Computação Distribuída

Cap VI – Segurança

Licenciatura em Engenharia Informática

Universidade Lusófona

Prof. Paulo Guedes (paulo.guedes@ulusofona.pt)

Politicas de Segurança

- Quando é que se torna necessário uma política de segurança ?
  - Quando um bem com *valor* é colocado *num espaço partilhado* criam-se condições para que o bem possa ser alvo de ameaças
- Uma política de segurança procura garantir a protecção do bem dentro de determinados condicionantes.
- Uma política de segurança define-se respondendo às seguintes questões:
  - O que queremos proteger ?
  - Quais as ameaças potenciais ?
  - Como é que os ataques correspondentes às ameaças se podem materializar ?
  - Quais os procedimentos e mecanismos de protecção que podem impedir os ataques considerados ?
  - Qual o custo de implementação da política ?

Politicas de Segurança nos Sistemas Informáticos

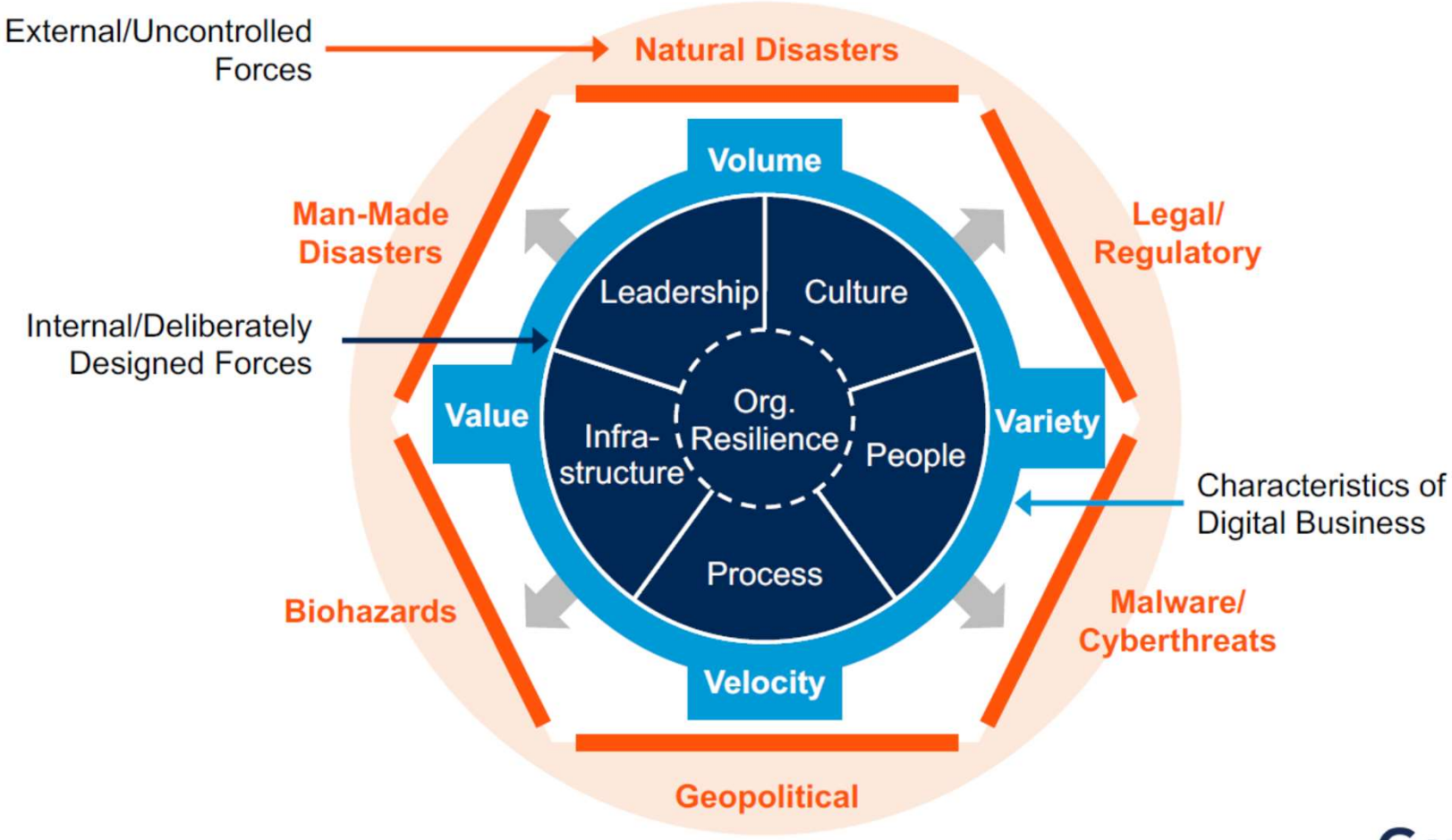
- Ameaças
  - Divulgação da informação
  - Modificação da informação
  - Execução de operações não autorizadas
  - Destruição de informação
  - Impedir o acesso a informação
- Motivações para os ataques
  - Reconhecimento, fama
  - “Diversão”, vandalismo
  - Bullying
  - Criminalidade (roubo, extorsão)
  - Ataque comercial
  - Ataque político

Cybersecurity in the 2010s

5 © 2020 Gartner, Inc. and/or its affiliates. All rights reserved. Gartner is a registered trademark of Gartner, Inc. and its affiliates.

Gartner

Cybersecurity as an enabler of organizational resiliency



15 © 2020 Gartner, Inc. and/or its affiliates. All rights reserved. Gartner is a registered trademark of Gartner, Inc. and its affiliates.

Gartner

Politicas de Segurança nos Sistemas Informáticos

- Isolamento dos Agentes
- Controlo dos Direitos de Acesso
- Controlo do Nível de informação

Isolamento dos Agentes

- Agentes: elementos activos no sistema (aplicações). Um agente representa no sistema um utilizador humano
- Autenticação: estabelecimento de forma segura da relação entre o utilizador humano e o agente que o representa no sistema
- O sistema garante que ao agente é atribuída uma máquina virtual em que ele executa as operações sobre os objectos em completo isolamento das máquinas virtuais atribuídas a outros agentes. Exemplo de máquina virtual: canal de comunicação privado.
- O isolamento implica que não existem meios para um agente ultrapassar os mecanismos de confinamento ou enviar informação através de canais encobertos

Controlo de direitos de acesso

DAC – Discretionary Access Control

- Os objectos são protegidos por um *monitor de segurança.*
- O agente (aplicação), antes de efectuar um *acção sobre um objecto*, pede autorização ao monitor
- O monitor verifica se o agente está ou não autorizado através de uma *matriz de direitos acesso*

Agentes / Objectos	Obj 1	Obj 2	Obj 3	Obj 4
Agente 1	R	RW	RX	---
Agente 2	RX	---	RW	R

- Construção da tabela
  - Listas de controlo de acesso (*Access Control Lists*, ACLs). Guardadas junto de cada objecto. Lista de pares (agente, direitos de acesso)
  - Capacidades (*capabilities*). Guardadas junto de cada agente. Lista de (objecto, direitos de acesso)
- Modelo habitual de controlo de segurança
- Os agentes (aplicações) decidem onde e como efectuar as verificações de segurança. O controlo é discricionário, fica à consideração da aplicação efectuar ou não esse controlo
- Falha mais habitual desta política: incorrecta verificação pelo agente do controlo do monitor de segurança.

