ACH 2147 — Desenvolvimento de Sistemas de Informação Distribuídos

Aula 28: Segurança (parte 2)

Prof. Renan Alves

Escola de Artes, Ciências e Humanidades — EACH — USP

21/06/2024

Autenticação

Essência

Verificar a identidade reivindicada de uma pessoa, um componente de software, um dispositivo, etc.

Meios de autenticação

1. Baseado no que um cliente sabe, como uma senha.
2. Baseado no que um cliente tem, como um cartão de identidade, telefone celular ou token de software.
3. Baseado no que um cliente é, ou seja, biometria estática, como uma impressão digital ou características faciais.
4. Baseado no que um cliente faz, ou seja, biometria dinâmica, como padrões de voz ou padrões de digitação.

Autenticação

Essência

Verificar a identidade reivindicada de uma pessoa, um componente de software, um dispositivo, etc.

Meios de autenticação

1. Baseado no que um cliente sabe, como uma senha.
2. Baseado no que um cliente tem, como um cartão de identidade, telefone celular ou token de software.
3. Baseado no que um cliente é, ou seja, biometria estática, como uma impressão digital ou características faciais.
4. Baseado no que um cliente faz, ou seja, biometria dinâmica, como padrões de voz ou padrões de digitação.

Observações

- É possível utilizar mais de um tipo de autenticação (multi-factor authentication).
- Quando autenticar?

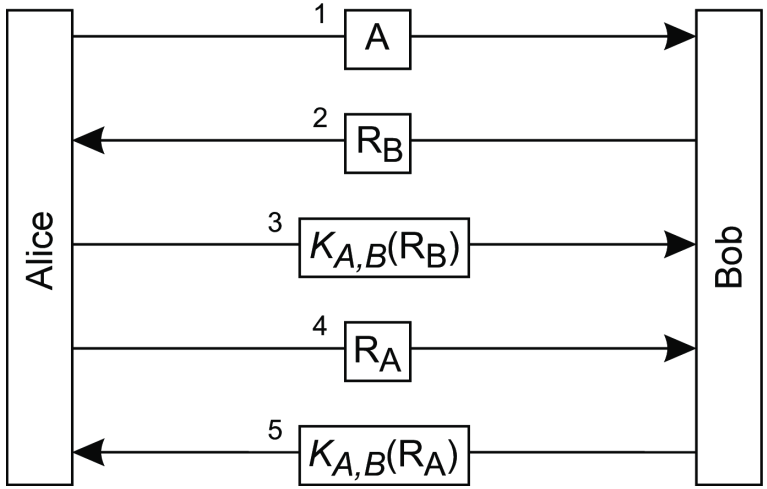
Autenticação versus integridade da mensagem

Importante

Autenticação sem integridade (e *vice-versa*) não tem sentido:

- Considere um sistema que suporta autenticação, mas sem mecanismos para garantir a integridade da mensagem. Bob pode saber com certeza que Alice enviou *m*, mas quanto esta informação é útil é isso se Bob não sabe que *m* pode ter sido modificado?
- Considere um sistema que garante a integridade da mensagem, mas não fornece autenticação. Bob pode ficar feliz com uma mensagem garantida não modificada que diz que ele acabou de ganhar R$1,000,000?

Autenticação com base em uma chave secreta compartilhada

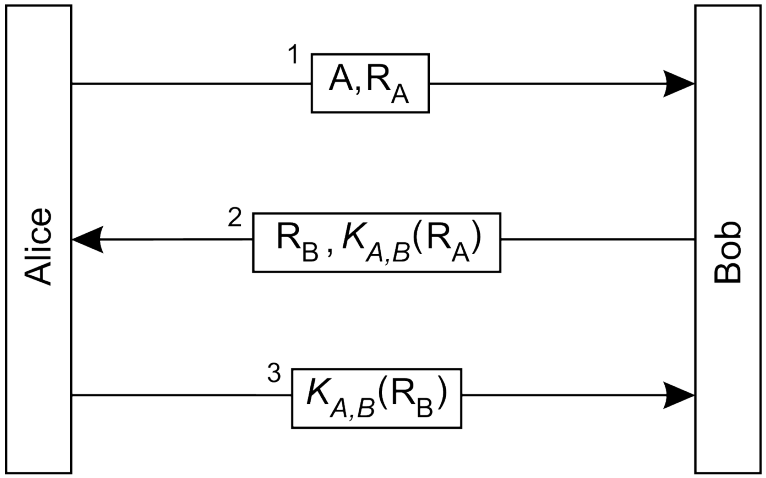


Protocolo de desafio e resposta

1. Alice anuncia que quer falar com Bob.
2. Bob retorna um nonce.
3. Alice criptografa o nonce com a chave compartilhada $K_{A,B}$, provando assim que ela possui $K_{A,B} \Rightarrow$ Bob sabe que está falando com Alice.
4. Alice envia um nonce para Bob.
5. Bob retorna a prova de que ele também possui a chave secreta compartilhada $\Rightarrow$ Alice sabe que está falando com Bob.

