Autenticação

Nuno Neves
Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa

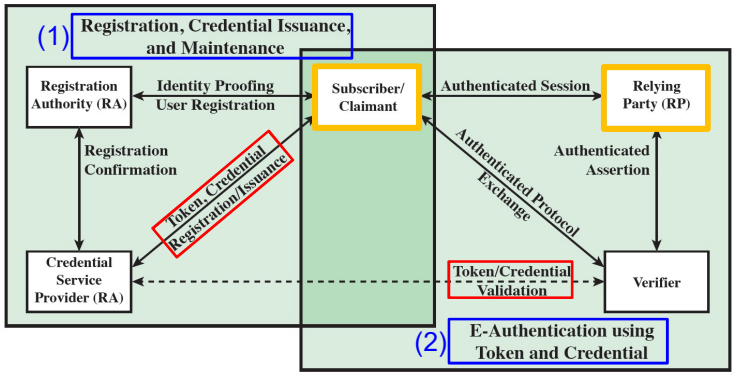
Autenticação

- Introdução
- Métodos de autenticação local / individual
- Métodos de autenticação remota
  - Autenticação unilateral
  - Autenticação mútua
  - Autenticação mediada

Objetivos de autenticação

- Um processo de autenticação permite a um utilizador demonstrar que é a entidade associada a um dado identificador (isto depois poderá ser usado para suportar decisões de autorização)

Modelo para Arquitetura de Autenticação (NIST SP 800-63-2)



AUTENTICAÇÃO LOCAL

Autenticação Local

- Verificar a identidade de Alice
- Entidades
  - Pessoas, serviços, servidores, máquinas
- Evitar a personificação de atacantes

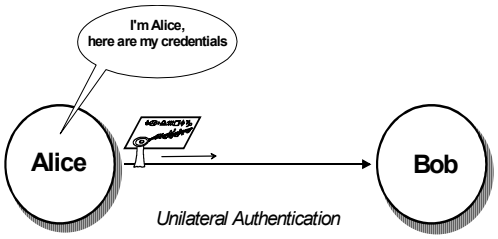
Métodos de autenticação

- Algo que se sabe
  - password, pin, …
- Algo que se possui
  - cartão magnético
  - smart card
- Algo que se é
  - biometria: característica física
    - impressão digital, íris, retina, voz, …
  - característica comportamental
    - padrão de escrita num teclado
- Algo que se tem acesso físico
  - estar próximo de quem autentica

- Vantagens/desvantagens
- Autenticação multi-fatores

Autenticação com password

- Autenticação
  - A autenticação é baseada no par <nome, *password*>
  - B verifica se a *password* corresponde à que está armazenada



- Questões a resolver:
  - Tipos de ataques mais comuns?
  - Como guardar as passwords ?

 9

Alguns ataques comuns às passwords

- **Ataque de dicionário offline:** captura o ficheiro onde as sínteses das passwords estão armazenadas e compara-as com uma lista pré computada (ou calcula essa lista na altura)
- **Ataque de dicionário online**: experimenta diferentes passwords no sistema para um dado utilizador
  - utiliza uma lista de passwords comuns
  - usa informação que recolheu sobre o utilizador para maximizar a probabilidade de acertar na password correta
- **Reutilização da password**: o utilizador emprega a mesma password em vários serviços; o adversário quebra um dos serviços para poder aceder aos restantes
- **Rapto de computador (workstation hijacking)**: espera que o utilizador deixe o computador e acede-o nessa altura
- **Engenharia social**: utiliza diferentes métodos para convencer o utilizador a dar-lhe a password

TPC: COMO NOS PODEMOS PROTEGER DESTES ATAQUES?