Controlo do Nível de Segurança da Informação

MAC – Mandatory Access Control

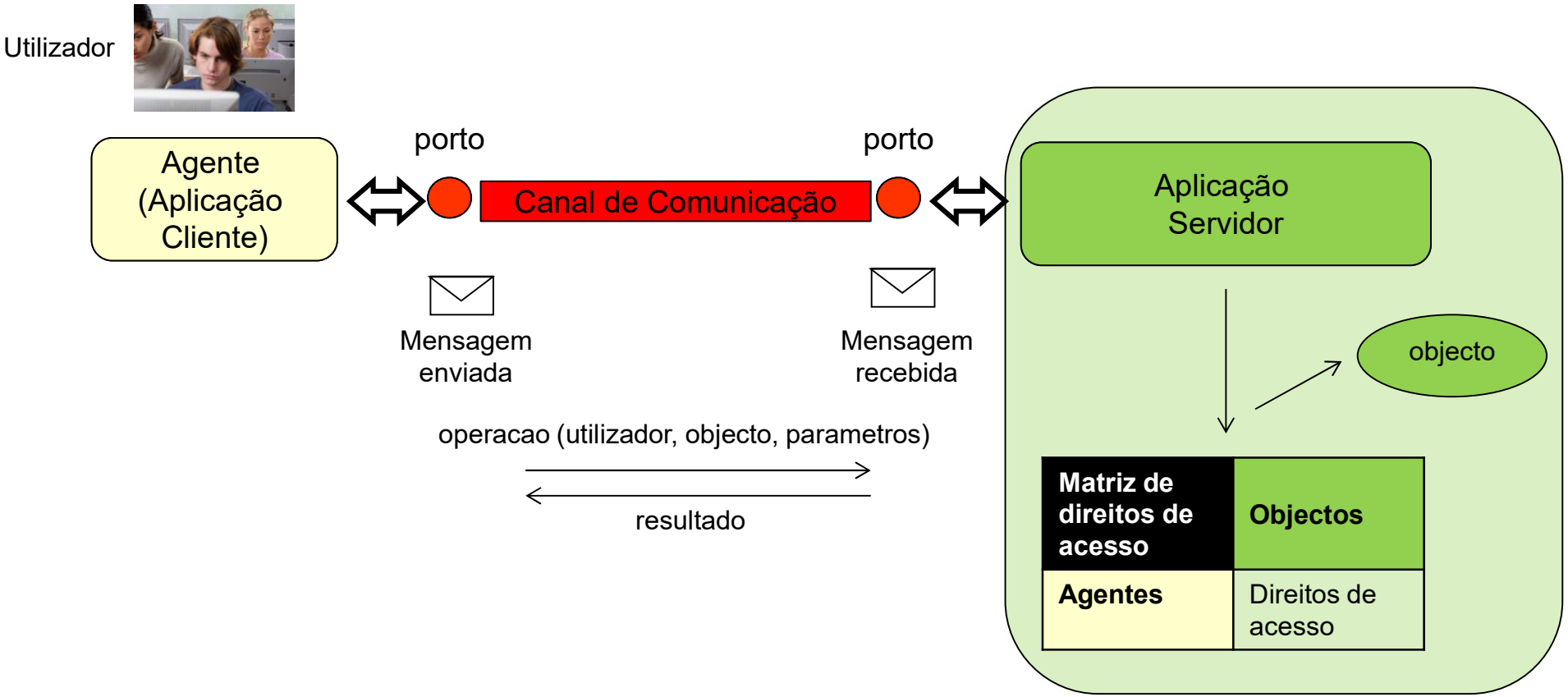
- *O sistema efectua sempre, de forma mandatória, um controlo de acesso aos objectos*
- Política oriunda da visão militar da segurança

- No MAC são definidos dois conceitos:
  - Níveis de confidencialidade da informação: determinam o grau de confidencialidade da informação. Ex: Não Classificado, Restrito, Secreto, Muito Secreto.
  - Níveis de acesso (clerance levels) dos agentes: definem o nível de confidencialidade da informação a que os agentes podem aceder
- Objectos e agentes são associados a níveis de confidencialidade e compartimentos:
  - Compartimentos (folders): classificam a informação por temas. Ex.: Notas de alunos, Salários de colaboradores
  - Organização da informação (objectos) em compartimentos
  - Classificação da informação (objectos) em Restrito, Secreto, Muito secreto
- Controlo de acesso dos agentes
  - Um agente só tem acesso a informação se realmente tiver necessidade de conhecer (need to know), aplicando o sistema regras estritas para determinar quem tem acesso a quê.
  - Só podem ler informação classificada no seu nível de acesso e nos níveis abaixo
  - Só podem escrever informação classificada no seu nível de acesso e nos níveis acima

Perguntas a que devo ser capaz de responder

- Porque razão a segurança informática é tão importante ?
- O que é uma política de segurança ?
- O que significa o isolamento dos agentes ?
- O que é uma Access Control List no modelo DAC ?
- Considerando uma aplicação cliente-servidor a funcionar no modelo DAC, onde e como têm que ser colocados as verificações de segurança ?
- No modelo MAC, quem faz as verificações de segurança ?
- Qual é a relação entre a classificação da informação e os níveis de acesso dos agentes ?