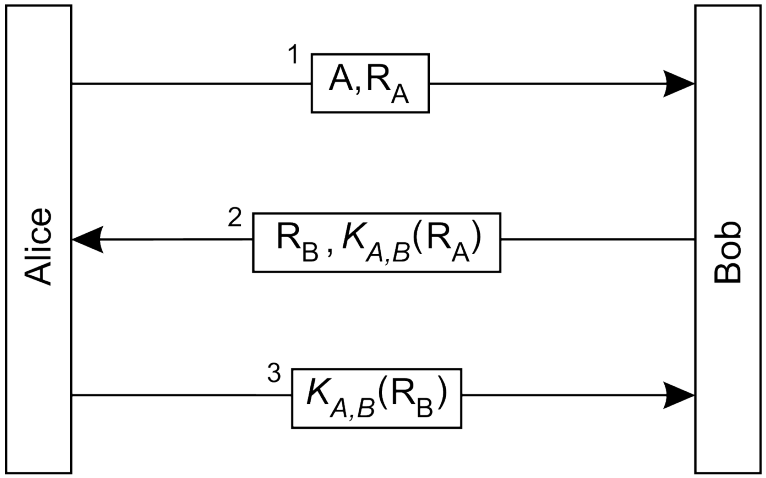
Sobre otimizações

Vamos reduzir o número de mensagens

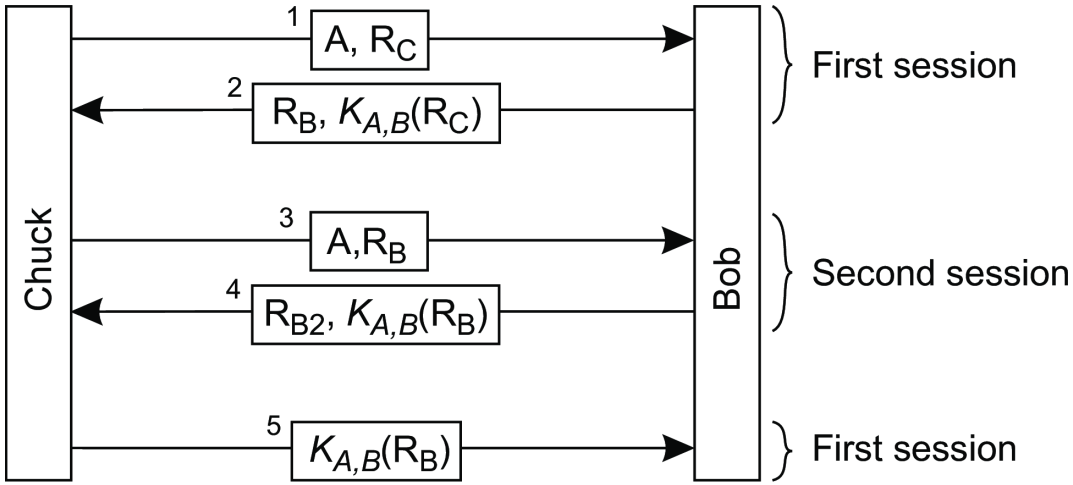


Sobre otimizações

Vamos reduzir o número de mensagens



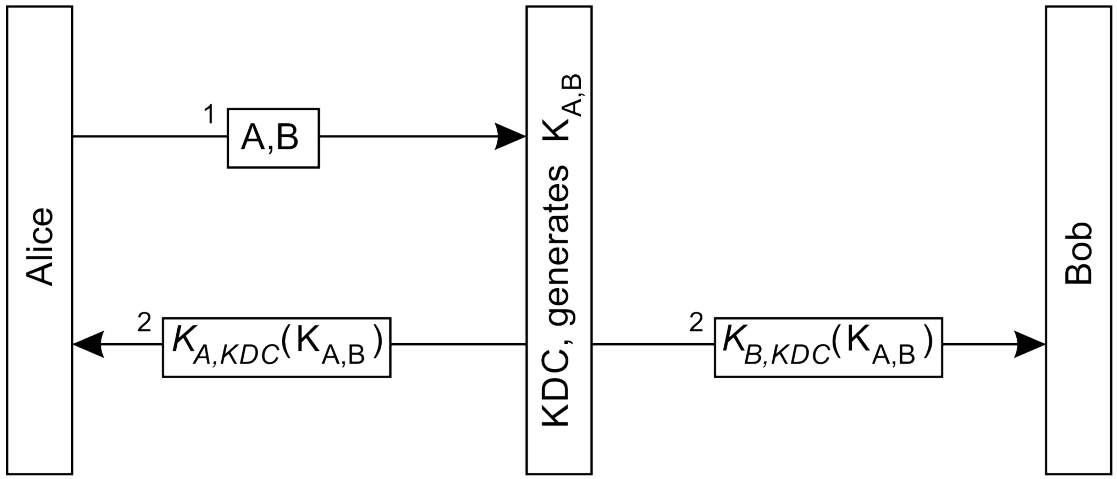
Nós acabamos de quebrar o protocolo: Bob se tornou um oráculo de encriptação



Autenticação com base em um Centro de Distribuição de Chaves (KDC)

Escalabilidade em manter chaves compartilhadas

Em um sistema com N entidades, cada entidade precisaria ter (N-1) chaves, num total de N*(N-1) chaves

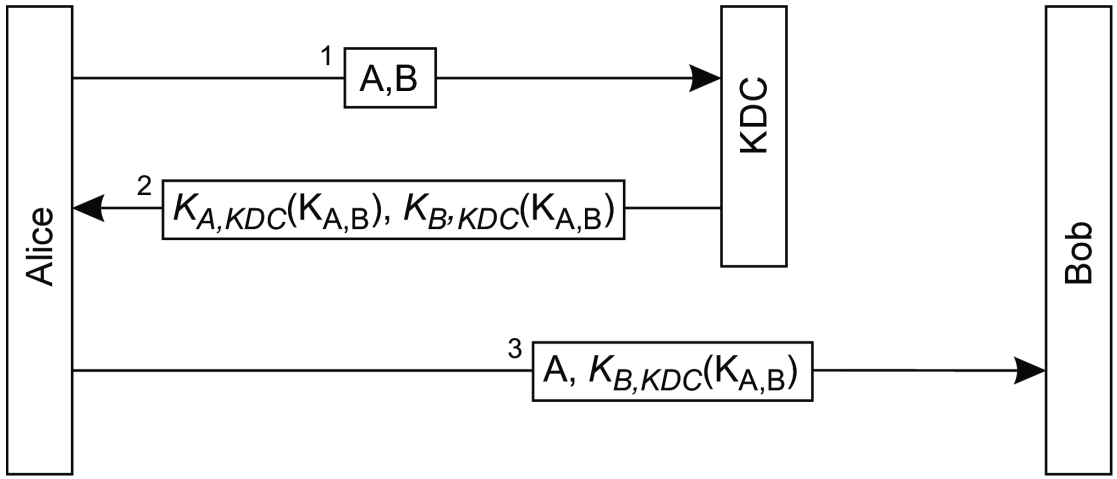


Alternativa: usar um KDC

Todo cliente tem uma chave secreta compartilhada com o KDC.

1. Alice diz ao KDC que quer falar com Bob
2. O KDC envia uma chave secreta nova, compartilhada por Alice e Bob

Autenticação com base em um Centro de Distribuição de Chaves (KDC)

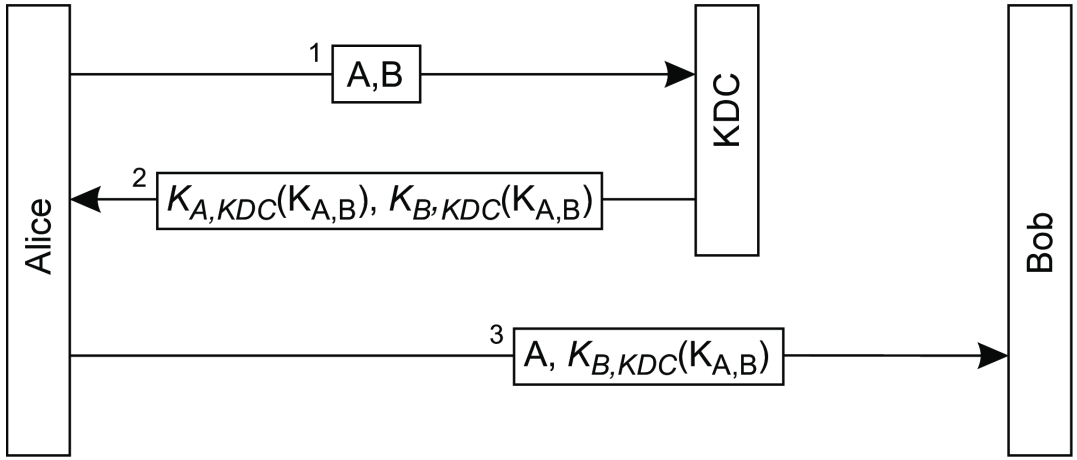


KDC com ticket

Usar um ticket é melhor na prática:

1. Alice diz ao KDC que quer falar com Bob
2. O KDC envia duas cópias de uma nova chave secreta, compartilhada por Alice e Bob.
   - Uma cópia é protegia por $K_{A,KDC}$: somente A pode decifrar
   - Uma cópia é protegia por $K_{B,KDC}$: somente B pode decifrar
3. Alice diz a Bob que quer conversar, junto com a chave a ser usada.

Autenticação com base em um Centro de Distribuição de Chaves (KDC)

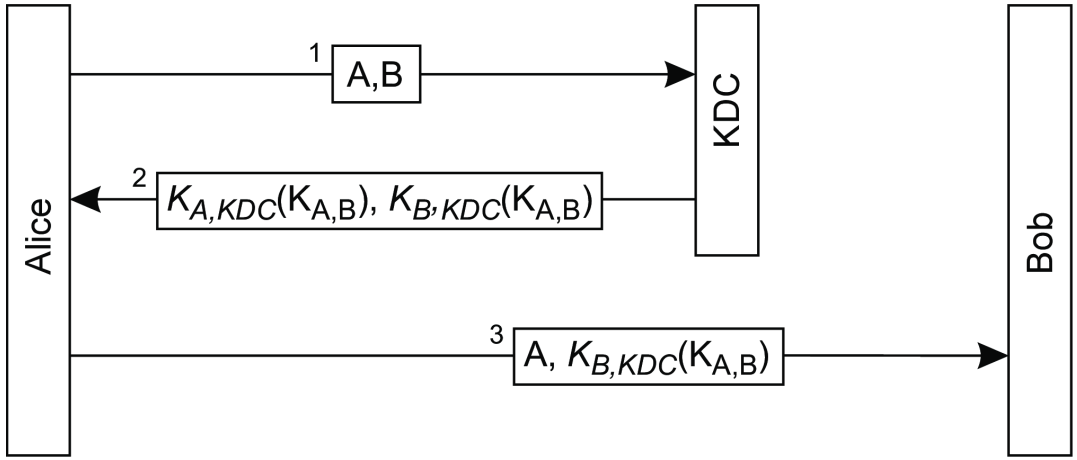


E se uma chave for comprometida?

Condições para o ataque:

1. Suponha que um atacante C obteve uma chave $K^{old}_{B,KDC}$
2. C havia interceptado uma resposta antiga do KDC

Autenticação com base em um Centro de Distribuição de Chaves (KDC)



E se uma chave for comprometida?

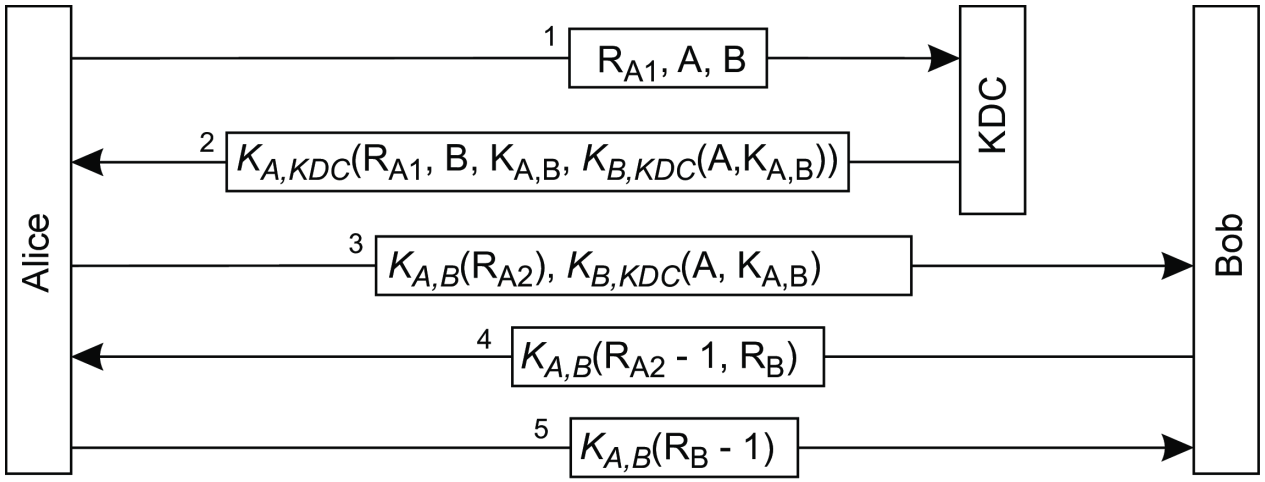
Condições para o ataque:

1. Suponha que um atacante C obteve uma chave $K^{old}_{B,KDC}$
2. C havia interceptado uma resposta antiga do KDC

Ataque:

1. Quando A fizer nova requisição, C consegue interceptar
2. C pode enviar a resposta anterior do KDC com $K^{old}_{B,KDC}$
3. C pode ser passar por B

O protocolo de Needham-Schroeder



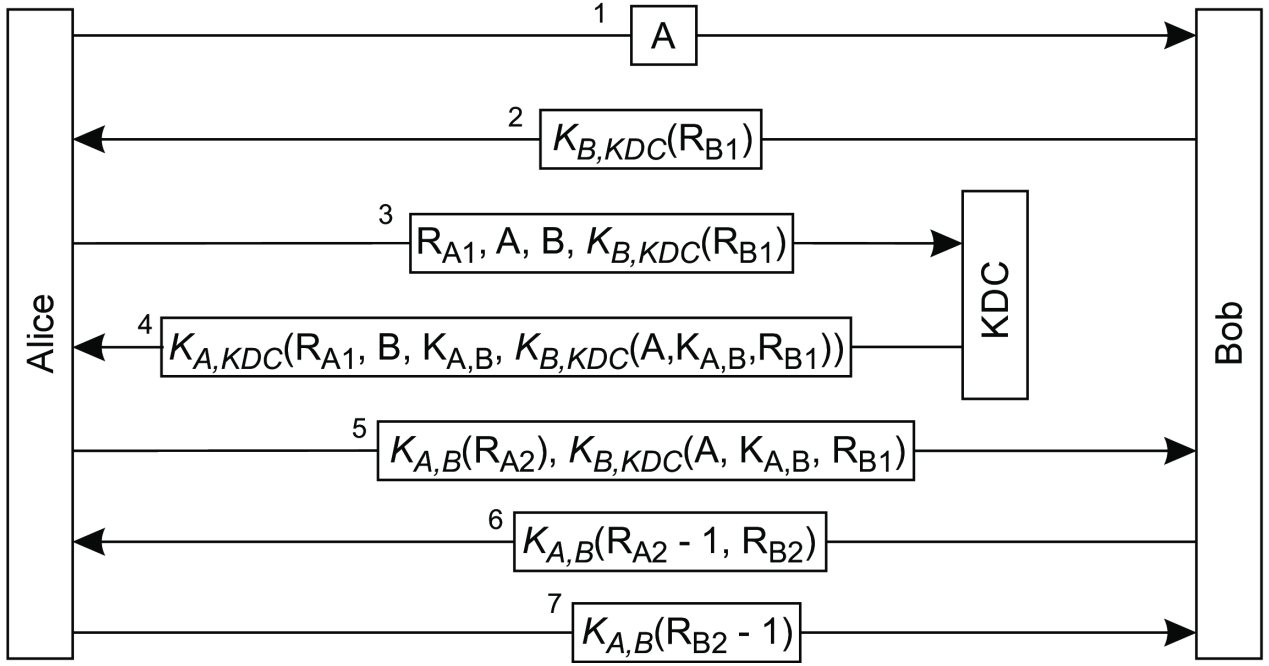
Observação importante

No caso de mensagens de requisição-resposta, você deve ter certeza de que a resposta recebida está estritamente associada à solicitação enviada.
Resultado: mitigação de ataques de repetição.

Princípio geral

Use nonces para relacionar qualquer combinação de mensagens de requisição-resposta.

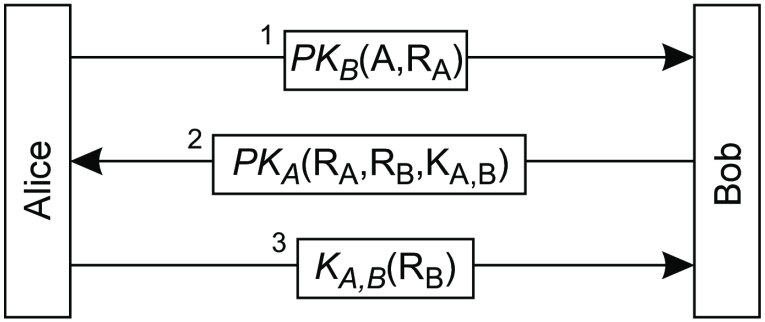
Mitigar contra a reutilização de chaves de sessão (K_{AB})



Algumas observações

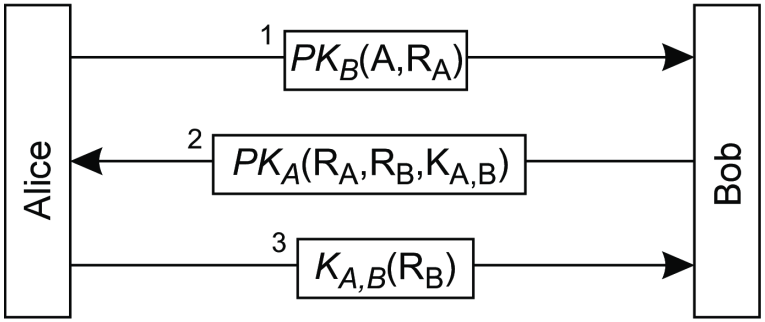
- Observe como *B1* vincula a mensagem #2 à #5
- Observe que ao retornar $R_{A2} - 1$ na #6, Bob prova que conhece $K_{A,B}$
- E, da mesma forma, no caso de Alice na #7 (modificando R_{B2}).

Usando criptografia de chave pública



1. Alice diz a Bob que quer conversar, enviando um nonce R_A e cifrando a mensagem com a chave pública de Bob.
2. Bob gera uma chave de sessão secreta compartilhada $K_{A,B}$, prova que é o proprietário de PK_B decifrando R_A e desafia Alice a provar que ela possui PK_A.
3. Alice decifra a resposta e prova a Bob que ela é Alice enviando de volta o nonce de Bob criptografado com a chave de sessão gerada $K_{A,B}$.

Usando criptografia de chave pública

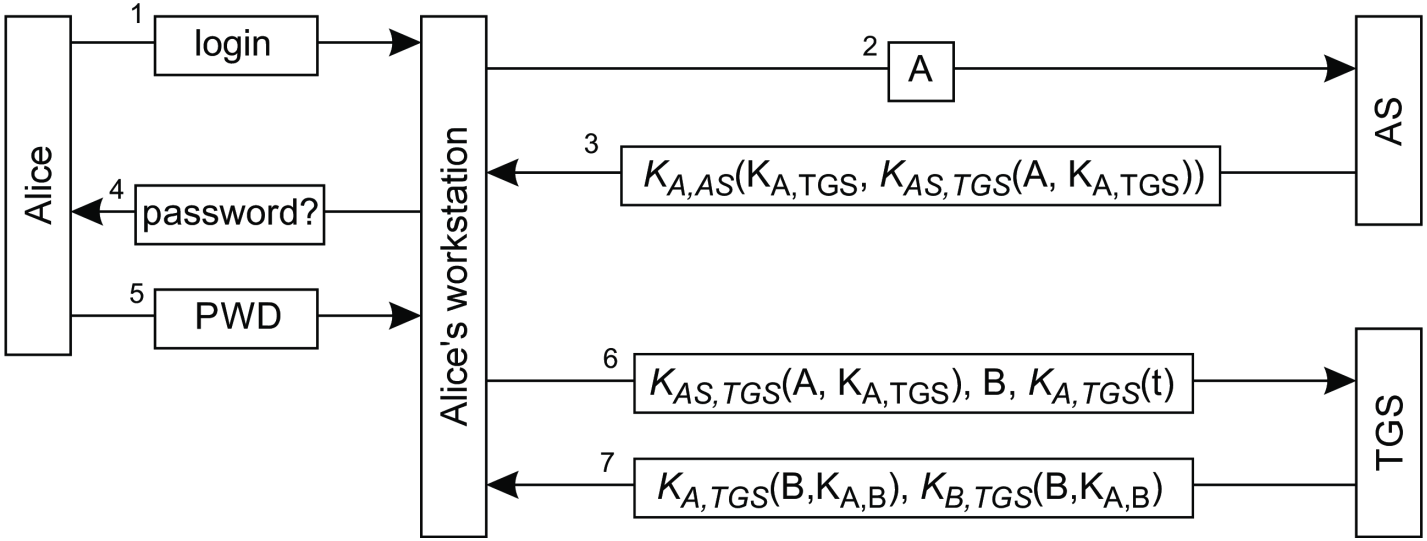


1. Alice diz a Bob que quer conversar, enviando um nonce R_A e cifrando a mensagem com a chave pública de Bob.
2. Bob gera uma chave de sessão secreta compartilhada $K_{A,B}$, prova que é o proprietário de PK_B decifrando R_A e desafia Alice a provar que ela possui PK_A.
3. Alice decifra a resposta e prova a Bob que ela é Alice enviando de volta o nonce de Bob criptografado com a chave de sessão gerada $K_{A,B}$.

Vantagens de estabelecer chaves de sessão

- Menos suscetível a ataques estatísticos
- Diminuição de danos caso chave seja exposta

Exemplo prático: Kerberos



Essência

1,2 Alice digita seu nome de login.

3 O Serviço de Autenticação (AS) retorna um ticket $K_{AS,TGS}(A, K_{A,TGS})$ que ela pode usar com o Serviço de Concessão de Tickets (TGS).

4,5 Para poder decifrar a mensagem, Alice deve digitar sua senha. Ela então estará logada na estação de trabalho. Usando o AS dessa maneira, temos um sistema de single sign-on.

6,7 Alice quer falar com Bob e solicita ao TGS uma chave de sessão.

Sobre confiança

Definição

Confiança é a garantia que uma entidade possui de que outra entidade realizará determinadas ações de acordo com uma expectativa específica.

Observação

- As expectativas foram explicitadas $\Rightarrow$ não há necessidade de falar sobre confiança?
- Exemplo: Considere um grupo de processos tolerante a falhas bizantinas de tamanho n
  - Especificação: o grupo pode tolerar que no máximo $k \leq (n-1)/3$ processos se tornem desonestos.
  - Consequência: se mais de k processos falharem, comportamento fica em aberto.
  - Consequência: não é sobre confiança, é sobre atender às especificações.

Ataque Sybil

Essência: criar múltiplas identidades controladas por uma única entidade

- No caso de uma rede peer-to-peer:

```
1  H = set of honest nodes
2  S = set of Sybil nodes
3  A = Attacker node
4  d = minimal fraction of Sybil nodes needed for an attack
5
6  while True:
7      s = A.createNode()          # create a Sybil node
8      S.add(s)                    # add it to the set S
9
10     h = random.choice(H)        # pick an arbitrary honets node
11     s.connectTo(h)              # connect the new sybil node to h
12
13     if len(S) / len(H) > d:     # enough sybil nodes for...
14         A.attack()              # ...an attack
```

Ataque Sybil

Essência: criar múltiplas identidades controladas por uma única entidade

- No caso de uma rede peer-to-peer:

```
1  H = set of honest nodes
2  S = set of Sybil nodes
3  A = Attacker node
4  d = minimal fraction of Sybil nodes needed for an attack
5
6  while True:
7      s = A.createNode()          # create a Sybil node
8      S.add(s)                    # add it to the set S
9
10     h = random.choice(H)        # pick an arbitrary honets node
11     s.connectTo(h)              # connect the new sybil node to h
12
13     if len(S) / len(H) > d:     # enough sybil nodes for...
14         A.attack()              # ...an attack
```

Exemplo: web-of-trust

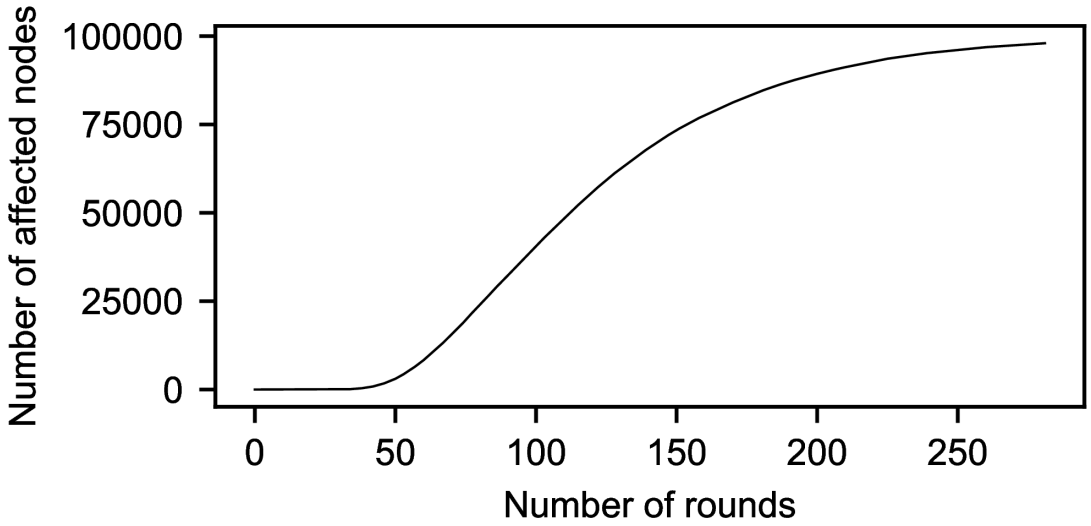
- Sistema com endosso de chave pública sem verificação fora da banda.
- *Bob* verifica com $k > 1$ outros que eles endossaram a chave de *Alice*.
- *Alice* cria $k > 1$ identidades cada uma declarando que sua chave é válida.

Ataque Eclipse

Essência: tentativa de isolar um nó da rede

Exemplo: um ataque de hub no caso de um serviço baseado em gossip.

Neste caso, ao trocar links com outros peers, um nó em conluio retorna links apenas para outros conspiradores.



Nós afetados: têm links apenas para nós conspiradores.

Solução geral

Use uma autoridade de certificação centralizada.

Prevenindo ataques Sybil: Blockchain

Essência: a criação de uma identidade tem um custo

No caso de blockchains sem permissão:

- Proof-of-Work: os validadores participam de uma corrida computacional. Essa abordagem requer consideráveis recursos computacionais
- Proof-of-Stake: a escolha do validador depende da quantidade de tokens que ele possui. Essa abordagem possui o risco de perda de tokens.