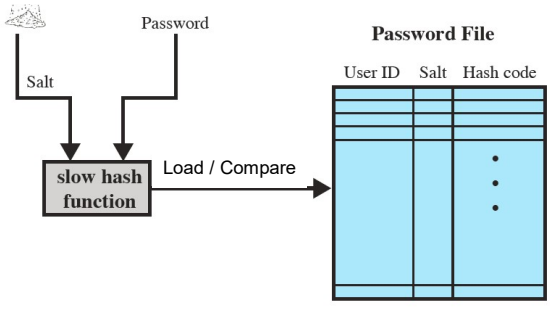
 10

Porque é que se continua a usar passwords?

- Técnicas que se baseiam em algum **dispositivo de hardware na máquina** (e.g., leitor de impressão digital) introduzem um custo adicional e também têm as suas limitações
- **Tokens físicos** (e.g., um cartão) introduzem custos e são inconvenientes de levar porque normalmente apenas servem para autenticar num local (o que obriga a que se tenha de levar vários tokens para os diversos locais)
- **Gestores de passwords** podem ser usados para manter informação sobre as passwords (e até executar o processo de autenticação em nome do utilizador) mas normalmente têm um suporte limitado se o utilizador tiver de se mover e aceder a vários computadores; para além disso são um ponto único da falha

Armazenamento de *Passwords*

- Texto em claro
  - Não é uma boa solução (quem lê o ficheiro vê as *passwords*)
- Síntese da *password*
  - Não é perfeita, mas torna a vida do adversário mais difícil
  - Quanto melhor a *password*, melhor a protecção obtida
- Síntese da *password* + *salt*



Exemplo: Autenticação no UNIX

Solução tradicional

1. *the 8-character (7-bit ASCII)* password *is converted into a 56-bit key to be used in a DES-derived encryption function called* crypt*; crypt is also parameterized with a 12-bit* salt *value, a randomized quantity that makes two entries of the same password always look different;*
2. `crypt` *is a cryptographic checksum algorithm; it starts using an all-zero block as input, uses the result as the input of the next round, and performs 25 rounds, using the key and the salt;*
3. *the result is translated to an 11-character printable ASCII* string*, and the password entry of a user in the password file becomes the triplet* ⟨userId, salt, string⟩*;*
4. *when a user logs in, she supplies* ⟨userId, password⟩*; the operating system indexes the password file with* userId *and grabs* ⟨salt, string⟩*;*
5. *the cryptographic checksum is performed on* ⟨salt, password⟩ *and the result compared with* string*. If they match, the user is authenticated.*

Solução mais recente (existem outras dependendo da versão de UNIX)

1. *Utiliza-se uma versão **"lenta"** da função de hash MD5 para calcular um hash de 128 bits da password junto com um salt de 48 bits*

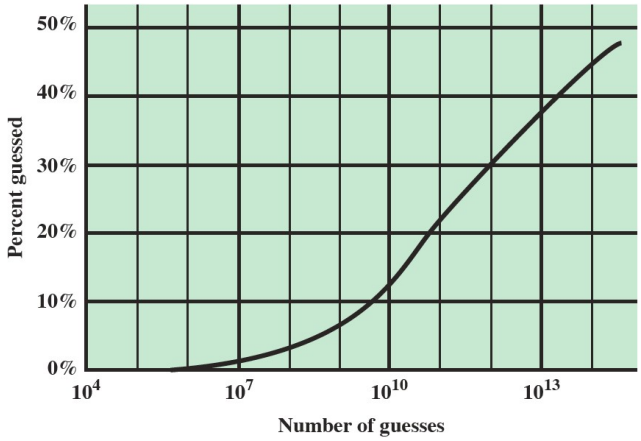
Vantagens do salt

- Evitam que utilizadores que usem a mesma password tenham o mesmo hash armazenado
- Aumentam significativamente a execução de ataques de dicionário offline (na proporção da dimensão do salt)
  - porquê?
- Tornam difícil determinar através da observação dos ficheiros de passwords se um utilizador usou a mesma password em mais de um sistema

- **Dificuldade**
  - *se o adversário capturar o ficheiro e assim tiver acesso aos salts e hashes das passwords, pode usar um programa que experimenta automaticamente diversas passwords até ter sucesso **ou** uma tabela com estes valores pré-computados (uma* **rainbow table**)