Modelo de Segurança num Sistema Distribuído



Requisitos e ataques à segurança

Requisitos de segurança:

- Privacidade: acesso à informação apenas por agentes autorizados
- Integridade: a informação está correcta e completa
- Autenticidade: identificação do agente que produziu a informação
- Não repúdio: impossibilidade de um agente negar uma acção que efectivamente cometeu

Ataques à segurança:

- Passivos:
  - Escuta de mensagens
- Activos
  - Interferência com o fluxo de mensagens (modificação, inserção, remoção, troca de ordem de mensagens)
  - Repetição de diálogos passados
  - Falsificação de identidades
  - Man-in-the-middle
  - Denial of service

Perguntas a que devo ser capaz de responder

- Quais são os requisitos de segurança de um sistema distribuído ?
- Quais são as ameaças à segurança um sistema distribuído ?
- Como é que cada ameaça pode comprometer os requisitos de segurança ?

Criptografia

- Criptografia: codificação de informação de forma a tornar o seu conteúdo incompreensível

Emissor	**Receptor**	
$E(K1, M) \Rightarrow \{M\}_{K1}$		Emissor cifra ("encrypt") a mensagem M com a chave K1
Enviar $(\{M\}_{K1}) \rightarrow$	Receber $(\{M\}_{K1})$	Emissor envia a mensagem cifrada ao Receptor
	$D(K2, \{M\}_{K1}) \Rightarrow M$	Receptor decifra ("decrypt") a mensagem com a chave K2

- Algoritmos de cifra: funções bijectivas definidas por
  - Um algoritmo de transformação de dados
  - Uma chave que altera essa transformação
- As cifras são *reversíveis* apenas por quem possuir o *algoritmo inverso*
  - Modelo inverso de transformação dos dados
  - Chave inversa
- Nomenclatura

  $M \rightarrow \{M\}_{K1}$: cifra da mensagem M com a chave K1.

  É gerado um criptograma

  $\{\{M\}_{K1}\}_{K2} \rightarrow M$: decifra do criptograma $\{M\}_{K1}$ com a chave K2.

  É obtida a mensagem original

Exemplo: algoritmo de cifra xOR

```
void encrypt(char k[], char text[])
{
      int n;
     for (n= 0; n < 8; n++)
     {
          text[n] = text[n] ^ k[n];
     }
}
```

```
void encrypt(char k[], char text[])
{
      int n;
     for (n= 0; n < 8; n++)
     {
          text[n] = text[n] ^ k[n];
     }
}
```

```
void cipher (FILE *infile, FILE *outfile, char k[]) {
     char text[8]; int i;
     while(!feof(infile)) {
          i = fread(text, 1, 8, infile);          /* read 8 bytes from infile */
          if (i <= 0) break;
          while (i < 8) { text[i++] = ' ';}       /* pad last block with spaces */
          encrypt(k, text);
          fwrite(text, 1, 8, outfile);            /* write 8 bytes to outfile */
     }
}
```

Exemplo: cifra com xOR

Mensagem original (em código binário): 1101 0101 1111 1011

Chave (4 bits): 1101

Cifrar:

Mensagem original	1101	0101	1111	1011
Chave	1101	1101	1101	1101
Mensagem cifrada	0000	1000	0010	0110

xOR

Decifrar:

Mensagem cifrada	0000	1000	0010	0110
Chave	1101	1101	1101	1101
Mensagem original	1101	0101	1111	1011

xOR

Exemplo: algoritmo TEA (função de cifra)

```
void encrypt(unsigned long k[], unsigned long text[]) {
    unsigned long y = text[0], z = text[1];                          1
    unsigned long delta = 0x9e3779b9, sum = 0; int n;                2
    for (n= 0; n < 32; n++) {                                        3
        sum += delta;                                                4
        y += ((z << 4) + k[0]) ^ (z+sum) ^ ((z >> 5) + k[1]);        5
        z += ((y << 4) + k[2]) ^ (y+sum) ^ ((y >> 5) + k[3]);        6
    }
    text[0] = y;  text[1] = z;                                       7
}
```


© Pearson Education 2012

Exemplo: algoritmo TEA (função de decifra)

```
void decrypt(unsigned long k[], unsigned long text[]) {
    unsigned long y = text[0], z = text[1];                          1
    unsigned long delta = 0x9e3779b9, sum = delta << 5;  int n;      2
    for (n= 0; n < 32; n++) {                                        3
        z -= ((y << 4) + k[2]) ^ (y + sum) ^ ((y >> 5) + k[3]);      4
        y -= ((z << 4) + k[0]) ^ (z + sum) ^ ((z >> 5) + k[1]);      5
        sum -= delta;                                                6
    }
    text[0] = y; text[1] = z;                                        7
}
```


© Pearson Education 2012

Criptografia: cifras simétricas e assimétricas

- Cifras simétricas: K1 = K2

Chave Secreta

Emissor	**Receptor**	
$E(K, M) \Rightarrow \{M\}_K$		Emissor cifra ("encrypt) a mensagem M com a chave K
Enviar $(\{M\}_K) \rightarrow$	Receber $(\{M\}_K)$	Emissor envia a mensagem cifrada ao Receptor
	$D(K, \{M\}_K) \Rightarrow M$	Receptor decifra ("decrypt") a mensagem com a chave K

  - A chave K tem que ser secreta: só pode ser conhecida pelo emissor e receptor
  - Normalmente usam técnicas de confusão e difusão. São muito mais rápidas que as assimétricas

- Cifras assimétricas: K1 ≠ K2

Chave Pública

  - Uma das chaves é pública e a outra privada
  - Normalmente usam operações matemáticas complexas. Ex:
    - Logaritmo modular
      - $Y = a^X \bmod b$; Dados a, b e Y, calcular X
    - Factorização de grandes números
      - Y = ab, a e b primos; Dado Y, calcular a ou b

Cifra por blocos: exemplos de ataques

Mensagem original	1101	0101	1111	1011
Chave	1101	1101	1101	1101
Mensagem cifrada	0000	1000	0010	0110

Repetição da mensagem

Ataque 1: repetição de mensagens anteriores

Mensagem cifrada	0000	1000	0010	0110

Ataque 2: repetição de blocos de mensagens anteriores

Mensagem original	1001	1110	0011	1000
Chave	1101	1101	1101	1101
Mensagem cifrada	0100	0011	1110	0101
Mensagem alterada	0000	**1000**	**0010**	0110