Prevenindo ataques Sybil: contabilidade descentralizada

Exemplo

- Cada nó P mantém uma lista de nós interessados em realizar alguma tarefa para P: o conjunto de escolha de P ($choice(P)$).
- Selecionar $Q \in choice(P)$ depende do trabalho feito por Q para os outros (ou seja, sua reputação).
- P mantém uma visão (subjetiva) sobre as reputações. Evidentemente, P sabe precisamente o que fez para os outros, e o que os outros fizeram para P. Porém, não necessariamente possui informação completa sobre as outras relações.
- P pode calcular uma capacidade ($cap(Q)$):

$$cap(Q) = \max\{MF(Q,P) - MF(P,Q), 0\}$$

com $MF(A,B)$ a quantidade de trabalho que A fez diretamente, ou contribuiu indiretamente para tarefas pedidas por B (ou seja, incluindo o trabalho feito por outros).

Prevenindo ataques Sybil: Contabilidade descentralizada

Como os ataques Sybil são prevenidos

- Suponha um $Q \in choice(P)$ criar n nós Sybil $Q_1^*, \ldots, Q_n^*$; $Q = Q_0^*$
- Para aumentar $cap(Q_i^*)$: Q_i^* precisa ter trabalhado para algum nó R

  Em outras palavras: Q só pode atacar com sucesso se tiver trabalhado para nós honestos.
- Considere a capacidade total dos atacantes:

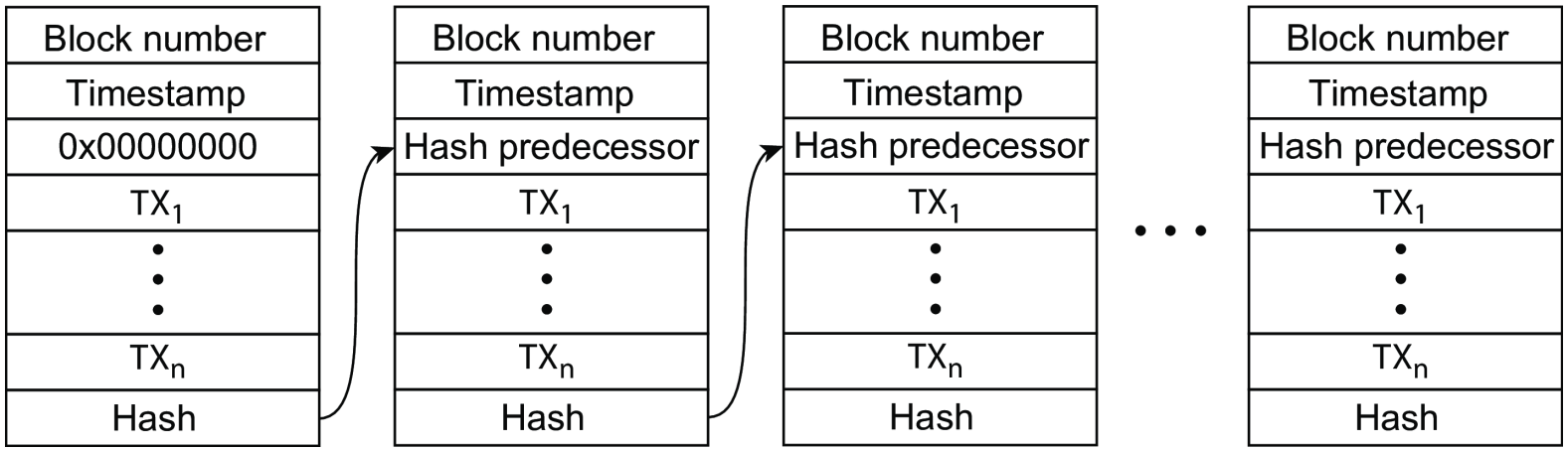
$$Tcap(Q) = \sum_{k=0}^{n} cap(Q_k^*)$$

- Suponha que P trabalha 1 unidade para $Q_i^* \Rightarrow MF(P, Q_i^*)$ aumenta em 1 unidade $\Rightarrow cap(Q_i^*)$ diminui em 1 unidade, e assim também $Tcap(Q)$.
- Assim que $Tcap(Q)$ cai para 0, P dará prioridade para outros nós.

Confiando em um sistema: Blockchains

Essência

É preciso saber com certeza que as informações em um blockchain não foram adulteradas: garantia de integridade de dados. Solução: certificar-se de que nenhuma alteração possa passar despercebida (lembre-se: um blockchain é uma estrutura de dados “append-only”).



Observação

Qualquer alteração no bloco B_k afetará seu valor hash, e assim o de B_{k+1}, que também precisará ser alterado, afetando por sua vez o valor hash de B_{k+2}, e assim por diante.