Exemplo de ataque de dicionário

- Apesar de os utilizadores utilizarem passwords cada vez mais seguras, as capacidades dos adversários também têm melhorado: **1)** uso de paralelismo com vários GPUs; **2)** algoritmos sofisticados para gerar passwords a testar



Um GPU moderno pode testar na ordem de 10^{10} passwords por segundo

Resultados usando um algoritmo moderno de geração de passwords

Métodos para se selecionar passwords mais seguras

- **Educação dos utilizadores:** ensinar os utilizadores como devem escolher as suas passwords
  - existe o risco de os utilizadores não seguirem os ensinamentos
- **Computador gera as passwords:** o computador gera aleatoriamente a password
  - dificuldade que os utilizadores não são capazes de as memorizar
- **Teste periódico (reactive password checking)**: de tempos a tempos testam-se as passwords com um programa de ataque para verificar se estas são seguras
  - dificuldade devido a ter-se de gastar tantos recursos como o adversário
- **Teste no registo (proactive password checker)**: quando o utilizador indica a sua password pela primeira vez, são feitos testes que evitam o registo de passwords fracas
  - tem de haver um equilíbrio entre a segurança e usabilidade

Autenticação com algo que se possui (Token)

- **Cartões de memória**
  - por ex., guardam numa fita magnética ou numa memória interna um segredo e outra informação, podendo estes ser obtidos por um leitor
  - tipicamente usa-se em conjunto com uma password
  - dificuldades: perca do cartão; requer um leitor específico

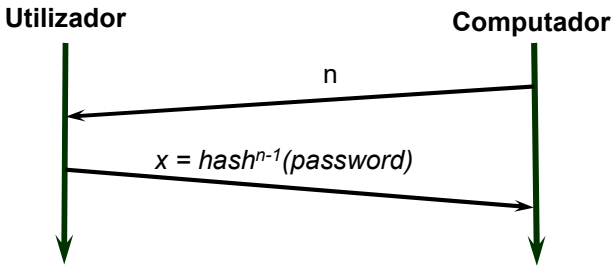
- **Cartões/dispositivos com processamento** (smart cards / smart token)
  - têm um processador embebido, capacidade de armazenar dados e alguma forma de interface (e.g., display/teclado, contatos elétricos ou comunicação sem fios)
  - suportam/correm algum tipo de protocolo de autenticação seguro com o leitor, sem divulgar as chaves armazenadas
    - o protocolo pode ser executado diretamente entre o cartão e leitor (ou através de informação que é passada pelo utilizador)

Passwords Descartáveis *(one time passwords)*

- *Passwords* que são usadas no máximo uma vez
  - após o uso, perdem sua validade
  - ainda que um *sniffer* escute a rede, nada pode fazer com a password que viu …
- Mecanismos para geração de passwods descartáveis
  - Sincronização por tempo entre cliente/dispositivo e o servidor
    - password válida durante um curto período de tempo
  - Baseado em algoritmos criptográficos
    - a nova password é baseada na password anterior
    - *ver acetato seguinte*
  - Baseado em algoritmos de desafio-resposta
    - A nova password é baseada num desafio (e.g., um número aleatório ou contador) fornecido pelo servidor de autenticação

Exemplo: *Lamport one-time password*

- É calculado hash(.....hash(password)...) *n* vezes
- Computador conhece n e $hash^n$(password)



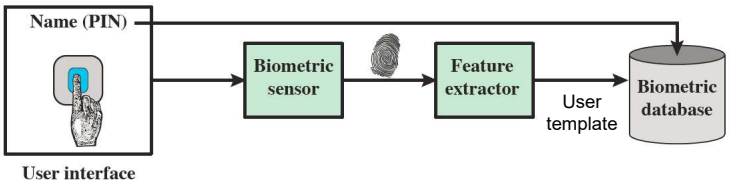
- Computador verifica a password:
  - *hash(x) é igual a $hash^n$(password) ???*

Autenticação com algo que se é (Biométrica)

