ATRAVÉS DE UM KDC – PRIMEIRA TENTATIVA



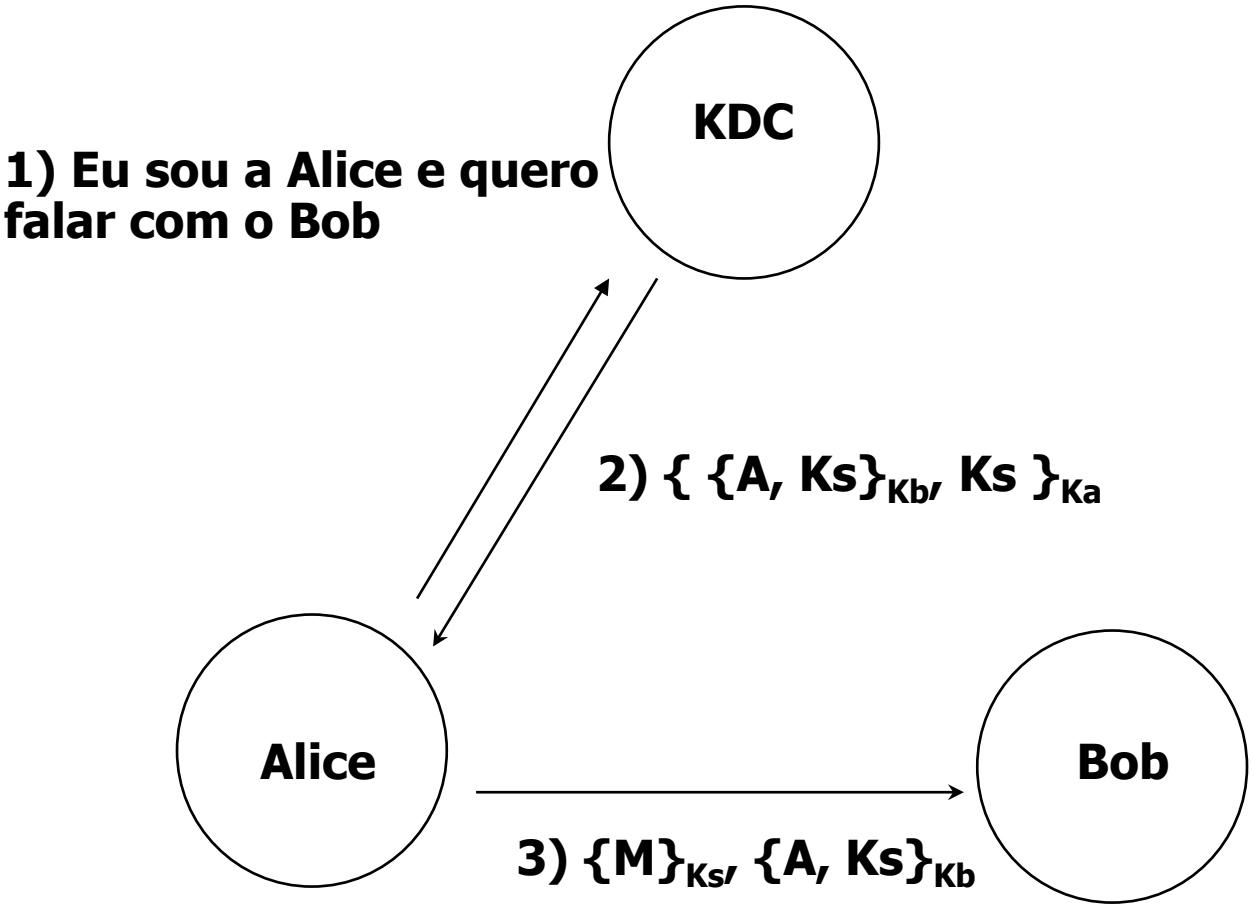
Problema:
Como é que o Bob pode ter a certeza que está a falar com a Alice?

DISTRIBUIÇÃO DE CHAVES DE SESSÃO ATRAVÉS DE UM KDC – SEGUNDA TENTATIVA



Problema:
Como é que Bob pode ter a certeza que está a falar com Alice?
Por Alice ter sido capaz de obter $\{A,Ks\}K_b$. Só Alice conhece K_a

DISTRIBUIÇÃO DE CHAVES DE SESSÃO ATRAVÉS DE UM KDC – SEGUNDA TENTATIVA



Problema:
Pode Trudy obter o conteúdo da mensagem?
Não, porque apenas Bob conhece K_b. Mas Trudy pode gravar a mensagem e mais tarde tentar incomodar Bob.

PROTOCOLO DE *NEEDHAM-SCHROEDER*

Este protocolo permite a dois principais A e B (Alice e Bob) estabelecerem um canal seguro entre si e autenticarem-se mutuamente.

Baseia-se na presença de um KDC (Key Distribution Center) que conhece as chaves secretas de A e B (Ka e Kb) e que é capaz de gerar chaves de sessão (Ks) que vão ser usadas por A e B para comunicarem de forma segura.

O protocolo resiste aos ataques *eavesdropping*, *masquerading*, *message tampering* e *replaying*.

O PROTOCOLO NS COM CHAVES SIMÉTRICAS

1) A -> KDC: **A, B, Na**

A solicita uma chave para comunicar com *B*, sendo *Na* um número aleatório gerado por *A* para garantir a unicidade da transacção (*Na* deve ser único).

2) KDC -> A: **{ Na, B, Ks, { Ks, A }K_b }K_a**

O *KDC* devolve Na, a chave e um *ticket* ({ Ks, A }Kb), tudo cifrado com a chave de A.

3) A -> B: **{Ks, A}K**b**, {Na'}Ks**

A solicita comunicar com B, enviando-lhe o *ticket*

4) B -> A: **{Na'-1}Ks**

O PROTOCOLO NS COM CHAVES SIMÉTRICAS

1) A -> KDC: **A, B, Na**

A solicita uma chave para comunicar com *B*, sendo *Na* um número aleatório gerado por *A* para garantir a unicidade da transacção (*Na* deve ser único).

2) KDC -> A: **{ Na, B, Ks, { Ks, A }Kb }Ka**

O *KDC* devolve Na, a chave e um *ticket* ({ Ks, A }Kb), tudo cifrado com a chave de A.

O que garante a A ter falado com o KDC ?

3) A -> B: **{Ks, A}Kb, {Na'}Ks**

A solicita comunicar com B, enviando-lhe o *ticket*

4) B -> A: **{Na'-1}Ks**

O PROTOCOLO NS COM CHAVES SIMÉTRICAS

1) A -> KDC: **A, B, Na**

A solicita uma chave para comunicar com *B*, sendo *Na* um número aleatório gerado por *A* para garantir a unicidade da transacção (*Na* deve ser único).

2) KDC -> A: **{ Na, B, Ks, { Ks, A }Kb }Ka**

O *KDC* devolve Na, a chave e um *ticket* ({ Ks, A }Kb), tudo cifrado com a chave de A.

A recepção do *Na* único garante que A comunicou com o KDC – só o KDC conhece Ka, logo só KDC pode gerar a mensagem indicada.

3) A -> B: **{Ks, A}K**b**, {Na'}Ks**

A solicita comunicar com B, enviando-lhe o *ticket*

O que garante a B estar a falar com A ?

4) B -> A: **{Na'-1}Ks**

O PROTOCOLO NS COM CHAVES SIMÉTRICAS

1) A -> KDC: **A, B, Na**

A solicita uma chave para comunicar com *B*, sendo *Na* um número aleatório gerado por *A* para garantir a unicidade da transacção (*Na* deve ser único).

2) KDC -> A: **{ Na, B, Ks, { Ks, A }Kb }Ka**

O *KDC* devolve Na, a chave e um *ticket* ({ Ks, A }Kb), tudo cifrado com a chave de A.

A recepção do *Na* único garante que A comunicou com o KDC – só o KDC conhece Ka, logo só KDC pode gerar a mensagem indicada.

3) A -> B: **{Ks, A}K**b**, {Na'}Ks**

A solicita comunicar com B, enviando-lhe o *ticket*

B sabe que está a falar com A, porque apenas A pode obter Ks associada a um ticket indicando A (porque Ks passa cifrado com Ka em 2)). Além disso, é impossível forjar um ticket, porque só B e KDC conhecem Kb

4) B -> A: **{Na'-1}Ks**

O que garante a A estar a falar com B ?

O PROTOCOLO NS COM CHAVES SIMÉTRICAS

1) A -> KDC: **A, B, Na**

A solicita uma chave para comunicar com *B*, sendo *Na* um número aleatório gerado por *A* para garantir a unicidade da transacção (*Na* deve ser único).

2) KDC -> A: **{ Na, B, Ks, { Ks, A }Kb }Ka**

O *KDC* devolve Na, a chave e um *ticket* ({ Ks, A }Kb), tudo cifrado com a chave de A.

A recepção do *Na* único garante que A comunicou com o KDC – só o KDC conhece Ka, logo só KDC pode gerar a mensagem indicada.

3) A -> B: **{Ks, A}K**b**, {Na'}Ks**

A solicita comunicar com B, enviando-lhe o *ticket*

B sabe que está a falar com A, porque apenas A pode obter Ks associada a um ticket indicando A (porque Ks passa cifrado com Ka em 2)). Além disso, é impossível forjar um ticket, porque só B e KDC conhecem Kb

4) B -> A: **{Na'-1}Ks**

B prova ser B por ser capaz de decifrar {Na'}Ks, o que pressupõe ter sido capaz de decifrar {Ks, A}Kb

Porque não pode B enviar {Na'}Ks ?