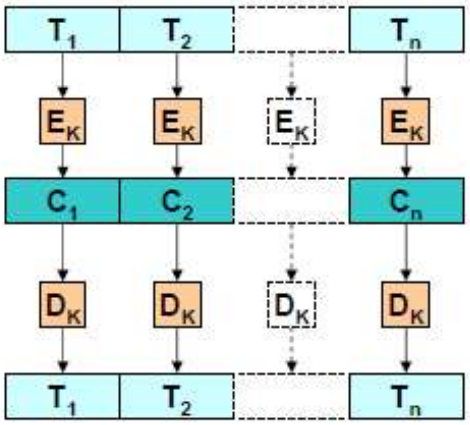
Repetição de blocos da mensagem

Cifra por blocos e cifra contínua

- Cifra por blocos
  - Um bloco cifrado é obtido pela cifra do bloco original
  - Não depende dos blocos anteriores nem dos seguintes. O mesmo valor, cifrado com a mesma chave, produz sempre o mesmo resultado
  - Algoritmo mais simples, facilita a análise
  - Susceptível de ataques de repetição de blocos
  - Exemplo: Electronic Code Book (ECB)

    $Ci = E_K(Ti)$

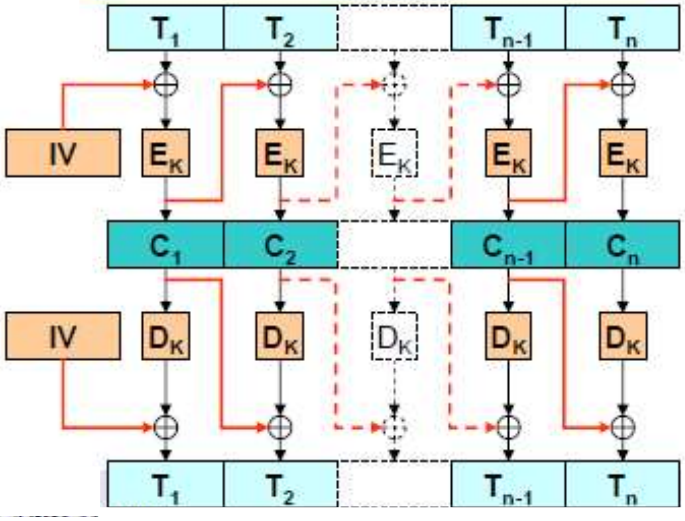
    $Ti = D_K(Ci)$



- Cifra Contínua
  - Um bloco cifrado depende da cifra do bloco original e dos blocos anteriores
  - Início da mensagem tem que conter um *initialization vector* com um valor diferente em cada mensagem (ex: timestamp)
  - Exemplo: Cipher Block Chaining

    $Ci = E_K(Ti \oplus Ci\text{-}1)$

    $Ti = D_K(Ci) \oplus Ci\text{-}1$



Utilização da criptografia

- Privacidade

  Informação cifrada $\{M\}_K$

  - Mesmo que o atacante veja a informação (escute as mensagens, aceda ao ficheiro), não entende o seu conteúdo

- Autenticidade

  Informação cifrada $\{M\}_K$

  - O facto da informação estar cifrada (e fazer sentido) garante que é autêntica: só quem conhece a chave a podia ter cifrado

- Integridade

  Informação assinada: calcula-se um resumo D de M (D - digest)

  Cifra-se o digest: M, $\{D\}_K$

  - O digest da mensagem garante
    - Integridade: a mensagem não foi alterada e o digest também não (por estar cifrado)
    - Autenticidade: o digest cifrado garante que foi gerado por quem conhece a chave – **assinatura digital**

- Ataques ao canal de comunicação
  - Criptografia não evita a reutilização das mensagens, denial of service, etc.

Perguntas a que devo ser capaz de responder

- O que é a criptografia ?
- É boa ideia a segurança de uma comunicação cifrada estar baseada no secretismo do algoritmo de cifra ? Porquê ? Qual é a solução correta ?
- O que é cifra simétrica e cifra assimétrica ? Qual é mais rápida ? Porquê ?
- Qual é a vulnerabilidade da cifra por blocos ?
- Como é que a cifra contínua ultrapassa a vulnerabilidade da cifra por blocos ?
- Como é possível garantir privacidade e autenticidade numa comunicação usando cifra simétrica ? E cifra assimétrica ?
- Como é construída e verificada uma assinatura digital ? O que é que ela garante ?

Autenticação

- **A autenticação é a base da segurança**

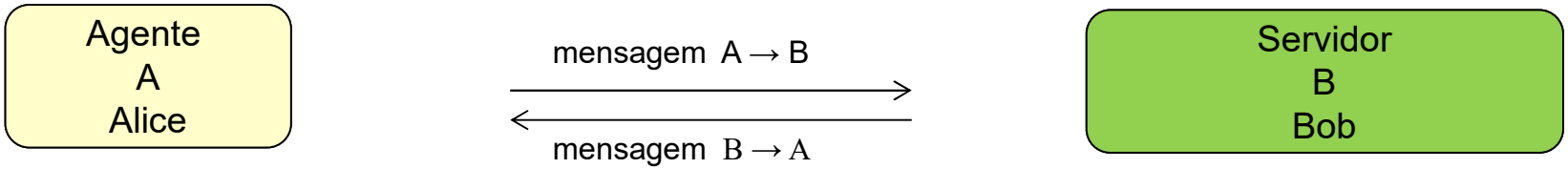
  Autenticação forte: Baseia-se na combinação de pelo menos 2 de 3 elementos: conhecimento, propriedade e inerência:

  1. **Conhecimento**: o que **sabe**. Ex: password, código PIN, número cartão cidadão
  2. **Propriedade**: o que **possui**. Ex: token, cartão matriz, telemóvel
  3. **Inerência**: o que **é**. Ex: características biométricas como impressão digital, retina

- Os elementos seleccionados devem ser independentes entre si: a violação de um não compromete os outros
- Pelo menos um dos elementos deve ser reutilizável, não replicável e não susceptível de ser roubado clandestinamente
- O procedimento de autenticação forte deve ser concebido de forma a proteger a confidencialidade dos dados de autenticação

- **A criptografia suporta normalmente o elemento de autenticação forte baseado no conhecimento**

Autenticação com Chave Secreta



Kab - Chave Secreta, só conhecida por A e B

Autenticação e Privacidade

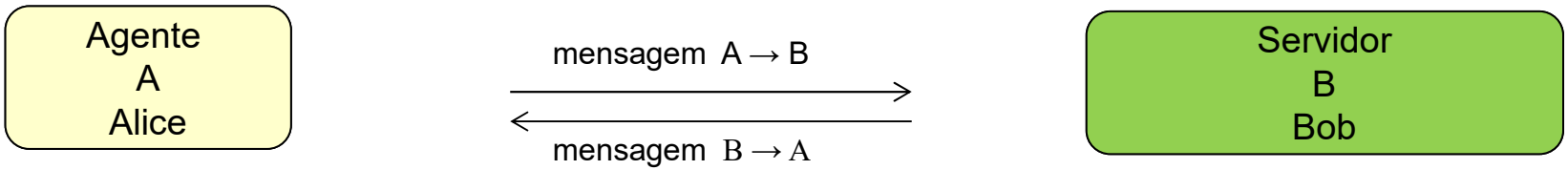
$A \rightarrow B \{M\}_{Kab}$ A envia a B uma mensagem M cifrada com a chave secreta só conhecida por A e B