- Baseia-se em características físicas estáticas (e.g., impressão digital) ou dinâmicas (e.g., voz)
- Em termos gerais, funciona recolhendo um padrão associado a uma característica biométrica da pessoa e depois comparando-o mais tarde

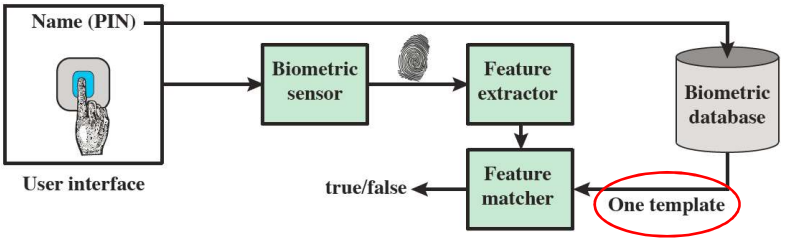
- Funcionamento

Registo

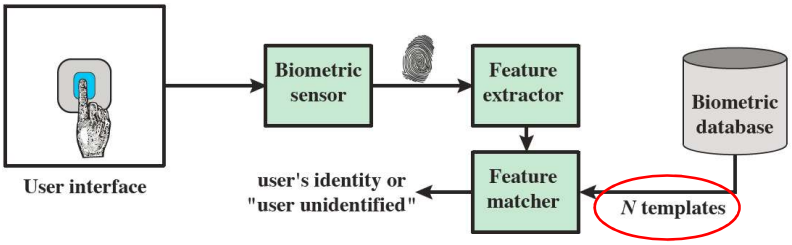


Usos da autenticação biométrica

*Verificação sendo fornecido o **nome do utilizador**:*

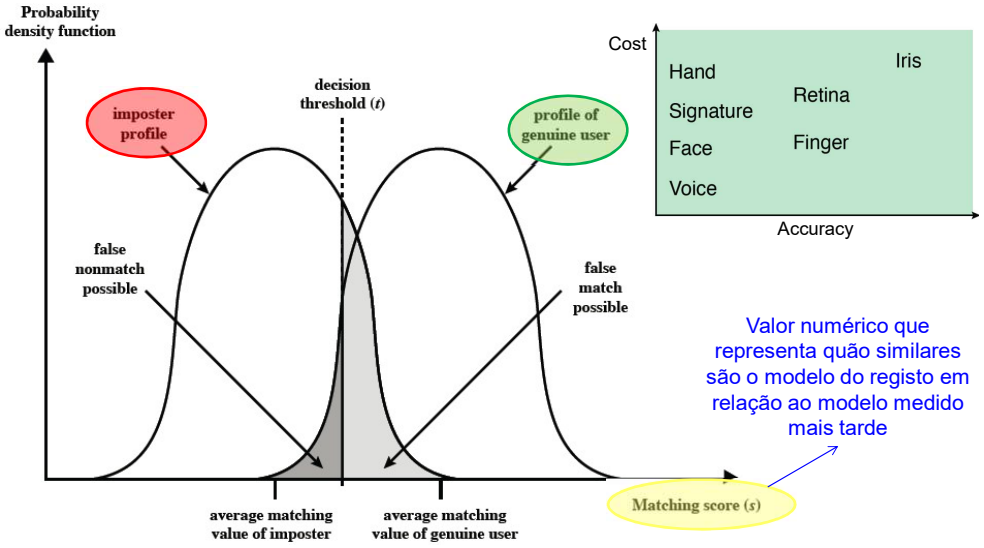


***Identificação** de um utilizador:*



 21

Exatidão



Valor numérico que representa quão similares são o modelo do registo em relação ao modelo medido mais tarde