UM POSSÍVEL ATAQUE E RESPECTIVA SOLUÇÃO

A mensagem 3) pode ser *replayed* mais tarde. Isso, em princípio, não tem problema mas se por acaso Ks for um dia comprometida, então Trudy pode conseguir autenticar-se perante Bob como se fosse Alice. A solução consiste em acrescentar uma *timestamp* ao *ticket* que deve ser testada por Bob após ter recebido a mensagem 3). O protocolo fica:

1) A -> KDC: A, B, Na

2) KDC -> A: { Na, B, Ks, { Ks, A, t }Kb }Ka

3) A -> B: {Ks, A, t}K_b, {Na'}Ks

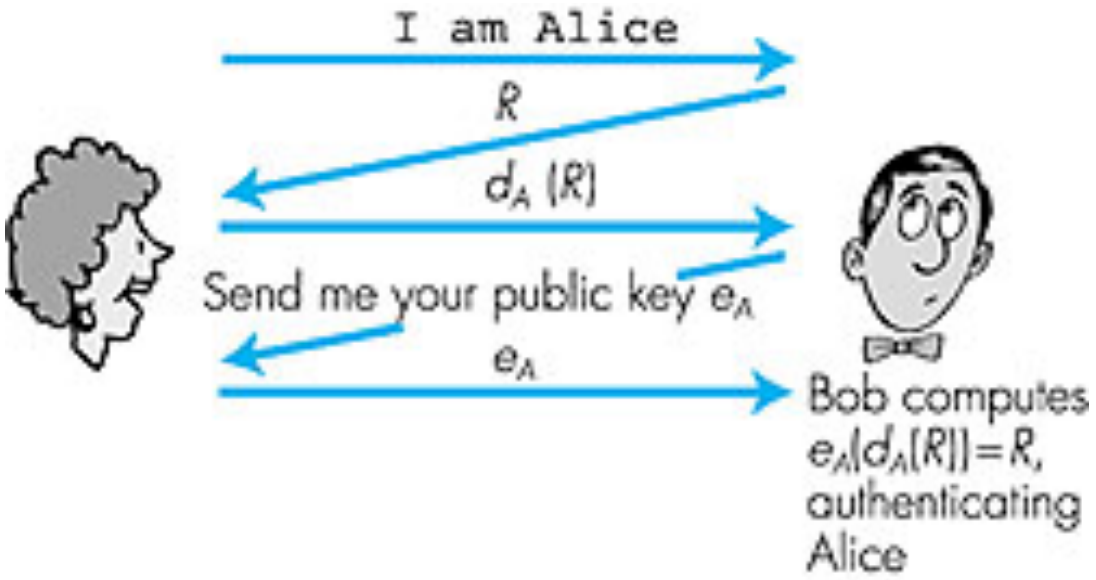
4) B -> A: {Na'-1}Ks

Exige que os relógios de A, B e do KDC estejam sincronizados a menos de um valor considerado suficiente para tornar impossível a quebra da chave Ks.

AUTENTICAÇÃO VIA CHAVE PÚBLICA

Problema: como é que Bob e Alice se autenticam utilizando chaves assimétricas?

Resposta: usa um *nonce*, e criptografia assimétrica



PROTOCOLO DE NEEDHAM-SCHROEDER COM CHAVES PÚBLICAS

Pressupondo que A e B **conhecem** as **chaves públicas** um do outro de forma **certificada**, podem estabelecer um canal seguro e autenticarem-se mutuamente através de:

1. **A -> B:** **{ A, Na }K_{Bpub}**
2. **B -> A:** **{ Na, Nb, Ks } K_{Apub}**
3. **A -> B:** **{ Nb } K_s**

O que garante a A estar a falar com B?

Receber Na em 2 – apenas B consegue obter Na, porque apenas B tem K_{Bpriv}

O que garante a B estar a falar com A?

Receber Nb em 3 – apenas A consegue obter Nb, porque apenas A tem $_{KApriv}$

O que garante que Ks é seguro?

Ks apenas passa na rede em 2 – apenas A consegue obter Ks, porque apenas A tem K_{Apriv} (B conhece K_s porque gerou K_s)

Criptografia assimétrica é lenta. Como solucionar o problema?

Negoceia chave simétrica para usar durante a comunicação

Protocolo de Needham-Schroeder com chaves públicas

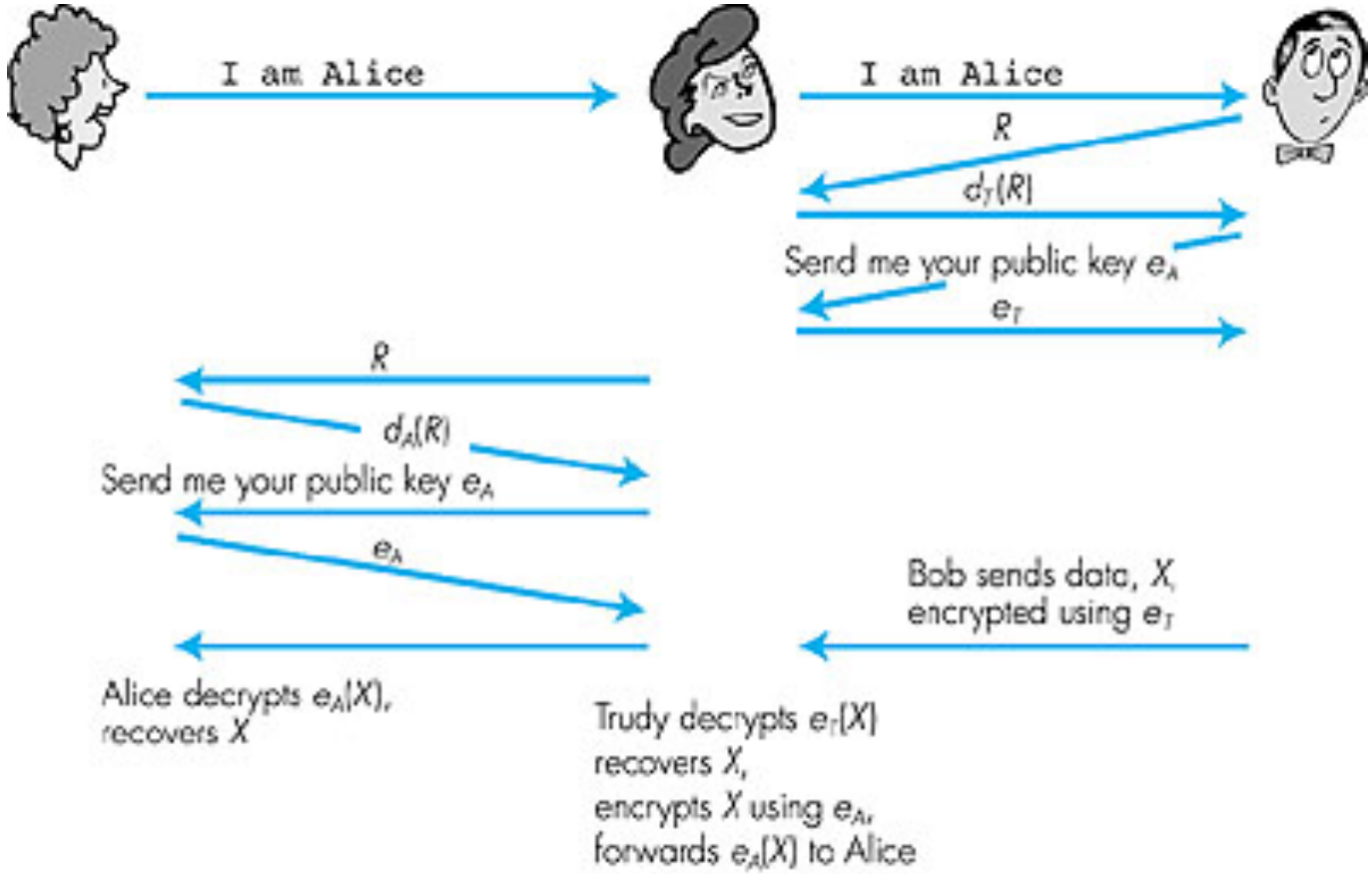
Se A apenas quiser criar um canal cifrado e ter a certeza de que está a falar com B basta:

A -> B: **{ A, Na, Ks }KBpub**

B -> A: **{ Na } Ks**

O que garante a A estar a falar com B?