$B: \{\{M\}_{Kab}\}_{Kab}$ B decifra a mensagem M com a chave secreta só conhecida por A e B

- Mensagem autenticada: só pode ter sido gerada por A, pois para além de B, só A conhece a chave secreta
- Conteúdo confidencial: mais ninguém conhece a chave secreta para decifrar a mensagem

<u>Todas</u> as mensagens têm que ser autenticadas

Autenticação com Chave Pública



KApriv - Chave privada de A, só conhecida por A
KApub - Chave pública de A, conhecida por toda a gente

KBpriv - Chave privada de B, só conhecida por B
KBpub - Chave pública de B, conhecida por toda a gente

Autenticação

$A \rightarrow B \{M\}_{KApriv}$ A envia a B uma mensagem M cifrada com a sua chave privada

$B: \{\{M\}_{KApriv}\}_{KApub}$ B decifra a mensagem M com a chave pública de A

- Mensagem autenticada: só pode ter sido gerada por A, pois só A conhece a sua chave privada
- Conteúdo da mensagem é público: qualquer um pode decifrar a mensagem com a chave pública de A

Privacidade

$A \rightarrow B \{M\}_{KBpub}$ A envia a B uma mensagem M cifrada com a chave pública de B

$B: \{\{M\}_{KBpub}\}_{KBpriv}$ B decifra a mensagem M com a sua chave privada

- Mensagem privada: só pode ser decifrada com a chave privada de B, que só este conhece
- Conteúdo não autenticada: qualquer um pode enviar esta mensagem cifrada com a chave pública de B

Autenticação e Privacidade

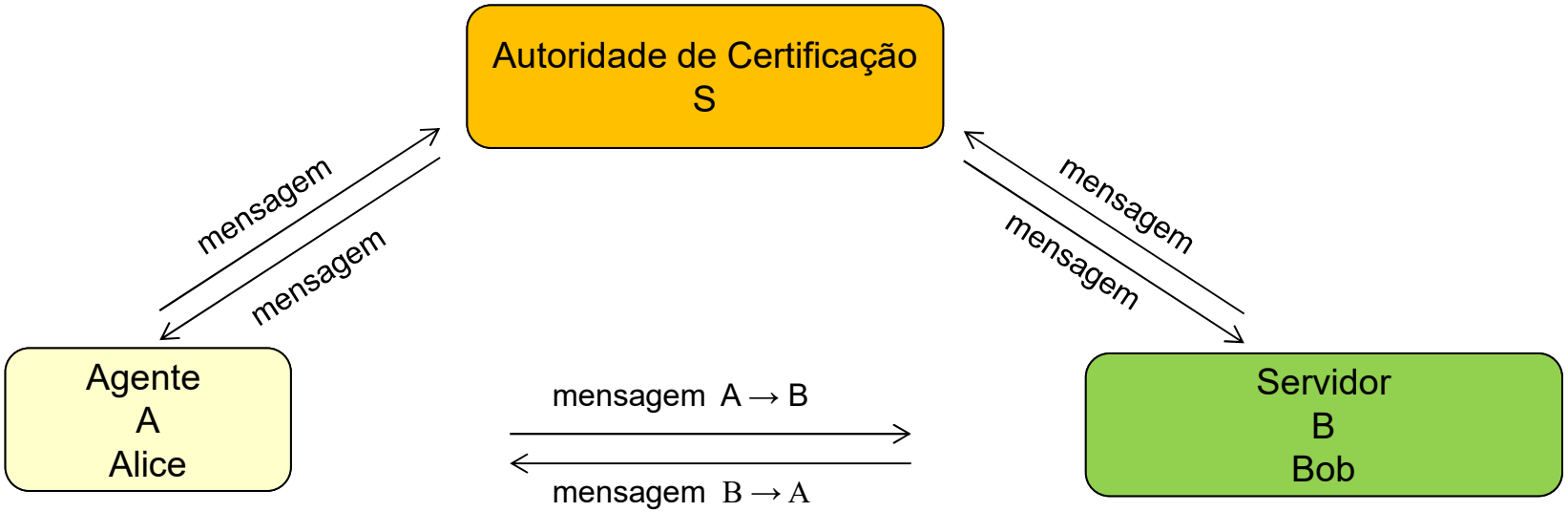
$A \rightarrow B \{\{M\}_{KApriv}\}_{KBpub}$ A envia a B uma mensagem M cifrada com a sua chave privada e com a chave pública de B

$B: \{\{\{\{M\}_{KApriv}\}_{KBpub}\}_{KBpriv}\}_{KApub}$ B decifra a mensagem M com a sua chave privada e depois com a chave pública de A

Utilização de Chave Pública e Chave Secreta

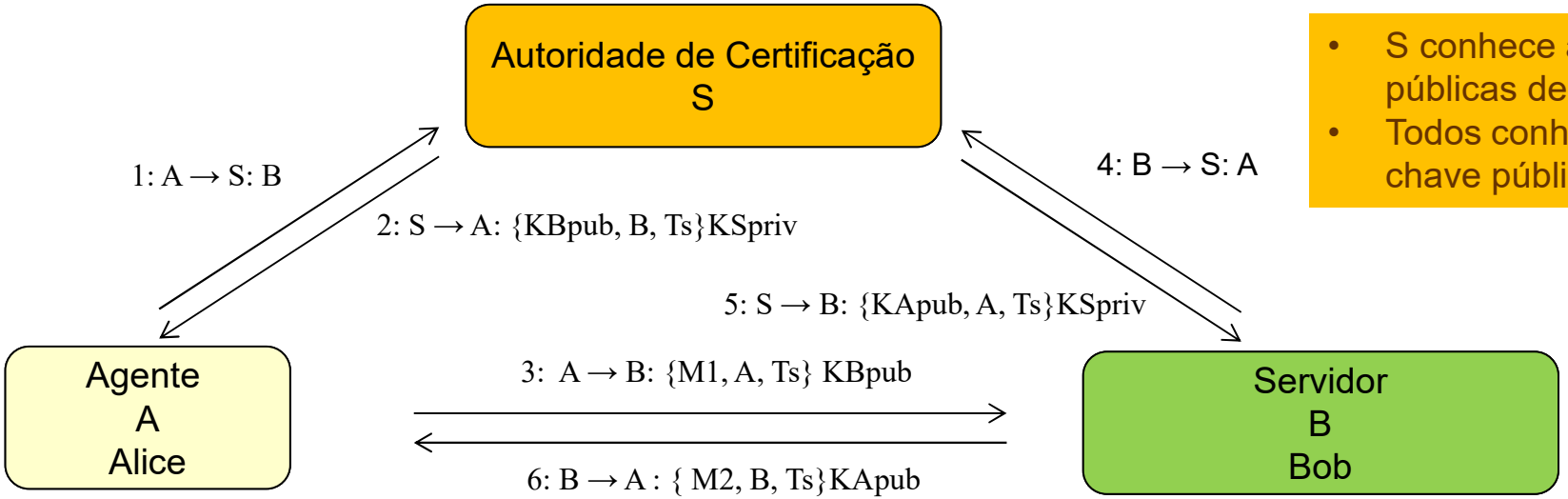
- A cifra com chave pública é muito mais lenta que com chave secreta
- Se a comunicação entre A e B for frequente, o mecanismo de chave pública é utilizado inicialmente para obter uma chave secreta entre A e B
- As mensagens seguintes são cifradas com a chave secreta entre A e B

Estabelecimento das Chaves



- Um aspecto crucial da autenticação é estabelecer as chaves de forma segura
- Autoridade de Certificação:
  - Entidade de confiança das várias partes
  - Armazena de forma segura as chaves públicas de cada entidade
  - Oferece um protocolo para distribuir ou certificar as chaves

Protocolo de distribuição de chaves públicas



- S conhece as chaves públicas de A e B
- Todos conhecem a chave pública de S

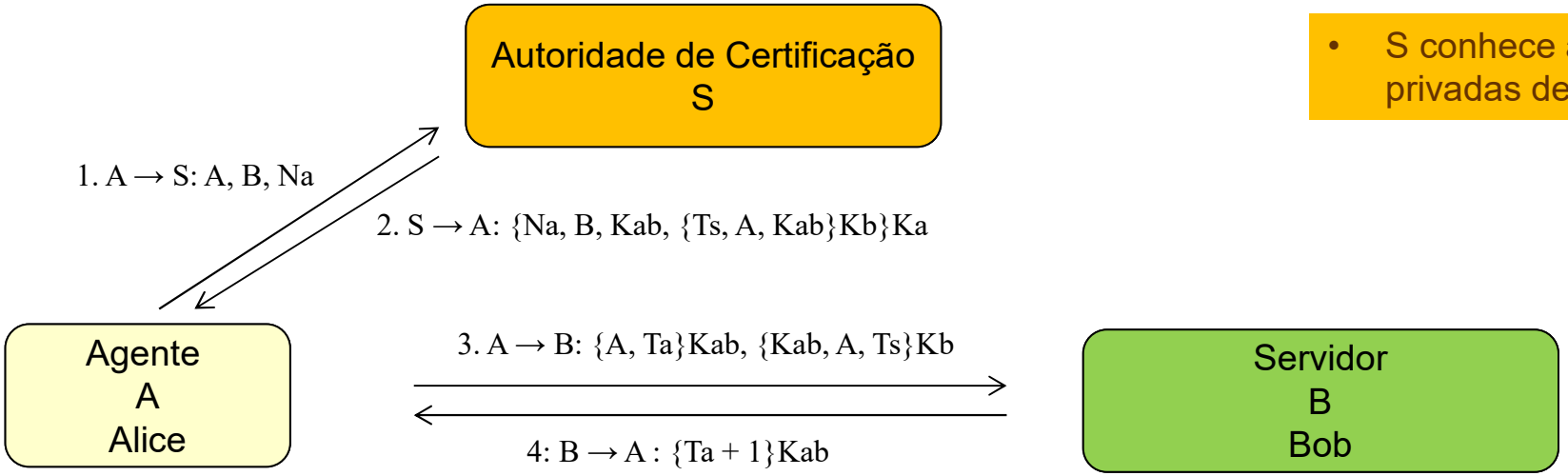
1. A → S: B	A pede à Autoridade de Certificação a chave pública de B.
2. S → A: {KBpub, B, Ts}KSpriv	A Autoridade de Certificação envia para A a chave pública de B (KBpub), autenticada através da cifra com a sua chave privada A mensagem é decifrada utilizando a chave pública do servidor de autenticação, conhecida de todos. O cliente fica seguro, ao decifrar a mensagem, de que esta foi enviada por S, porque apenas este conhece a chave privada correspondente.
3. A → B: {M1, A, Ts} KBpub	A envia a B uma mensagem cifrada com a chave pública de B que contém o seu identificador e um carimbo temporal.
4. B → S: A	As etapas 4 e 5 repetem as etapas 1 e 3 o protocolo do lado de B. Este pede à Autoridade de Certificação a chave pública de A.
5. S → B: {KApub A} KSpriv	A Autoridade de Certificação envia para B a chave pública de A (KApub), autenticada através da cifra com a sua chave privada
6. B → A: {M2, B, Ts}KApub	B envia a A a mensagem de resposta cifrada com a chave pública de A

Certificados de chaves públicas

- Certificado: documento assinado por uma autoridade de certificação
  - Confirma a validade de uma chave pública
  - Associa-a a informação do seu possuidor:
    - Informação sobre o dono (nome, e-mail, etc.)
    - Datas (de emissão, de validade)
- A norma X.509 é a mais utilizada para formato de certificados

Subject	Distinguished Name, Public Key
Issuer	Distinguished Name, Signature
Period of validity	Not Before Date, Not After Date
Administrative information	Version, Serial Number
Extended Information	

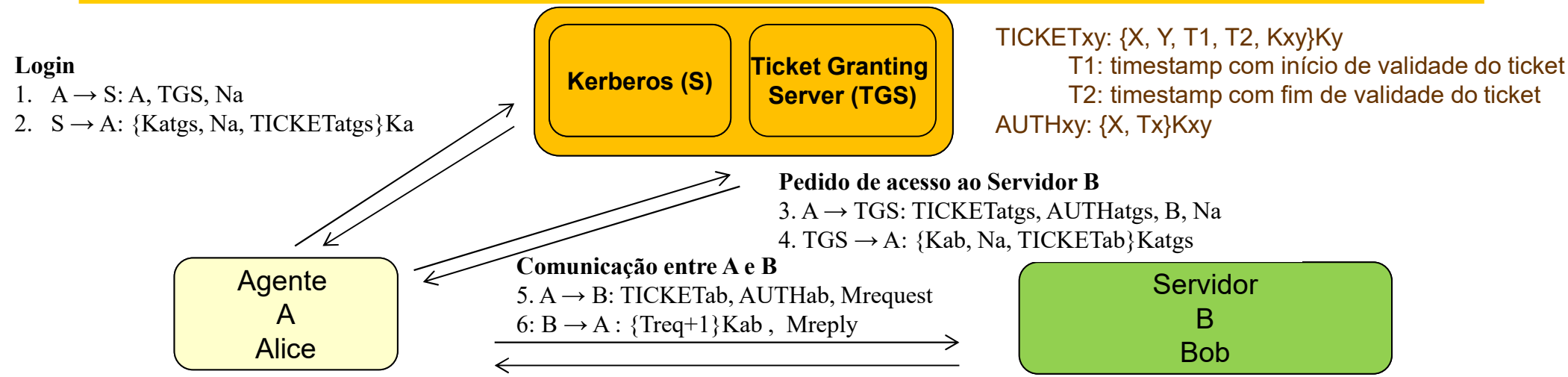
Protocolo de Needham-Schroeder
Autenticação com chave secreta



- S conhece as chaves privadas de A e B

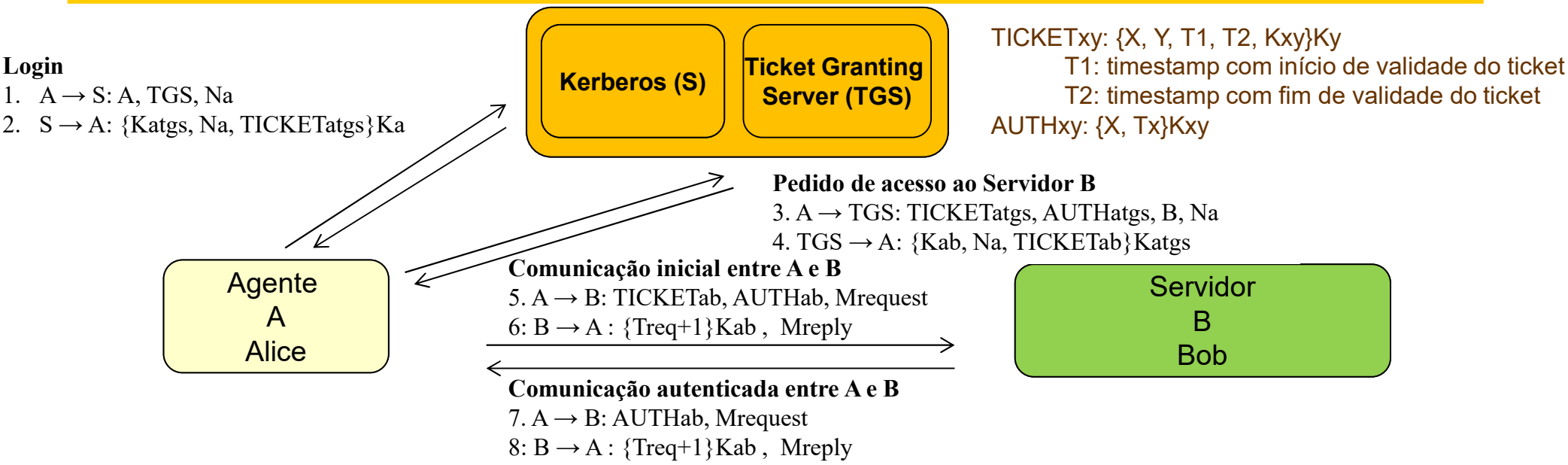
1. A → S: A, B, Na	A pede a S uma chave privada para comunicar com B. Inclui um valor arbitrário Na.
2. S → A: {Na, B, Kab, {Ts, A, Kab}Kb}Ka	S gera a chave Kab. Envia a A, cifrado com a chave de A, a chave Kab, um timestamp e um ticket incompreensível para A porque está cifrado com a chave de B. A decifra a mensagem com a sua chave e obtém a chave Kab, a identificação de B e Na. Confia na mensagem pois está cifrada com a sua chave. Na garante que esta mensagem é a resposta ao seu pedido.
3. A → B: {A, Ta}Kab, {Kab, A, Ts}Kb	A envia a B uma mensagem cifrada com a chave Kab contendo o seu identificador A, um timestamp Ta e o ticket que recebeu de S. B decifra o ticket com a sua chave secreta e obtém a chave Kab. Confia no ticket porque está cifrado com a sua chave secreta e Ts garante que é recente. Usando Kab decifra a outra parte da mensagem, obtém o identificador A e Ta. B confia na identidade de A pois recebeu essa informação cifrada com uma chave Kab que só pode ter sido gerada por S, que conhece a chave secreta de B.
4. B → A: {Ta + 1}Kab	B responde a A com uma mensagem contendo um valor derivado de Ta e cifrada com a chave Kab. A confia na identidade de B pois este respondeu ao seu desafio (Ta) numa mensagem cifrada com a chave secreta

Kerberos: servidor de autenticação baseado no protocolo Needham-Schroeder



1. A → S: A, TGS, Na	A pede a S uma chave para comunicar para o TGS.
2. S → A: {Katgs, Na, TICKETatgs}Ka	S cria uma chave secreta para A comunicar com o TGS e devolve-a cifrada com a chave de A. Envia também um ticket incompreensível para A. A obtém a chave para o TGS, guarda-a e apaga da memória a sua chave secreta (password).
3. A → TGS: TICKETatgs, AUTHatgs, B, Na	A envia ao TGS um pedido para comunicar com B onde inclui o ticket que recebeu de S, o autenticador cifrado com a chave secreta entre A e o TGS e a identificação de B
4. TGS → A: {Kab, Na, TICKETab}Katgs	O TGS cria a chave Kab para A comunicar com B e um ticket cifrado com a chave de B. Envia ambos cifrados com a chave secreta de A e TGS
5. A → B: TICKETab, AUTHab, Mrequest	A obtém a chave secreta Kab, envia a B o ticket recebido do TGS e autenticador cifrado com a chave secreta entre A e B. B decifra o ticket com a sua chave e obtém a chave secreta entre A e B. Com esta decifra o autenticador e confirma a identidade de A. O pedido está autenticado.
6: B → A : {Treq+1}Kab , Mreply	B responde a A cifrando o “challenge” Treq com a chave Kab. A decifra a resposta e obtém o “challenge” correcto. A resposta está autenticada.

Kerberos: servidor de autenticação baseado no protocolo Needham-Schroeder (cont.)



7. A → B: AUTHab, Mrequest	Depois da autenticação inicial, A já não necessita de enviar o ticket a B, mas o autenticador tem que ser enviado em todas as mensagens.
8: B → A : {Treq+1}Kab , Mreply	B tem que responder sempre com o "challenge" cifrado com a chave Kab para autenticar a resposta

Kerberos: implementação

- Nounces e timestamps são fundamentais para garantir a segurança do protocolo
  - Nounce (Nx): valor pré-definido conhecido por ambas as partes. Ex: nome do cliente, nome do servidor.
    - As mensagens têm que conter “nounces” para que o receptor reconheça que se trata de uma mensagem válida, e não um conjunto de valores arbitrários
  - Timestamp (Tx): tempo real na máquina do emissor
    - Uma chave antiga pode ser comprometida com um ataque brute force
    - As mensagens têm que conter timestamps para garantir que são mensagens recentes e não repetições de mensagens antigas
- Os relógios dos clientes e servidores têm que estar sincronizados para validar *tickets* e *authenticators*
  - Kerberos rejeita mensagens com antiguidade superior a ~2 minutos. Os relógios têm que estar sincronizados com um erro inferior a 2 minutos
- Segurança do servidor Kerberos
  - As chaves secretas dos clientes e servidores (A, B, TGS) são derivadas das suas passwords
  - Para as poder gerar, o servidor Kerberos e o TGS têm que saber as passwords originais dos clientes e servidores
  - A segurança lógica e física dos servidores e das respectivas BDs é fundamental
- Escalabilidade
  - Subdivisão em *realms*
  - Cada realm possui um S_{aut} e um TGS
  - Um *realm* pode aceitar autenticações feitas por outro

Protocolo Diffie-Hellman

- O objectivo deste protocolo é criar uma chave simétrica a partir da troca de valores em claro entre os dois interlocutores
- O algoritmo baseia-se na dificuldade computacional de calcular logaritmos de grandes números.

1. A e B escolhem um número primo p e um número g que é uma raiz primitiva módulo p
2. A e B trocam os números p e g abertamente através na rede
3. A e B escolhem cada um número aleatório, A escolhe a e B escolhe b. A e B mantêm os números a e b secretos. A e B calculam respectivamente:
   - $Ta = g^a \bmod p$
   - $Tb = g^b \bmod p$
4. Ta e Tb são trocados abertamente entre os dois interlocutores.
5. A e B calculam independentemente a chave secreta:
   - A calcula $Ka = Tb^a \bmod p$
   - B calcula $Kb = Ta^b \bmod p$
6. K = Ka = Kb. Os valores Ka e Kb são iguais e constituem a chave secreta

Exemplo

Cryptographic explanation

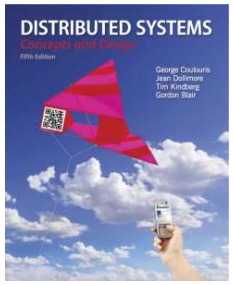
- The simplest and the original implementation of the protocol uses the multiplicative group of integers modulo p, where p is prime, and g is a primitive root modulo p. Here is an example of the protocol, with non-secret values in blue, and secret values in **bold**.
- Alice and Bob agree to use a prime number $p = 23$ and base $g = 5$ (which is a primitive root modulo 23).
- Alice chooses a secret integer $\boldsymbol{a} = \mathbf{6}$, then sends Bob $A = g^a \bmod p$
  - $A = 5^{\mathbf{6}} \bmod 23 = 8$
- Bob chooses a secret integer $\boldsymbol{b} = \mathbf{15}$, then sends Alice $B = g^b \bmod p$
  - $B = 5^{\mathbf{15}} \bmod 23 = 19$
- Alice computes $\boldsymbol{s} = B^a \bmod p$
  - $\boldsymbol{s} = 19^{\mathbf{6}} \bmod 23 = \mathbf{2}$
- Bob computes $\boldsymbol{s} = A^b \bmod p$
  - $\boldsymbol{s} = 8^{\mathbf{15}} \bmod 23 = \mathbf{2}$
- Alice and Bob now share a secret (the number **2**).
- Both Alice and Bob have arrived at the same value, because $(g^a)^b$ (for Bob, $8^{15} \bmod 23 = (g^a \bmod p)^b \bmod p = (g^a)^b \bmod p$) and $(g^b)^a$ are equal mod p. Note that only a, b, and$(g^{ab} \bmod p = g^{ba} \bmod p)$ are kept secret. All the other values – p, g, $g^a \bmod p$, and $g^b \bmod p$ – are sent in the clear. Once Alice and Bob compute the shared secret they can use it as an encryption key, known only to them, for sending messages across the same open communications channel.

Perguntas a que devo ser capaz de responder

- O que é autenticação forte ?
- Usando o mecanismo de chave pública (PKI), como é que A envia a B uma mensagem M autenticada ? E com privacidade ? E autenticada e com privacidade ?
- Como é que uma autoridade de certificação S envia a A a chave pública de B ? E como envia a A a chave privada de B ?
- O que é um certificado digital ?
- No protocolo Needham – Schroeder, porque é que B quando recebe a mensagem 3 acredita que ela vem de A ?
- Usando o protocolo Needham – Schroeder como é que A pode enviar a B uma mensagem M autenticada e com privacidade ?
- Para que servem os elementos Na e Ts nas mensagens 1 e 2 do protocolo Needham – Schroeder ?
- No Kerberos, porque é que os relógios dos clientes e servidores têm que estar sincronizados ?

Bibliografia

"Distributed Systems: Concepts and Design" (5th Edition), by Coulouris, Dollimore & Kindberg, Ed. Addison-Wiley, 2011 ISBN 0132143011



Capítulos da disciplina	Capítulos do livro
I. Introdução	1
II. Arquitecturas e Modelos de Comunicação Distribuída	2
III. Comunicação entre Processos Distribuídos (IPC)	4
IV. Invocação Remota (RPC) e Objectos Distribuídos (RMI)	5
V. Nomeação e Serviços de Directório	13
VI. Web Services (WS)	9
VII. Segurança	11
VII. Sistemas de Ficheiros Distribuídos	12
IX. Algoritmos Distribuídos	14, 15
X. Transacções	16, 17
XI. Hadoop	--