 22

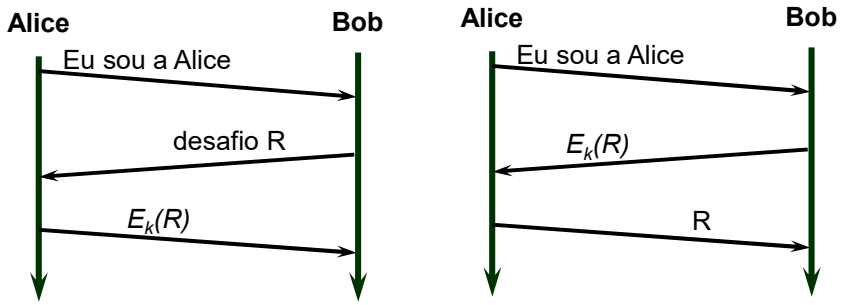
AUTENTICAÇÃO REMOTA

Autenticação unilateral (remota)

- **Autenticação unilateral**
  - apenas garante a autenticidade de quem inicia a comunicação
- Formas mais comuns no passado ☹
  - password
  - endereço IP
  - problemas ??
- Autenticação com passwords descartáveis
  - problemas ??
- Vamos ver a seguir
  - chave secreta partilhada
  - chaves assimétricas

Autenticação Unilateral por Partilha de Segredo

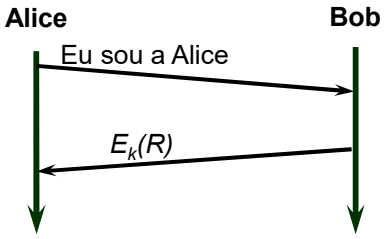
- A e B partilham um segredo *K*
- Mecanismos
  - Desafio/resposta
  - Marcas de tempo
    - Alice envia para Bob: timestamp, Hash(K || timestamp)



E - transformação criptográfica baseada em k, como um algoritmo de cifra (ou uma função de síntese no ex. da esquerda)

Autenticação Unilateral por Partilha de Segredo (cont)

- A e B partilham um segredo *K*



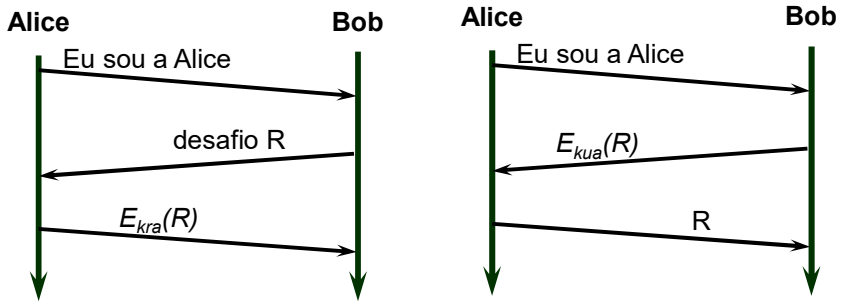
Será que se se enviasse apenas a segunda mensagem seria suficiente ?

R = desafio, ou seja, é uma número aleatório que não se deve repetir no espaço e no tempo (também chamado de nonce)

Se um adversário enviar um número aleatório na mensagem 2, a Alice após decifrar também irá obter um número aleatório. Logo, não consegue determinar se recebeu um número correto ou não.

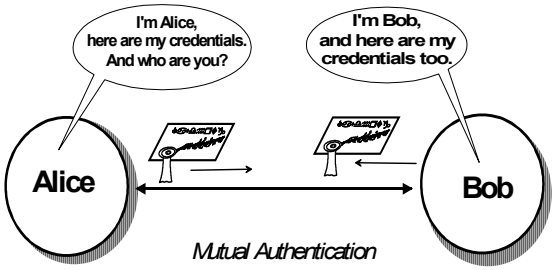
Autenticação Unilateral com chaves assimétricas

- Assinatura
  - B conhece a chave pública, *Kua*, de A
  - Após uma troca de mensagens, B deve ficar convencido que o seu interlocutor conhece chave privada *Kra*, logo deve ser A