ATAQUES POR INTERPOSIÇÃO

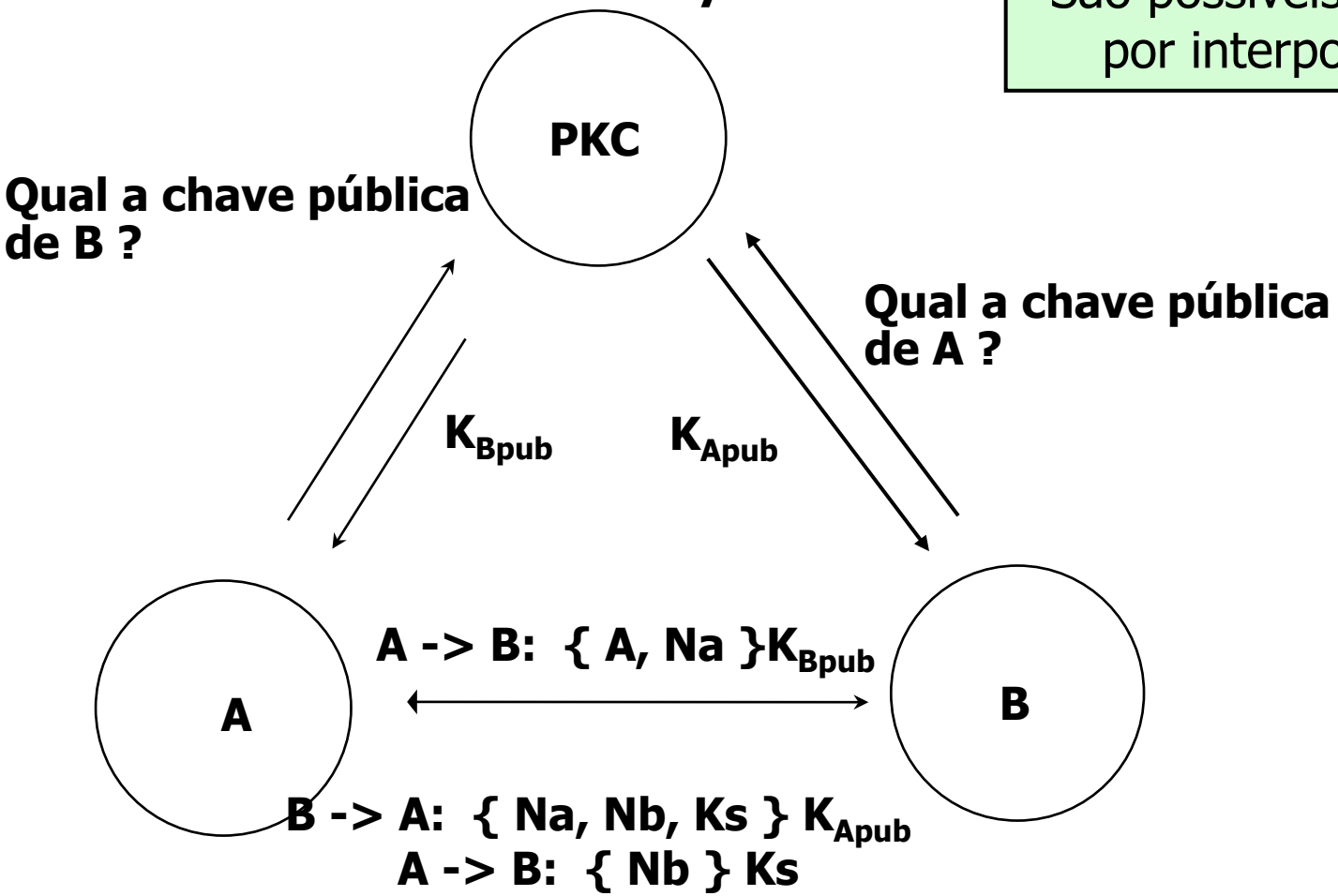


São necessárias chaves públicas *"certificadas"*

CENTRO DE DISTRIBUIÇÃO DE CHAVES PÚBLICAS (1)

PKC = Public Key Center

São possíveis ataques por interposição?



CENTRO DE DISTRIBUIÇÃO DE CHAVES PÚBLICAS (2)

O *Public Key Center* (PKC) apenas conhece aquilo que é público, isto é, as chaves públicas dos principais. Tal é um progresso notável pois o papel da *trusted computing base* foi drasticamente reduzido.

Para que tudo funcione bem é apenas necessário assegurar que o PKC tem as chaves públicas verdadeiras e que os principais têm a certeza que estão a falar com o verdadeiro PKC e não com um impostor.

Nos protocolos anteriores, um intruso que se consiga fazer passar pelo PKC, consegue levar A e B a usarem chaves conhecidas

São necessárias chaves públicas "*certificadas*"

DISTRIBUIÇÃO DAS CHAVES PÚBLICAS

É necessário ter absoluta segurança de que se está a dialogar com uma fonte fidedigna que nos está a entregar a verdadeira chave pública que pretendemos

Se se conhece a chave pública dessa fonte fidedigna, uma forma de obter esta confiança é essa fonte cifrar a sua resposta com a sua chave secreta

Métodos de distribuição de chaves:

Certificate Granting Authority – entidades cujas chaves públicas são bem conhecidas e que "assinam" as chaves / certificados que entregam. Este método está hoje em dia normalizado de forma oficial.

Web of trust - método informal por transitividade da relação de confiança, também suportado na "assinatura" das informações trocadas entre principais. Este método foi vulgarizado pelo programa PGP para troca de correio electrónico.

ASSINATURAS DIGITAIS

Objectivo: desenvolver um mecanismo digital que substitua as assinaturas efectuadas num documento em papel

Quais as propriedades?

Um sistema de assinaturas digitais deve ter as seguintes propriedades:

Autenticação: o receptor deve poder verificar que a assinatura é autentica

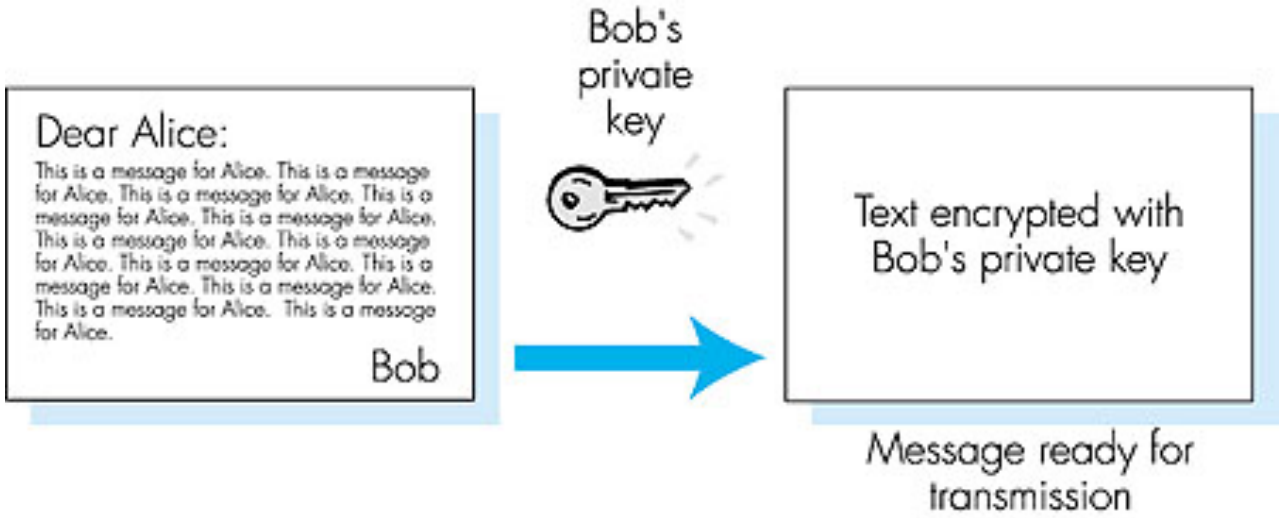
Integridade: a assinatura deve garantir que a mensagem assinada não foi alterada, nem durante o trajecto, nem pelo receptor, mesmo que tenha passado em claro

Não repudiamento: o emissor não poderá negar que de facto enviou a mensagem assinada

NOTA: O objectivo das assinaturas não é *esconder* o conteúdo

ASSINATURA DIGITAL: PRIMEIRA TENTATIVA

Assinatura digital da mensagem M efectuada por Bob: {M}KBpriv



- Propriedades:
  - Autenticação verificada decifrando M = $\{\{M\}K_{Bpriv}\}K_{Bpub}$
  - Integridade e não repudiamento garantidos porque ninguém consegue alterar/criar $\{M\}K_{Bpriv}$ sem conhecer K_{Bpriv}
  - Dimensão da assinatura idêntica à do documento
  - Necessidade de cifrar todo o documento com chaves assimétricas

Solução?

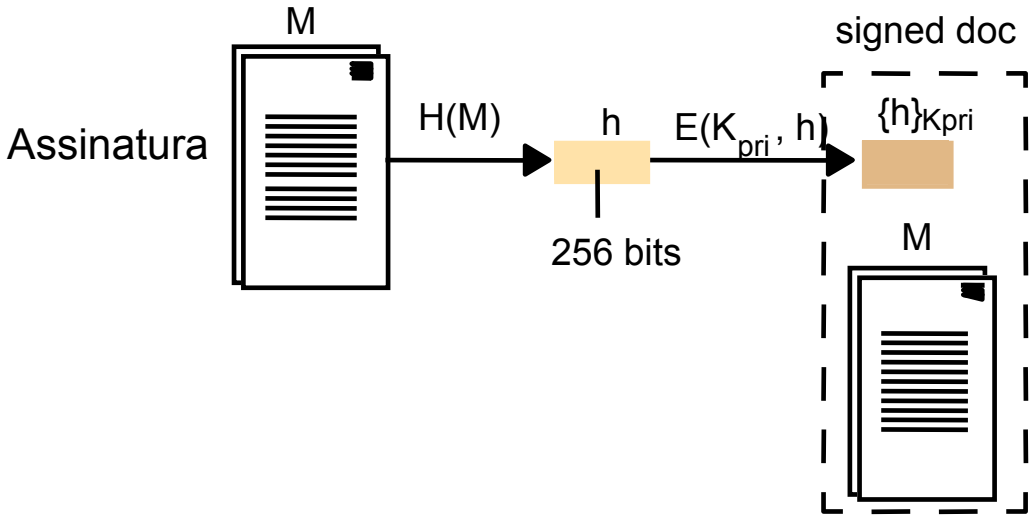
ASSINATURA DIGITAL: SOLUÇÃO COM CHAVES PÚBLICAS

Assinatura digital da mensagem M efectuada por Bob: $\{H(M)\}K_{Bpriv}$

Mensagem a enviar: $M,\{H(M)\}K_{Bpriv}$

H(M): função de síntese segura

UTILIZAÇÃO DE ASSINATURAS DIGITAIS COM CHAVES PÚBLICAS

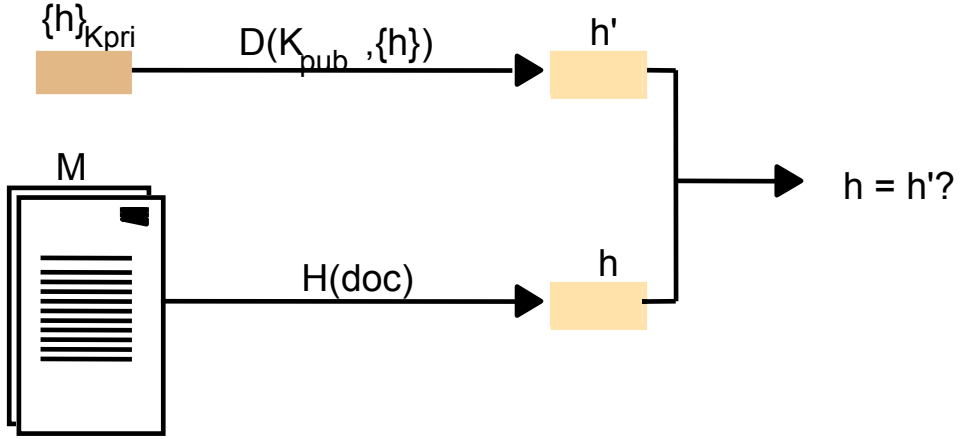


Bob envia uma mensagem com uma assinatura digital:

$M,\{H(M)\}K_{Bpriv}$

Alice verifica a assinatura e a integridade da mensagem:

$H(M) = \{H(M)\}K_{bpriv}\}K_{Bpub}$?



ASSINATURA DIGITAL: SOLUÇÃO COM CHAVES PÚBLICAS

Assinatura digital da mensagem M efectuada por Bob: {H(M)}KBpriv

Mensagem a enviar: M,{H(M)}KBpriv

H(M): função de síntese segura

Propriedades:

Verificação ao receber M',{H(M)}KBpriv : H(M') = {{H(M)}KBpriv}KBpub

Autenticação, integridade e não repudiamento garantidos porque apenas B consegue criar {H(M)}KBpriv e ninguém consegue alterar M para M' de forma que H(M')=H(M)

Dimensão da assinatura pequena e constante

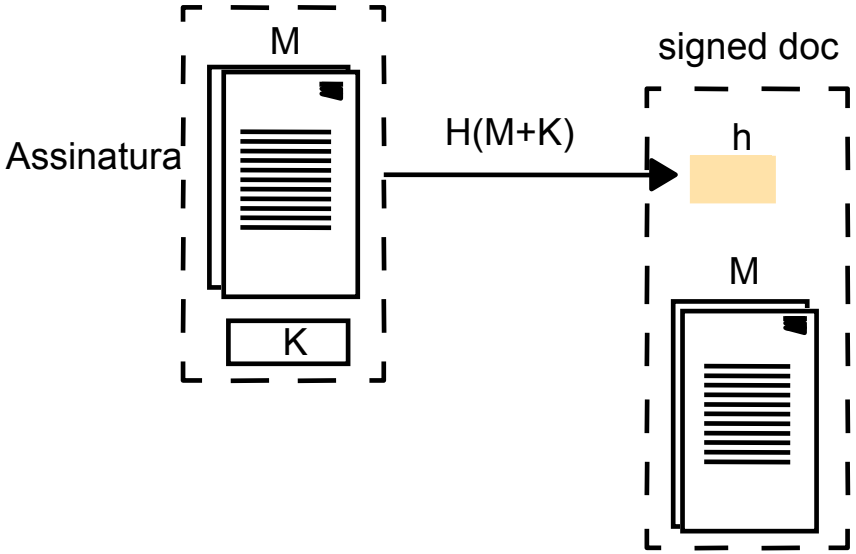
Apenas assinatura é cifrada – mensagem pode passar em claro

ASSINATURA DIGITAL – SOLUÇÃO COM CHAVES SIMÉTRICAS (MACS)

Assinatura digital da mensagem M efectuada por Bob: H(M+K)

- Mensagem a enviar: M,H(M+K)
- H(M): função de síntese segura

UTILIZAÇÃO DE ASSINATURAS DIGITAIS COM CHAVES SIMÉTRICAS (MACS)

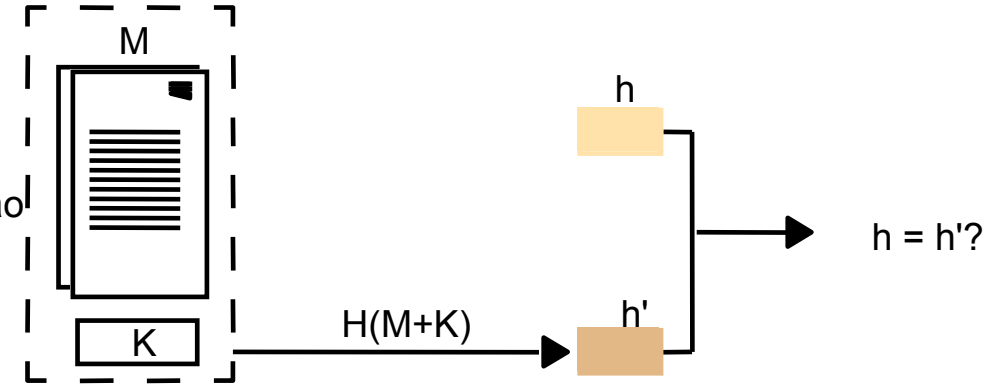


Bob envia uma mensagem com uma assinatura digital:

M, H(M+K)

Alice verifica a assinatura e a integridade da mensagem:

H(M+K) recebido = H(M+K) computado ?



Assinatura digital – solução com chaves simétricas (MACs)

Assinatura digital da mensagem M efectuada por Bob: h=H(M+K)

Mensagem a enviar: M,h

H(M): função de síntese segura

Propriedades:

Verificação ao receber M', h' : H(M'+K) = h'

Autenticação, integridade e não repudiamento garantidos porque apenas B consegue criar H(M+K) e ninguém consegue alterar M para M' de forma que H(M'+K)=H(M+K)

CERTIFICADOS DE CHAVES PÚBLICAS

A segurança baseada em chaves públicas depende da validade das chaves públicas.

Objectivo: um certificado deve permitir estabelecer a autenticidade de uma chave pública (por declaração de uma entidade fidedigna)

Certificado contém assinatura relativa, pelo menos, à informação do nome e chave pública da entidade. Exemplo:

1. *Certificate type*	**Public key**
2. *Name*	**Bob's Bank**
3. *Public key*	K_{Bpub}
4. *Certifying authority*	**Fred – The Bankers Federation**
5. *Signature*	$\{Digest(field\ 2 + field\ 3)\}_{K_{Fpriv}}$

AUTORIDADES DE CERTIFICAÇÃO

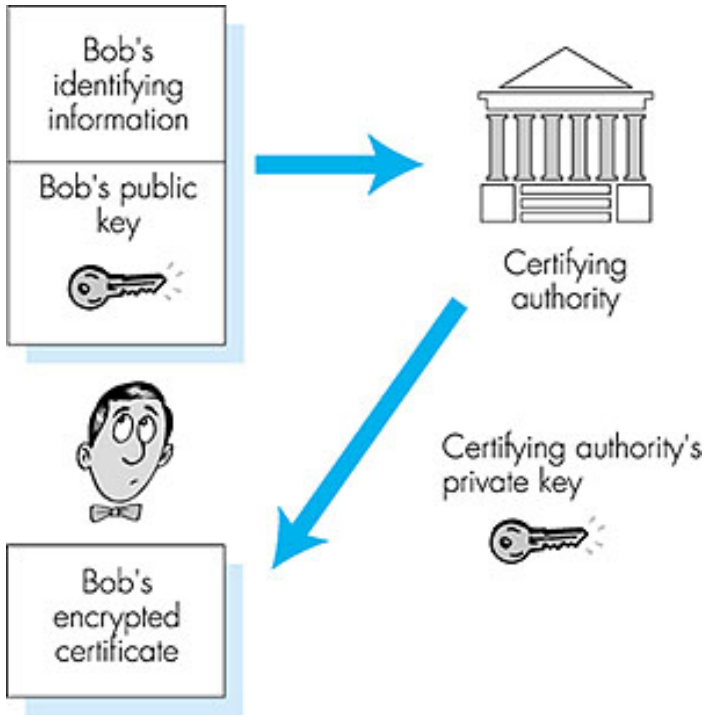
Uma *certification authority* (CA) associa chaves públicas a principais

- Em geral, as chaves públicas (certificados) das CAs são bem-conhecidos

Quando a Alice quer verificar qual a chave do Bob:

1) obtém um certificado dessa chave (Bob pode enviá-lo !).

2) verifica autenticidade do certificado verificando a assinatura efectuada pela CA (usando chave pública no certificado da CA)



CERTIFICADOS X.509

Entidade	Nome identificativo, chave pública
Emissor	Nome identificativo, assinatura
Período de validade	Data de início, data de fim
Informação administrativa	Versão, número de série
Informação adicional	

REPUDIAMENTO E VALIDADE DE UMA CHAVE

Quando uma chave privada é comprometida é impossível garantir a autenticidade de qualquer mensagem

Por isso, quanto antes, é necessário revogar a chave pública que lhe estava associada.

No caso dos certificados de chave pública, é necessário que as entidades de certificação memorizem os certificados válidos e revogados.

Como os certificados expiram ou podem ser revogados, é fundamental verificar o seu estado antes de confiar na informação.

A utilização de certificados, por si só, não é uma panaceia universal.

Algoritmo de Diffie-Hellman

O Algoritmo de Diffie-Hellman (e Merkle) permite negociar uma chave de sessão, evitando que esta passe em claro na rede.

Não requer segredos partilhados previamente.

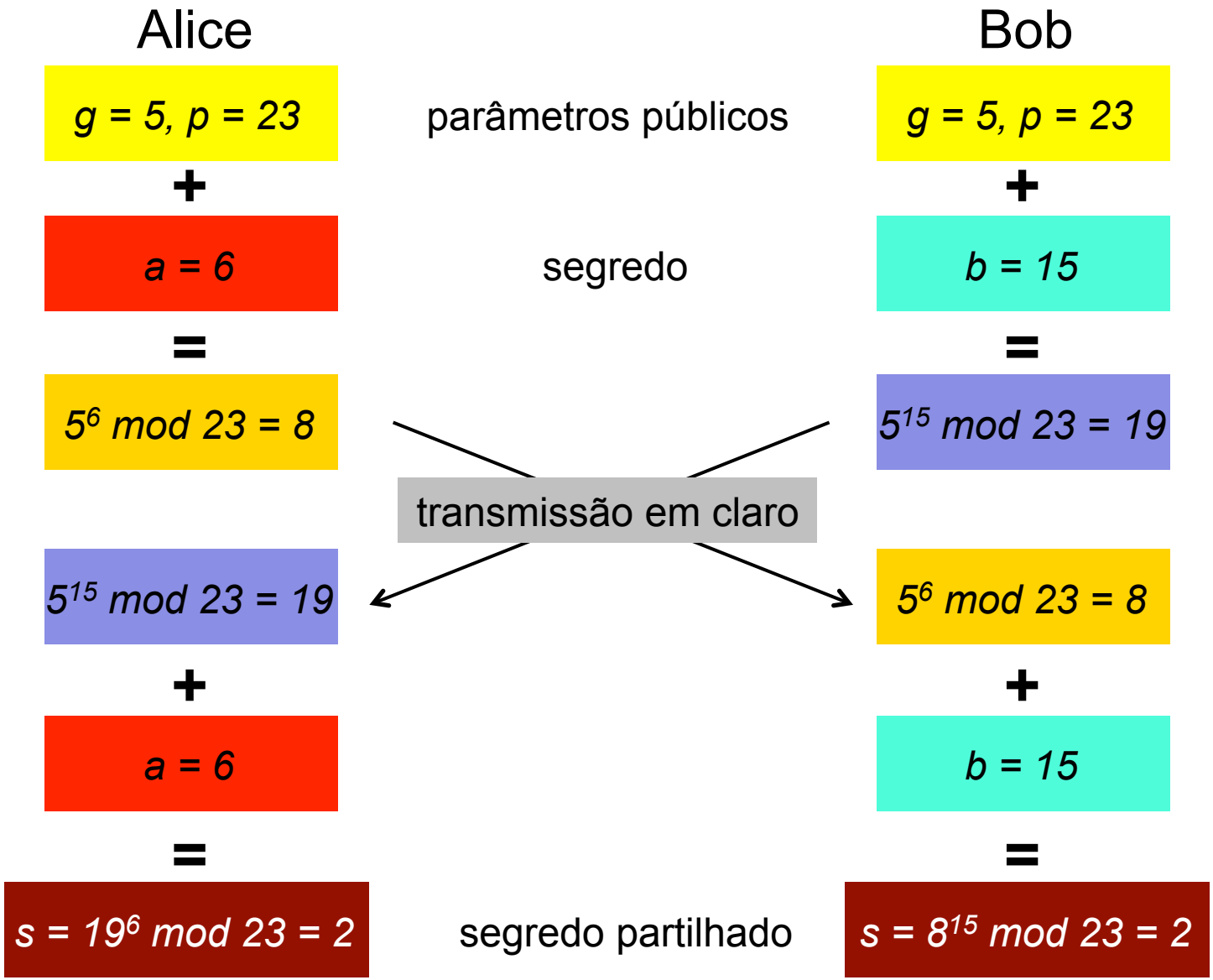
Usa segredos temporários, descartáveis, obtidos no momento.

A versão original tem como base o facto da exponenciação em aritmética modular ser comutativa:

$$(g^a \bmod p)^b \bmod p = (g^b \bmod p)^a \bmod p$$

e a dificuldade em determinar *x*, sabendo *g, p* e $g^x \bmod p$

ALGORITMO DE DIFFIE-HELLMAN :



SEGURANÇA FUTURA PERFEITA

Um protocolo de distribuição de chaves de sessão diz-se que garante **segurança futura perfeita** se não envolve segredos de longa duração que, uma vez comprometidos no futuro, permitiriam conhecer uma sessão do passado.

Nos algoritmos NS (e similares), se uma chave secreta/privada for comprometida é possível obter as chaves secretas negociadas com base nessa chave para uso nas comunicações

- Trudy pode ficar a conhecer o conteúdo das mensagens trocadas para todas as sessões que tenha gravado
- Os segredos são de longa duração

No algoritmo Diffie-Hellman, se a Alice e o Bob usarem valores aleatórios de Xa e Xb, que não voltem a ser usados, a chave é única e foi obtida sem recurso a segredos de longa duração visto que o algoritmo, m e n são públicos.

- Xa, Xb, Ka, Kb são segredos apenas enquanto dura a sessão: depois podem (devem) ser descartados

CASOS DE ESTUDO

TLS/SSL

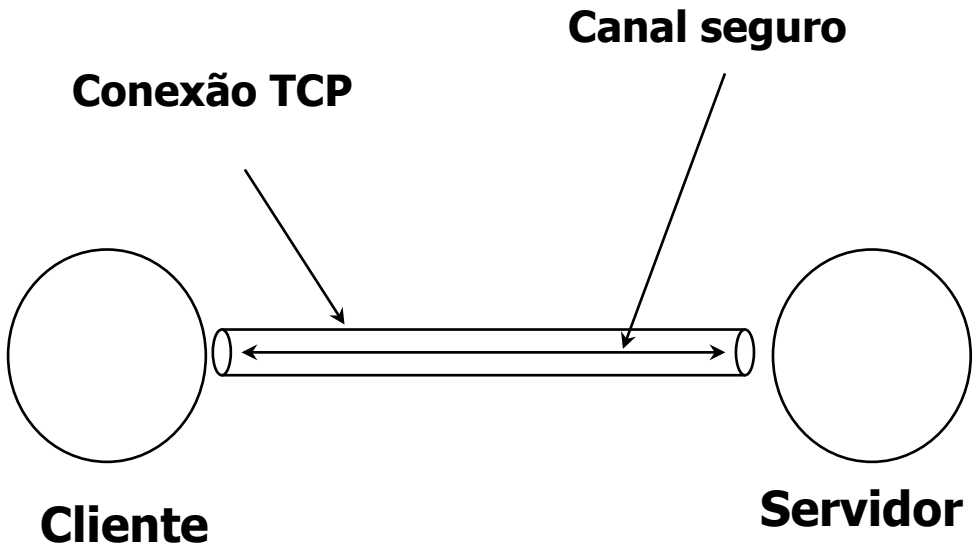
Oauth

TLS/SSL

TRANSPORT LAYER SECURITY

SSL - SECURE SOCKET LAYER

Trata-se de um protocolo proposto e desenvolvido pela Netscape, do tipo sessão, isto é sobre o nível transporte, que permite estabelecer canais seguros e autenticar clientes e servidores. Hoje em dia trata-se de uma norma IETF.



O PROTOCOLO SSL

O protocolo SSL funciona por cima do nível de transporte e permite fornecer segurança a qualquer aplicação baseada em TCP

É usado entre browsers e servidores WWW (https).

Funcionalidades de segurança:

- autenticação do servidor
- cifra dos dados
- autenticação do cliente (opcional)

Autenticação do servidor:

- O cliente (browser) inclui as chaves públicas de várias CAs
- O cliente solicita ao servidor um certificado do servidor emitido por uma CA em que ele confie.
- O cliente verifica o certificado do servidor com a chave pública da CA

Autenticação do cliente:

- Processa-se de forma semelhante

Veja no seu browser na secção de segurança.

SSL vs TLS

Os termos TLS e SSL são muitas vezes usados de forma indiferenciada (informalmente) para designar comunicação baseada em canais seguros)

Não existe uma só versão de SSL ou TLS.

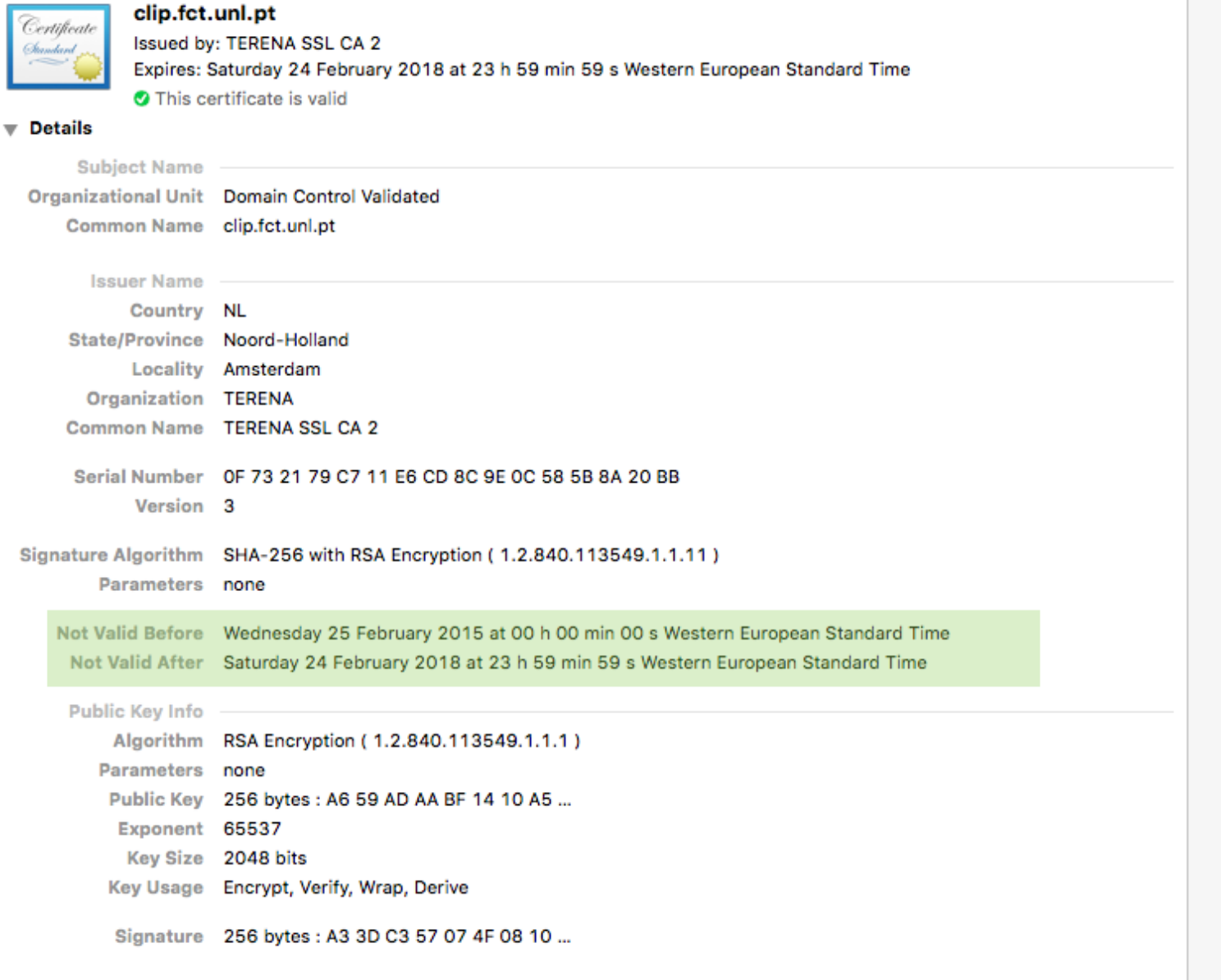
A versão 3.0 é a última versão SSL e é considerada obsoleta.

O protocolo TLS (TLS v1.0 ~ SSL 3.1) substitui o protocolo SSL

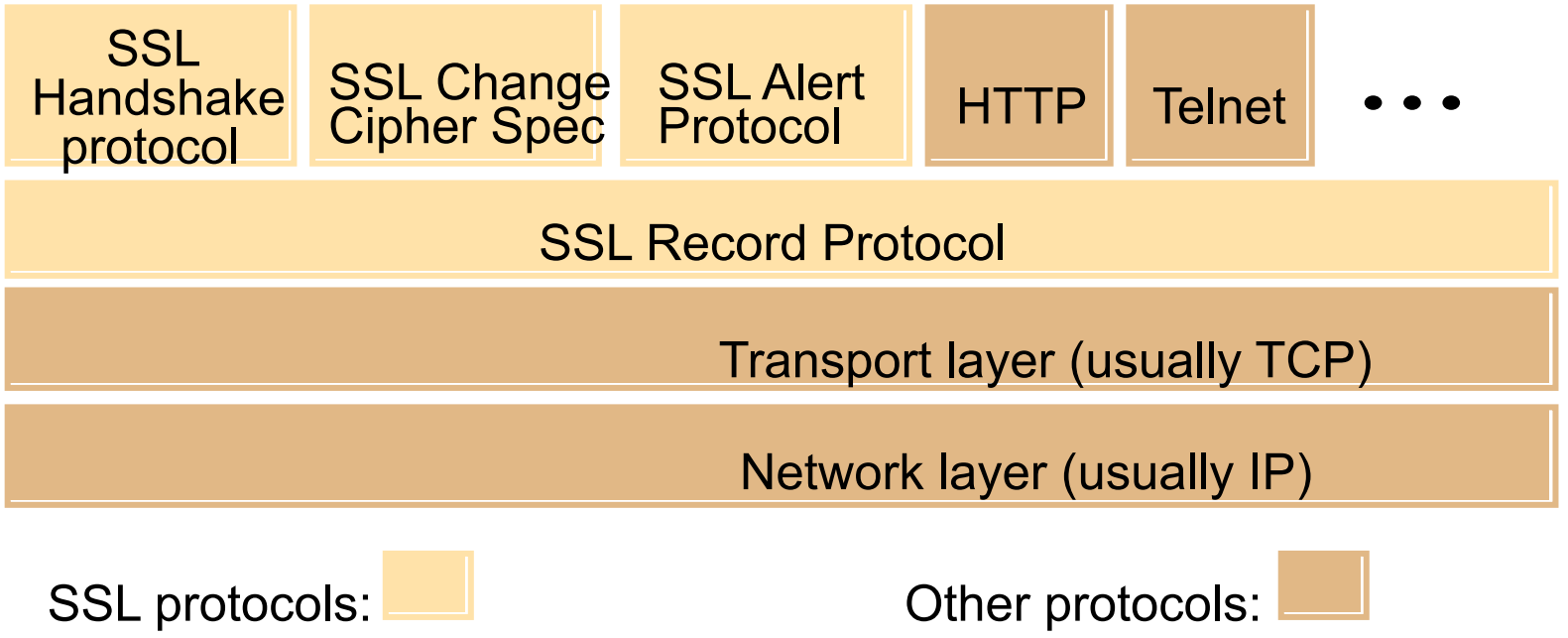
v1.2 em uso, v1.3 (em desenvolvimento)

A diferença entre SSL e TLS está no modo de negociação das chaves de sessão e não no tipo de chaves (e criptografia usada) que pode ser a mesma.

CERTIFICADO CLIP



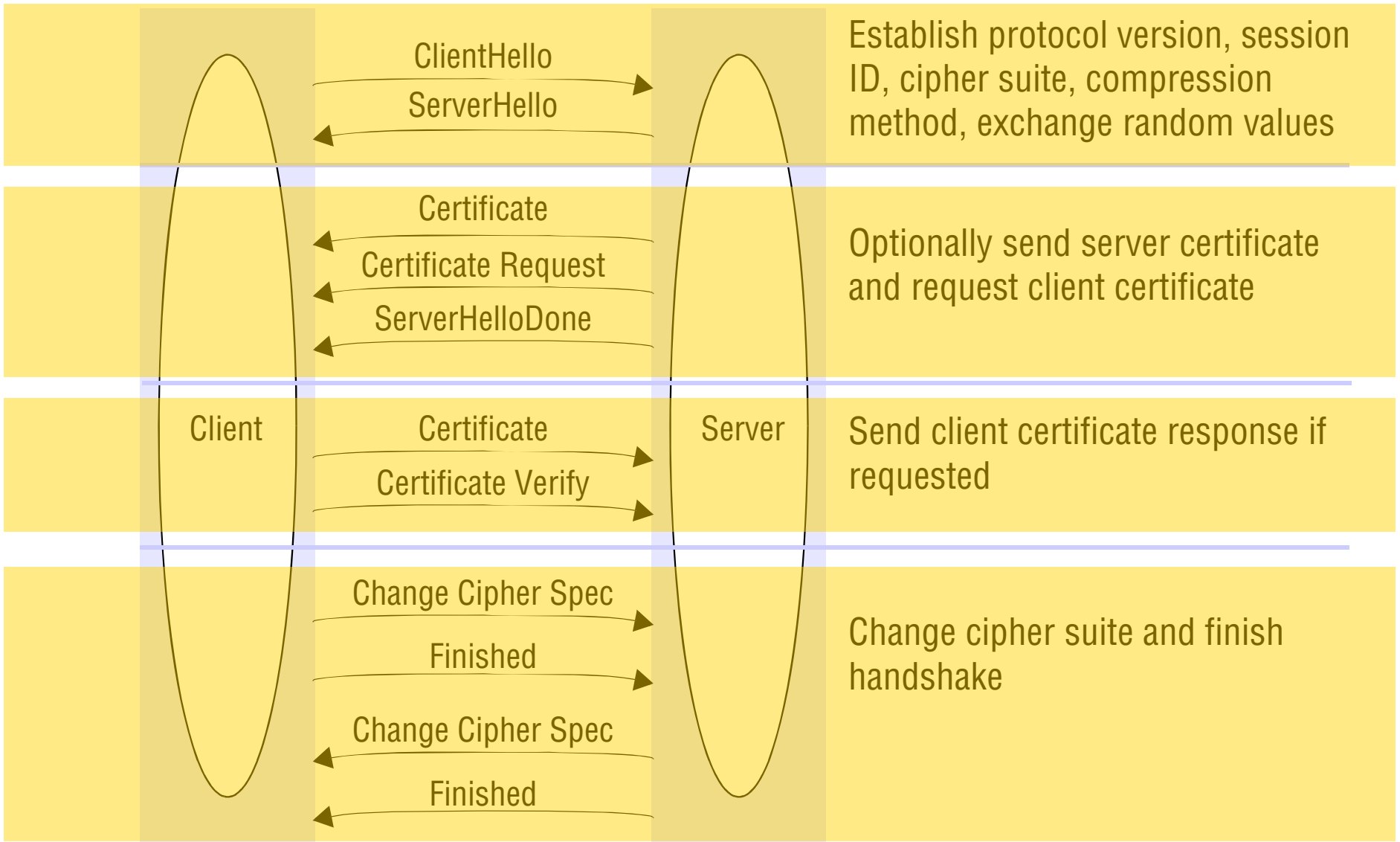
PROTOCOLOS SSL/TLS



SSL Record Protocol: implementa canal seguro, cifrando e autenticando as mensagens

SSL Handshake protocol + ***SSL CCS*** + ***SSL AP***: estabelecem e mantém um canal seguro entre um cliente e um servidor

SSL HANDSHAKE PROTOCOL



SSL HANDSHAKE CONFIGURATION OPTIONS

Component	*Description*	*Example*
Key exchange method	the method to be used for exchange of a session key	RSA with public-key certificates
Cipher for data transfer	the block or stream cipher to be used for data	IDEA
Message digest function	for creating message authentication codes (MACs)	SHA

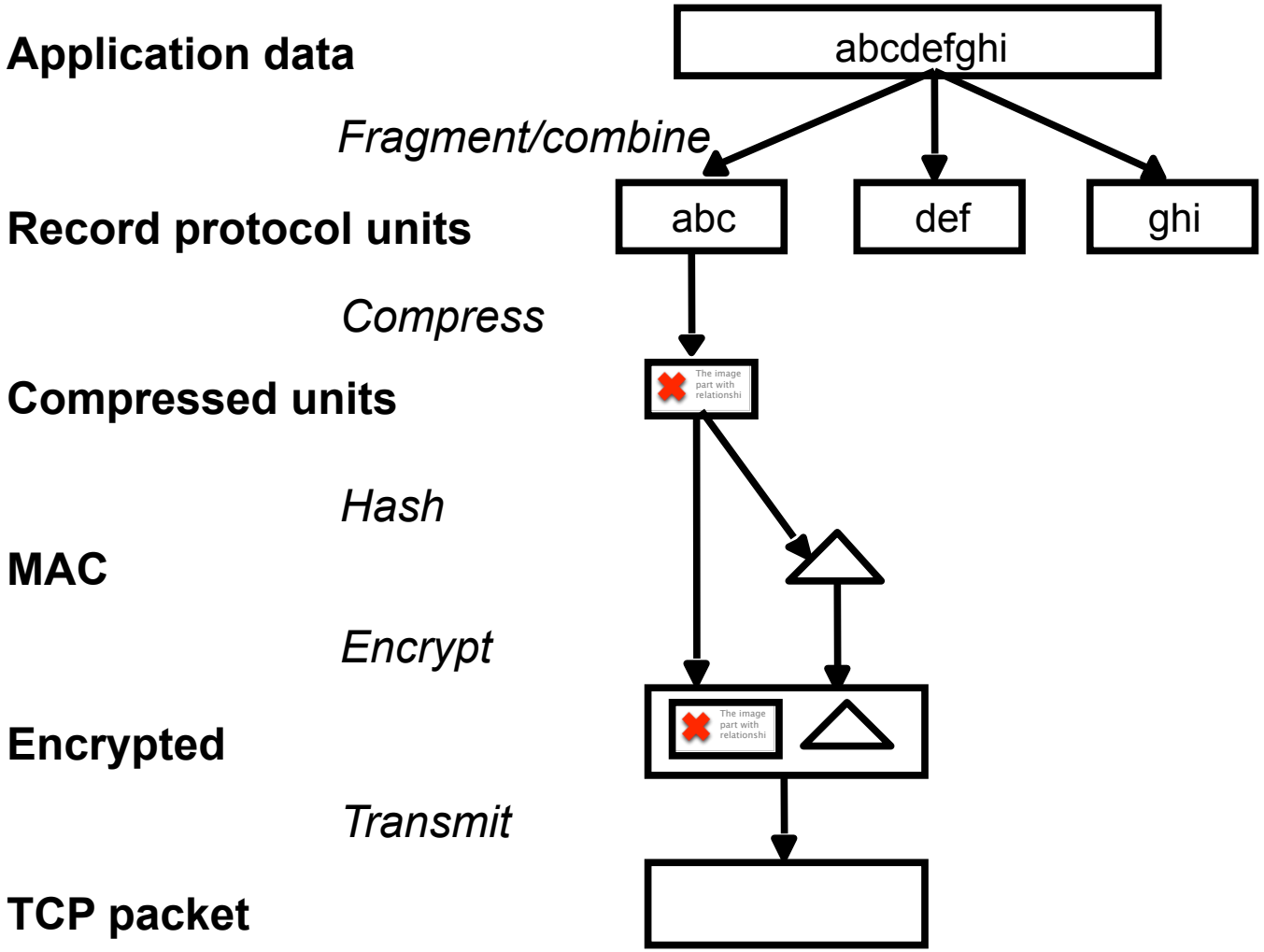
SSL permite usar diferentes *Cipher Suites*. Durante o *handshake* o servidor indica quais estão disponíveis e o cliente selecciona um.

Autenticação do servidor: servidor envia certificado e cliente envia *premaster secret* cifrado com a chave pública do servidor

> *Premaster secret* é usado para gerar as chaves de cifra (uma para cada sentido) e a chave a usar no MAC

Autenticação do cliente: cliente envia certificado e cliente envia assinatura de parte das mensagens trocadas

SSL RECORD PROTOCOL



OAuth

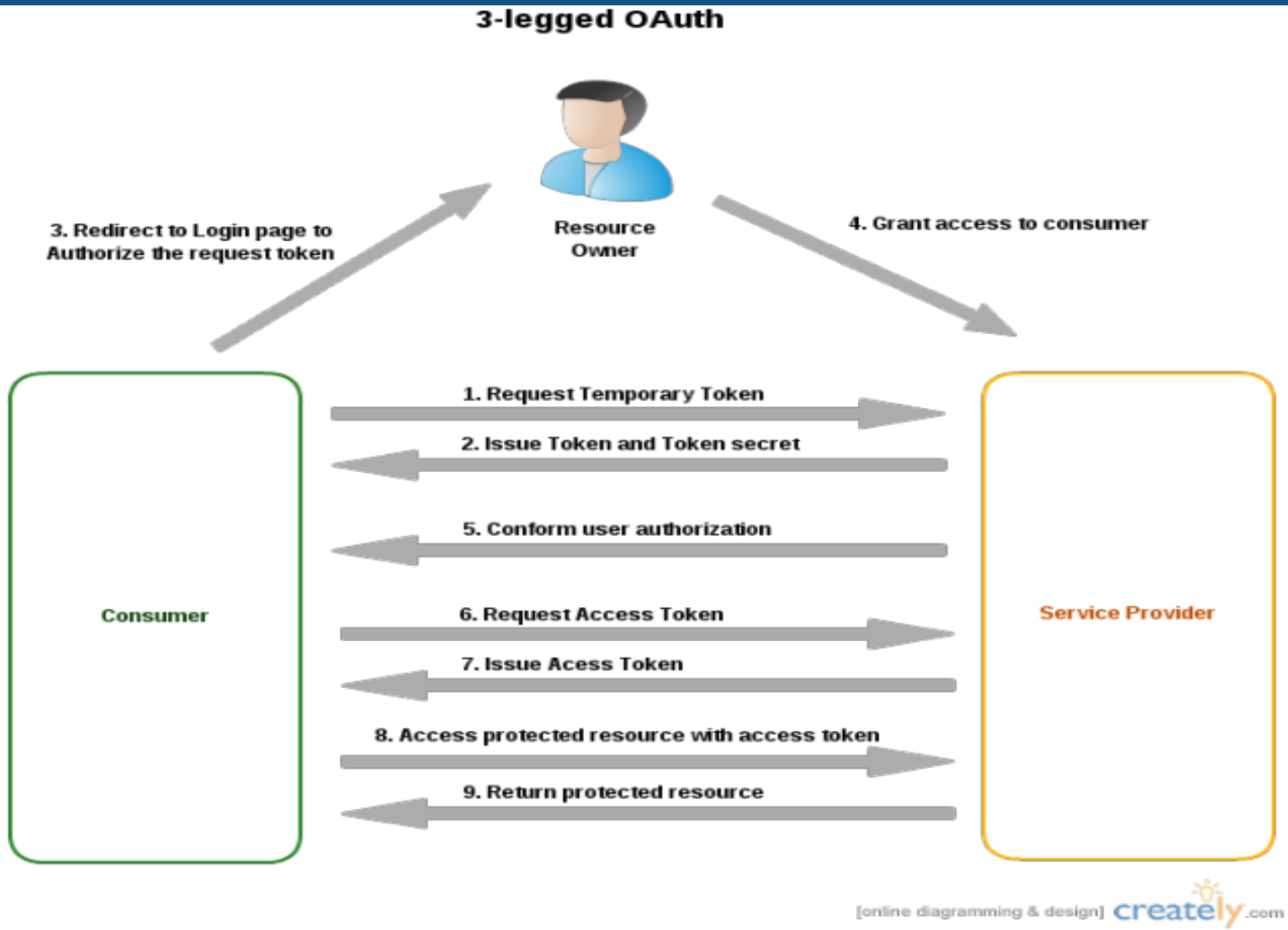
OAUTH: OBJETIVO

Permitir a **um cliente** aceder a **um recurso num servidor** em **nome de um utilizador**

Para que é que isto serve?

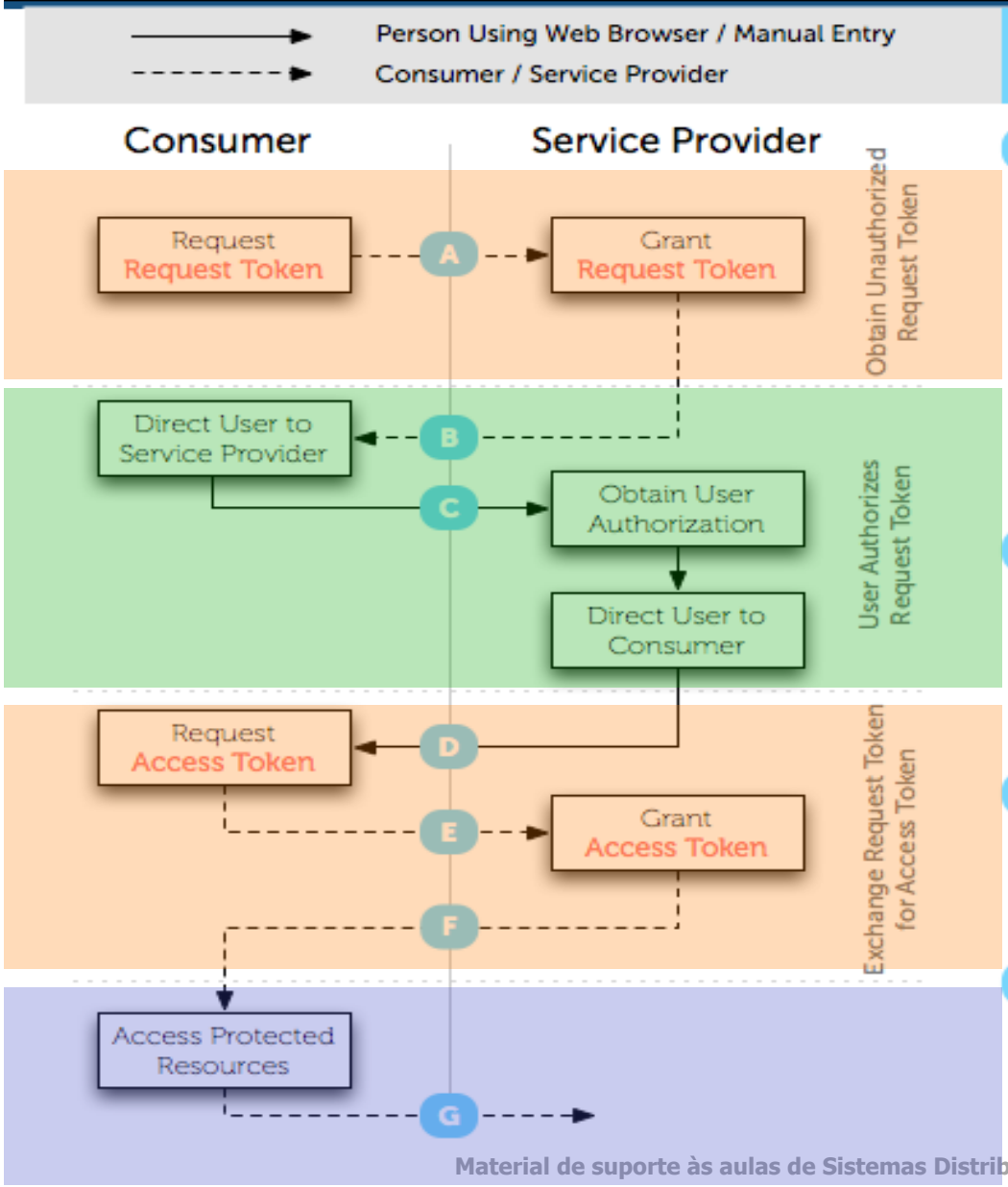
Porque é que o cliente não faz a autenticação diretamente?

OAUTH: 3 LEGGED OAUTH



src: https://malalanayake.wordpress.com/2013/01/09/3-legged-oauth-flow/

OAuth



OAUTH AUTHENTICATION FLOW v1.0a

A Consumer Requests Request Token

Request includes
oauth_consumer_key
oauth_signature_method
oauth_signature
oauth_timestamp
oauth_nonce
oauth_version (optional)
oauth_callback

B Service Provider Grants Request Token

Response includes
oauth_token
oauth_token_secret
oauth_callback_confirmed

C Consumer Directs User to Service Provider

Request includes
oauth_token (optional)

D Service Provider Directs User to Consumer

Request includes
oauth_token
oauth_verifier

E Consumer Requests Access Token

Request includes
oauth_consumer_key
oauth_token
oauth_signature_method
oauth_signature
oauth_timestamp
oauth_nonce
oauth_version (optional)
oauth_verifier

F Service Provider Grants Access Token

Response includes
oauth_token
oauth_token_secret

G Consumer Accesses Protected Resources

Request includes
oauth_consumer_key
oauth_token
oauth_signature_method
oauth_signature
oauth_timestamp
oauth_nonce
oauth_version (optional)

PROTOCOLO: PASSO 0: REGISTO

Para aceder a recurso, deve-se obter chave de acesso:

Application key : <app-key>
Application secret : <app-secret>

Protocolo: passo 1: obter Request Token

Aplicação cliente pede request token

Assinatura: HMAC-SHA1, RSA-SHA1, Plaintext

POST /request_temp_credentials HTTP/1.1

Host: server.example.com

Authorization: OAuth realm="Example",

oauth_consumer_key=<app-key>,

oauth_signature_method="PLAINTEXT",

oauth_callback="<client-callback-url>",

oauth_signature="<app-secret>&"

Request Token = token + segredo

HTTP/1.1 200 OK

Content-Type: application/x-www-form-urlencoded

oauth_token=<request-token>&oauth_token_secret=<request-token-secret>&
oauth_callback_confirmed=true

PROTOCOLO: PASSO 2: AUTORIZAÇÃO DO CLIENTE

Utilizador acede a URL para fornecer autorização, passando o token recebido no passo 1

Servidor regista autorização
(opcional) Servidor disponibiliza código de verificação <verifier>
(opcional) Servidor envia pedido ao endereço de callback

GET /authorize_access?oauth_token=<request token>
&oauth_callback=<client-callback-url> HTTP/1.1
Host: server.example.com

(chamada a callback)
GET /cb?x=1&oauth_token=<request-token>&oauth_verifier=<verifier>HTTP/1.1
Host: client.example.net

PROTOCOLO: PASSO 3: OBTER ACCESS TOKEN

Aplicação cliente pede access token

Envia: request token e código de verificação

Recebe token de acesso: token + segredo

```
POST /access_token HTTP/1.1
    Host: server.example.com
    Authorization: OAuth realm="Example",
      oauth_consumer_key=”<app-key>",
      oauth_token=”<request-token>",
      oauth_signature_method="PLAINTEXT”,
      oauth_signature=”<app-secret>&<request-token-secret>”

HTTP/1.1 200 OK
    Content-Type: application/x-www-form-urlencoded
    oauth_token=<access-token>&oauth_token_secret=<access-token-secret>
```

PROTOCOLO: PASSO 4: ACESSO AOS RECURSOS

Aplicação cliente acede a recurso usando token de acesso

```
POST /request?b5=%3D%253D&a3=a&c%40=&a2=r%20b HTTP/1.1
    Host: example.com
    Content-Type: application/x-www-form-urlencoded
    Authorization: OAuth realm="Example”
    oauth_consumer_key=”<app-key>",
                oauth_token=”<access-token>",
                oauth_signature_method="HMAC-SHA1",
                oauth_timestamp="137131201",
                oauth_nonce="7d8f3e4a",
                oauth_signature=HMAC(”<app-secret>&<access-token-secret>”)
```

PARA SABER MAIS

Bibliografia base

G. Coulouris, J. Dollimore and T. Kindberg, Distributed Systems - Concepts and Design, Addison-Wesley, 4th Edition, 2005

Capítulo 7

Bibliografia adicional:

James F. Kurose and Keith W. Ross, "Computer Networking - A Top-Down Approach Featuring the Internet," Addison Wesley Longman, Inc., Second Edition, 2003 – capítulo 7.

Tanenbaum and Maarten van Steen "Distributed Systems – Principles and Paradigms," Prentice-Hall, 2002 – capítulo 8.

W. Stallings, "Cryptography and Network Security – Principles and Practice," Second Edition 1999 – Este livro é integralmente dedicado a este tópico.

PROBLEMA 1:

Três entidades: Clientes (C), S1, S2

- S1 tem KpubS1/KprivS1. Clientes conhecem KpubS1
- S1 conhece nome,pwd de cada cliente
- S1 e S2 partilham Ks

C quer executar operação em S2

- C envia op para S2
- S2 envia res para C

Propriedades: secretismo, integridade, autenticação

PROBLEMA 2:

Três entidades: Clientes (C), S1, S2

- C partilha Ks com S1
- C partilha Ks2 com S2

C quer executar operação com duas partes, uma em S1 e outra em S2

Protocolo com 3 mensagens C->S1; S1->S2 ; S2->C

- C envia M1 para S1 e M2 para S2;
- S1 envia R1 para S2
- S2 envia R2 para C

Propriedades: secretismo, integridade, autenticação

PROBLEMA 3

Três entidades: Clientes (C), AP, S

- S conhece user/pwd de C
- S tem KpubS/KprivS. C conhece KpubS
- AP e S partilham Ks

C e AP querem negociar chave de sessão

Protocolo com 4 mensagens C->AP; AP->S ; S->AP; AP->C

Propriedades: secretismo, integridade, autenticação