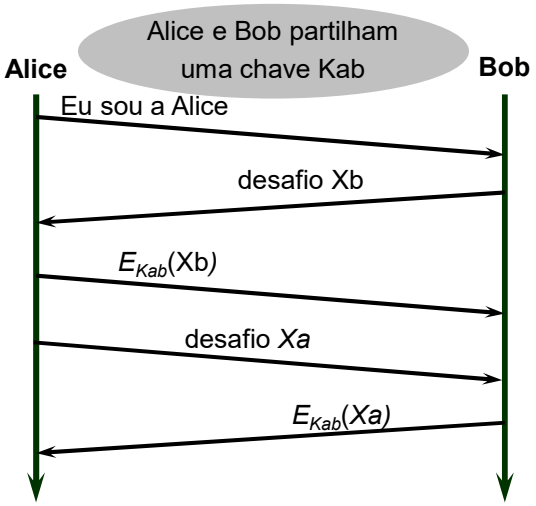


Autenticação mútua

- **Autenticação mútua** ☺
  - A prova a sua identidade a B, e vice versa
- Vamos ver a seguir
  - chave secreta partilhada
  - chaves assimétricas

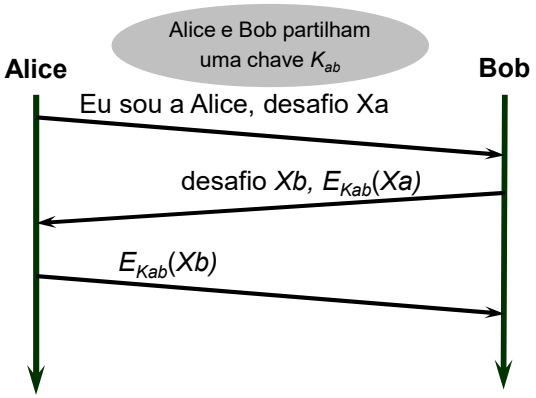


Autenticação Mútua com Segredo Partilhado



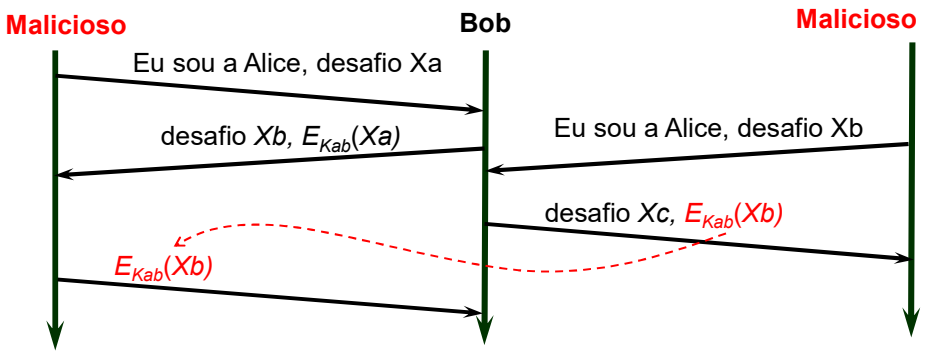
- Como optimizar para 3 mensagens ??

Autenticação Mútua com Segredo Partilhado



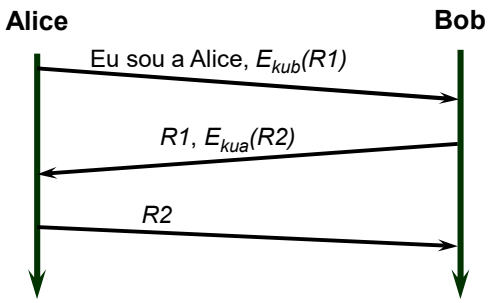
- Problema
  - Ataque de reflexo

Autenticação Mútua com Segredo Partilhado



- Soluções
  - Duas chaves, cada uma usada para uma das direções
  - Desafios com características diferentes (par vs. ímpar)
  - Variante com marca de tempo

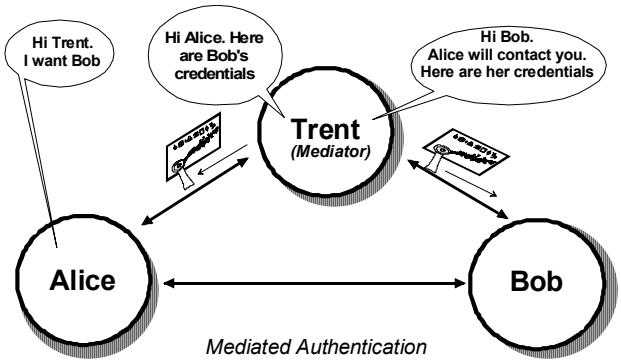
Autenticação Mútua com Criptografia Assimétrica



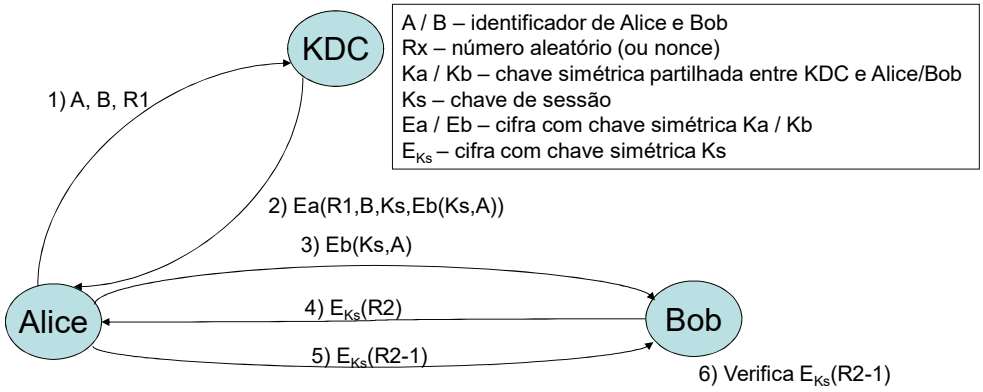
R1/R2 : são números aleatórios (*nonces*)

❖ Autenticação mediada

- A autentica-se perante B através de um "amigo comum" T
- A partilha um segredo Ka com T e B partilha um segredo Kb com T

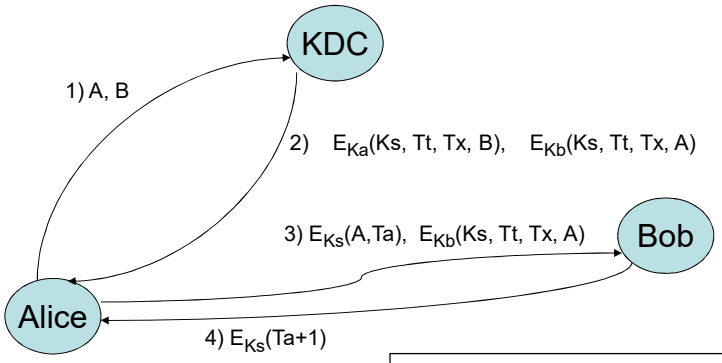


Autenticação Mediada: *Needham-Schroeder*



- Porquê R1? Porquê R2 ?
- Dois problemas :
  - Temos apenas autenticação unilateral. Quem autentica quem?
  - Atacante obtém chave de sessão antiga + mensagem do passo 3 -> consegue personificar a Alice ainda que ela altere a password; Possível solução: Usar marcas de tempo

Autenticação Mediada: *Kerberos*



A / B – identificador de Alice / Bob
Ka / Ka – chave simétrica entre KDC e Alice/Bob
Ks – chave de sessão
E_{Ka} / E_{Kb} / E_{Ks}– cifra simétrica com Ka / Kb / Ks
Tt – marca de tempo aquando da geração de Ks
Ta – marca de tempo corrente
Tx – Prazo de validade de Ks

- Dificuldade
  - Relógios sincronizados

Bibliografia

- Stallings & Brown, Computer Security: Principles and Practice, Third Edition, 2015
  - Leitura obrigatória: cap 3; cap 23 (Kerberos)
  - Leitura opcional: