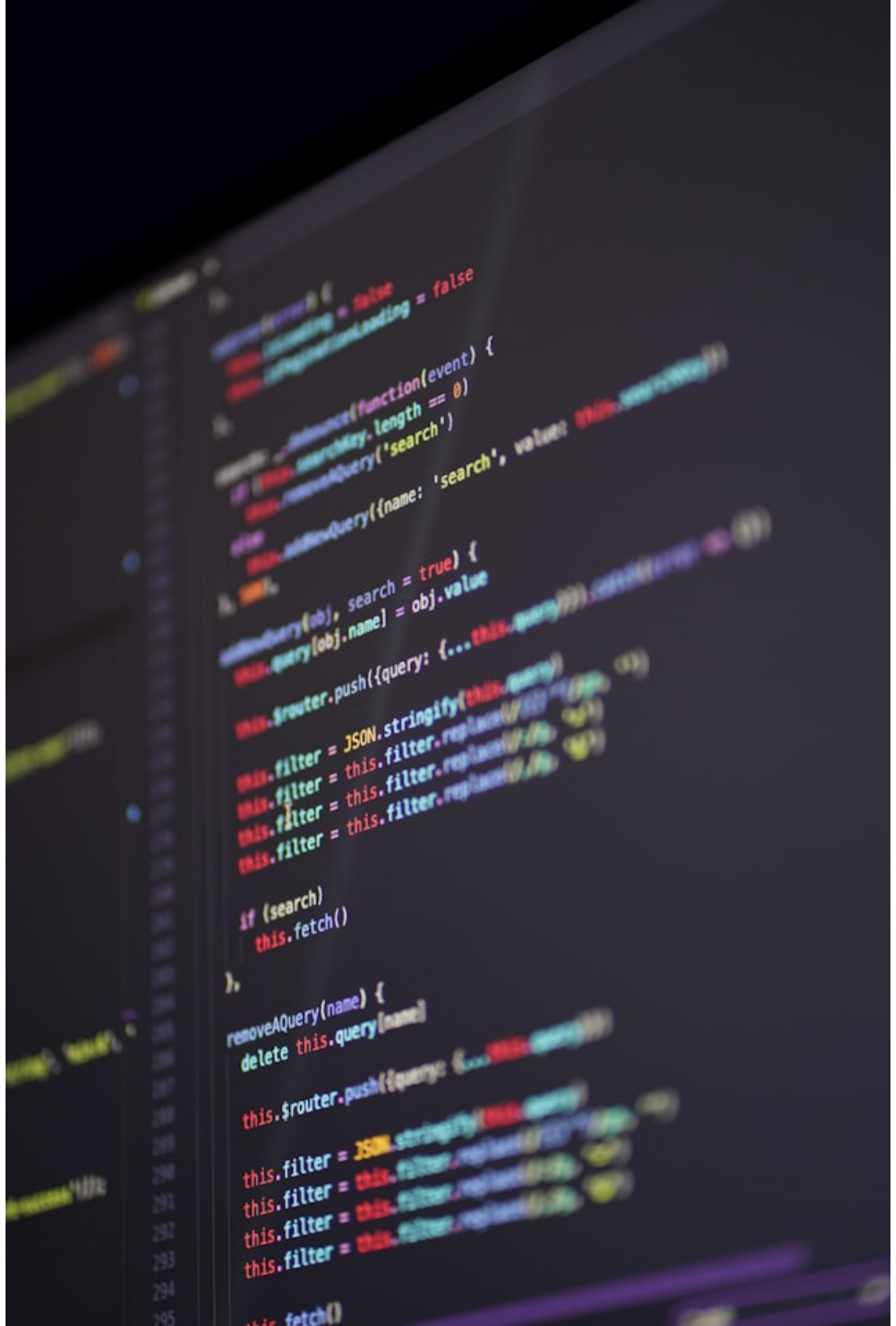


5954025 - Sistemas Distribuídos

Aula 13b - Segurança (Parte 2)

Prof. Dr. Denis M. L. Martins
DCM | FFCLRP | USP

Objetivos de Aprendizagem

- **Compreender o papel das funções de hash criptográficas**
- **Diferenciar criptografia simétrica e assimétrica**, explicando seus princípios, vantagens, limitações e cenários de uso.
- **Explicar como protocolos de autenticação verificam identidades** em sistemas distribuídos, incluindo desafios, nonces e prevenção de ataques.
- **Analisar o problema de gerenciamento de chaves**, comparando abordagens com chave compartilhada, KDC e chaves públicas.

Funções Hash Criptográficas

Entrada: Uma mensagem M (uma sequência de bits de comprimento arbitrário).

Saída: Um valor hash $\mathrm{H}(M)$ (uma sequência de bits de comprimento fixo).

Computação Unidirecional: Dado $\mathrm{H}(M)$, não é possível calcular a mensagem M.

Propriedades Essenciais:

- **Resistência à Pré-imagem:** Não é possível descobrir a entrada original a partir do hash.
- **Resistência à Colisão:** É extremamente difícil encontrar duas entradas diferentes com o mesmo hash.
- **Efeito Avalanche:** Uma pequena mudança na entrada muda completamente o hash.

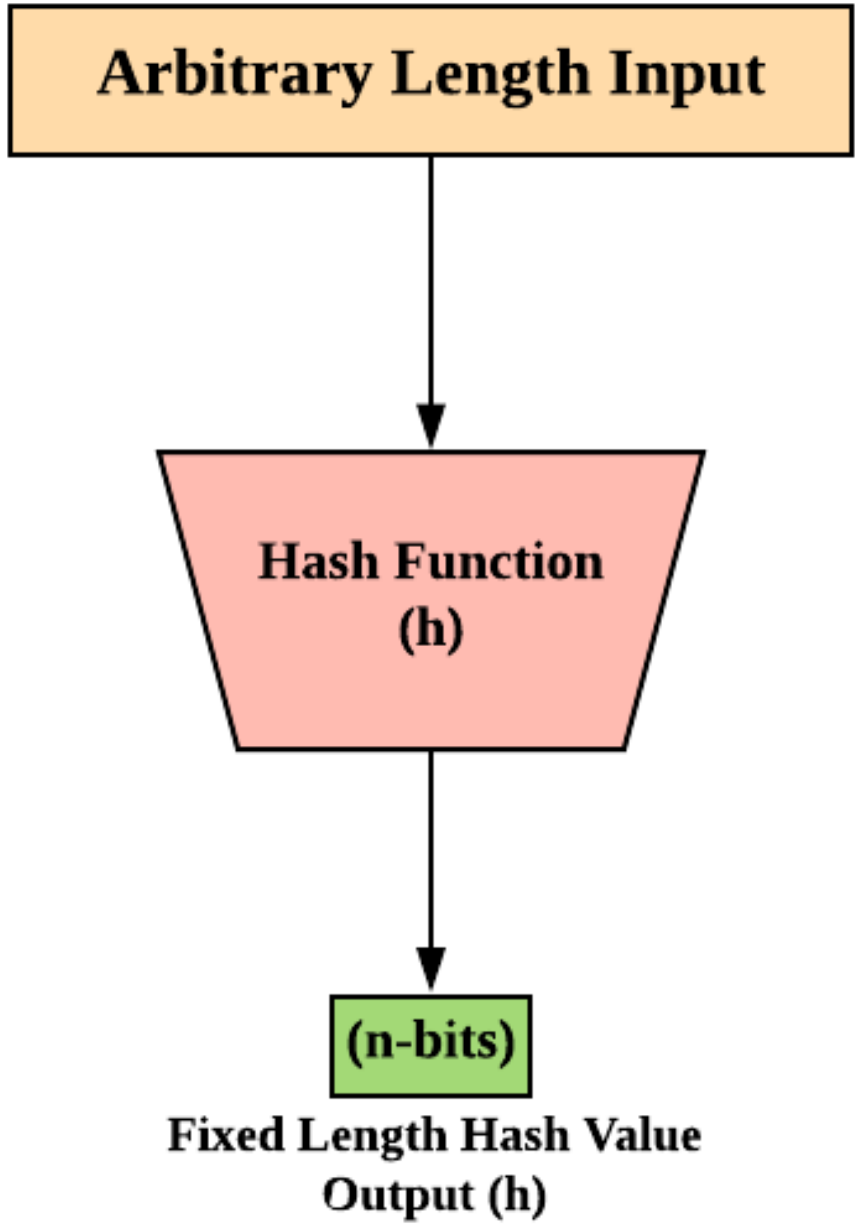
> **Aplicações Práticas**: armazenamento de senhas e verificação de integridade
> **Exemplo**: Verificação de checksum em downloads de ISOs de sistemas operacionais.

MD5, SHA-1, SHA-2 e SHA-3

Família	Observação
MD5	historicamente importante, não usar para segurança
SHA-1	legado, deve ser evitado em proteção criptográfica
SHA-2	família ainda amplamente usada: SHA-256, SHA-384, SHA-512
SHA-3	alternativa moderna baseada em construção diferente

Para projetos novos, prefira SHA-256, SHA-384, SHA-512 ou SHA-3, conforme o contexto.

> Teste algumas funções criptográficas de hash em Hash Function Demo by Data Proof Lab. Fonte da imagem: A Survey of Smart Contracts: Security and Challenges.



Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

O Problema do Ataque de Dicionário

Se dois usuários usam a mesma senha, o hash será igual.

- Atacantes podem usar "Rainbow Tables" (tabelas pré-calculadas) para descobrir senhas comuns.
- Isso torna sistemas vulneráveis se apenas o hash simples for usado.
- Exemplo: Um atacante descobre que o hash **x7y2...** corresponde à senha **123456**.

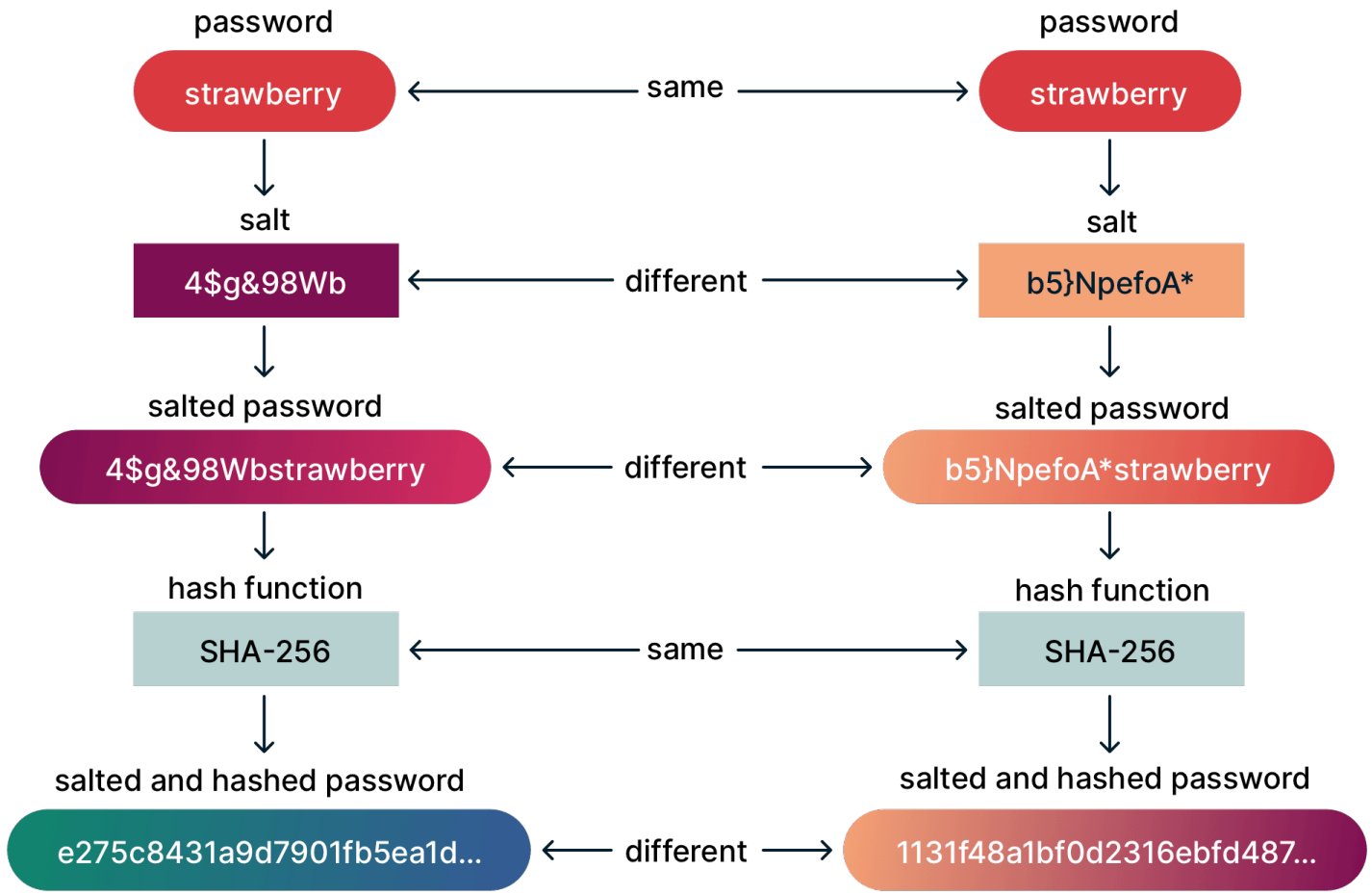
Password	Hash
123456	e10adc3949ba59abbe56e057f20f883e
password	5f4dcc3b5aa765d61d8327deb882cf99
12345	827ccb0eea8a706c4c34a16891f84e7b
12345678	25d55ad283aa400af464c76d713c07ad
football	37b4e2d82900d5e94b8da524fbeb33c0
qwerty	d8578edf8458ce06fbc5bb76a58c5ca4
1234567890	e807f1fcf82d132f9bb018ca6738a19f
1234567	fcea920f7412b5da7be0cf42b8c93759
princess	8afa847f50a716e64932d995c8e7435a
1234	81dc9bdb52d04dc20036dbd8313ed055
login	d56b699830e77ba53855679cb1d252da
welcome	40be4e59b9a2a2b5dffb918c0e86b3d7
solo	5653c6b1f51852a6351ec69c8452abc6
abc123	e99a18c428cb38d5f260853678922e03
admin	21232f297a57a5a743894a0e4a801fc3

> Fonte da Imagem: Javier Santos

Solução 1 para o Ataque de Dicionário

Salt: um valor aleatório adicionado à senha antes do hashing.

- Cada usuário deve ter um Salt único e público no banco de dados.
- Impede o uso de Rainbow Tables, pois cada hash se torna único.



> Fonte da imagem: Cheap SSL Web

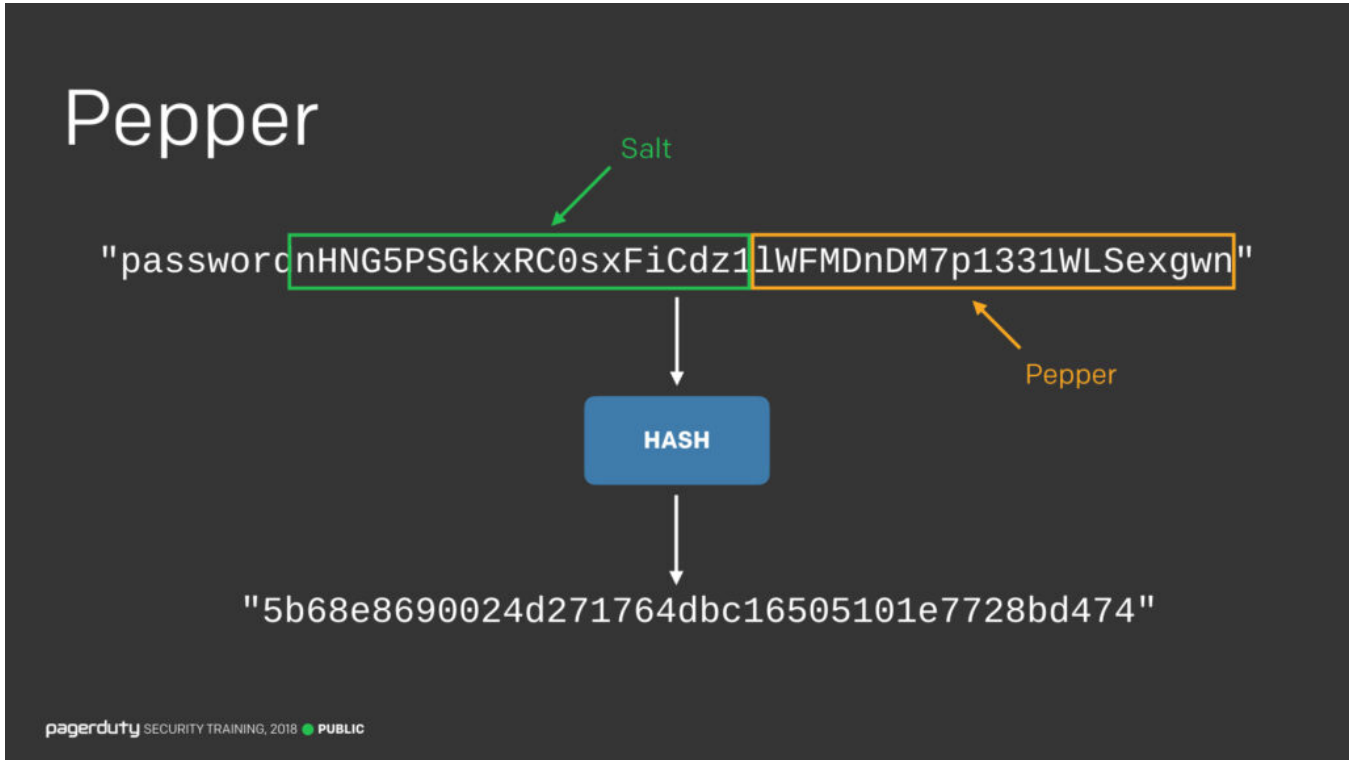
Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Solução 2 para o Ataque de Dicionário

Pepper: é um segredo adicionado ao processo de hash.

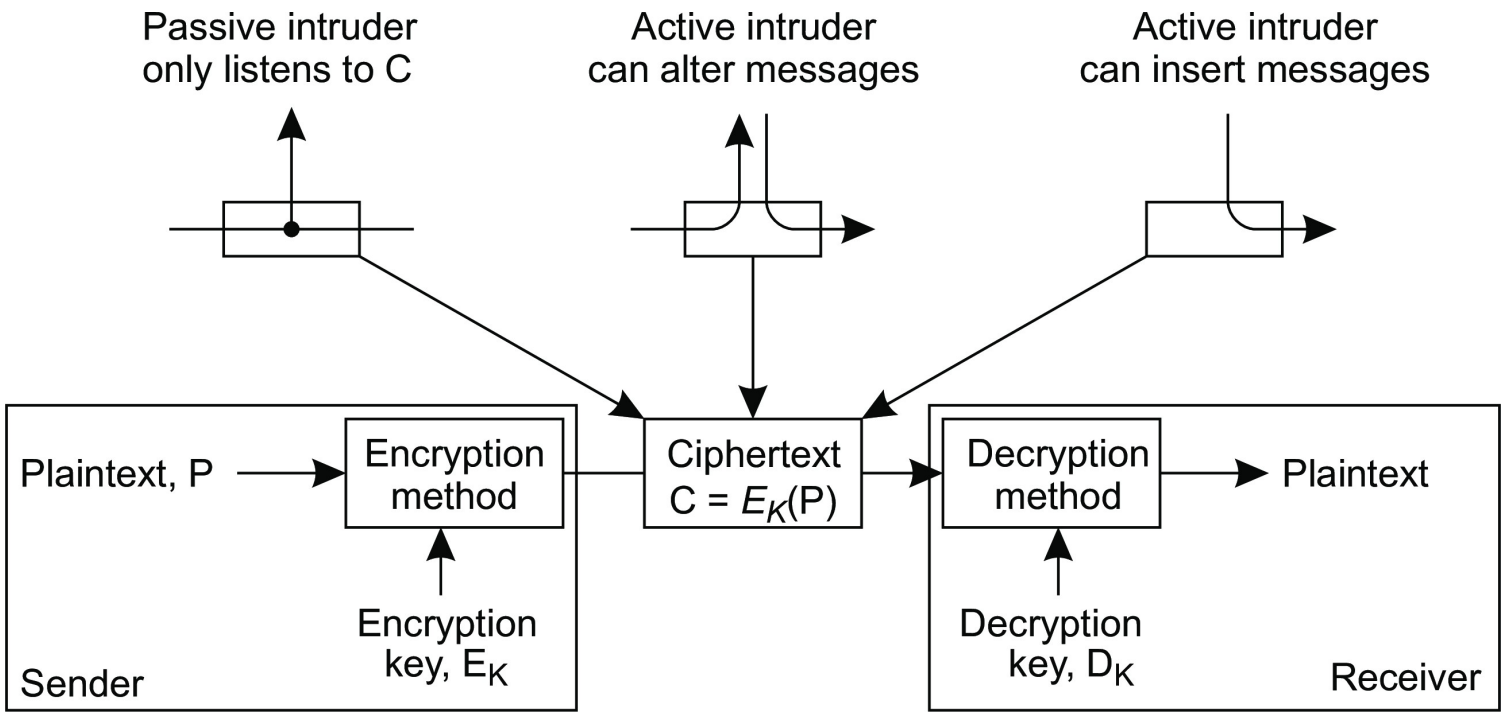
- Diferente do Sal, o Pepper **não** fica no banco de dados; ele fica no código ou em um HSM.
- Adiciona uma camada extra se o banco de dados for vazado.

> Veja também o vídeo Password Hashing, Salts, Peppers | Explained!



Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Criptografia: Modelo Estruturante



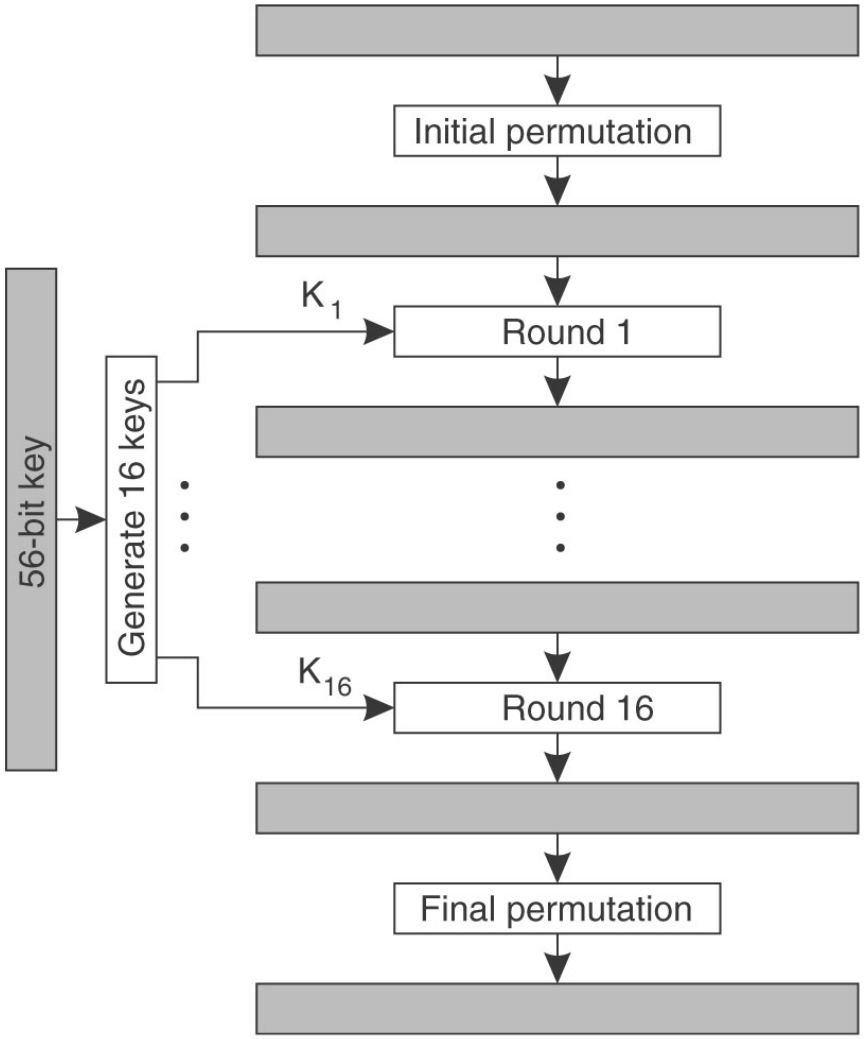
- **Chave de criptografia (Encryption key):** Um *input* E_K para uma função de criptografia, resultando em: $C = E_K(P)$.
- **Chave de descriptografia (Decryption key):** Um *input* D_K para uma função de descriptografia, resultando em: $P = D_K(C)$.

Fonte da Imagem: Van Steen e Tanenbaum. Distributed Systems (livro), 4a edição.

Criptografia Simétrica

- Mesma chave é utilizada para criptografar e decriptografar uma mensagem
- $P = D_K(E_K(P))$
- Na imagem ao lado: Data Encryption Standard (DES).
- O bloco de dados (64 bits) é transformado em um bloco cifrado ao longo de 16 rodadas de transformação utilizando chaves de 48 bits derivadas de uma chave mestre de 56 bits.

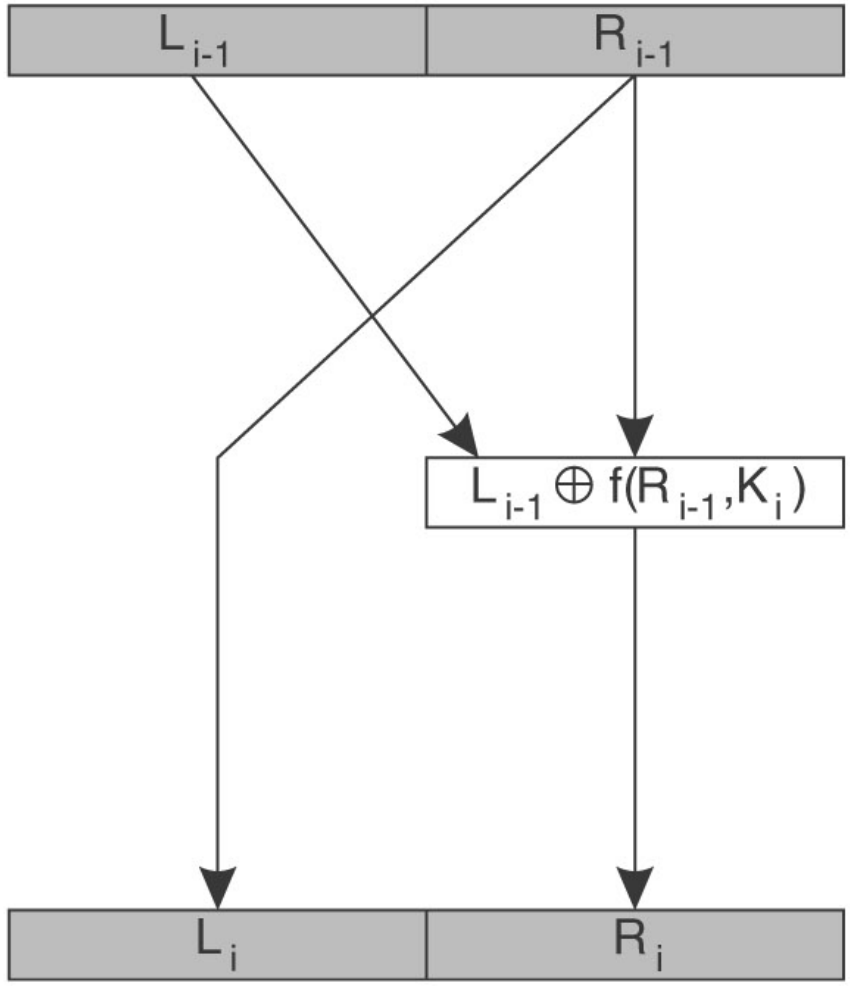
> Para uma visão geral do algoritmo DES, veja o vídeo em Data Encryption Standard (DES).



Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Criptografia Simétrica (cont.)

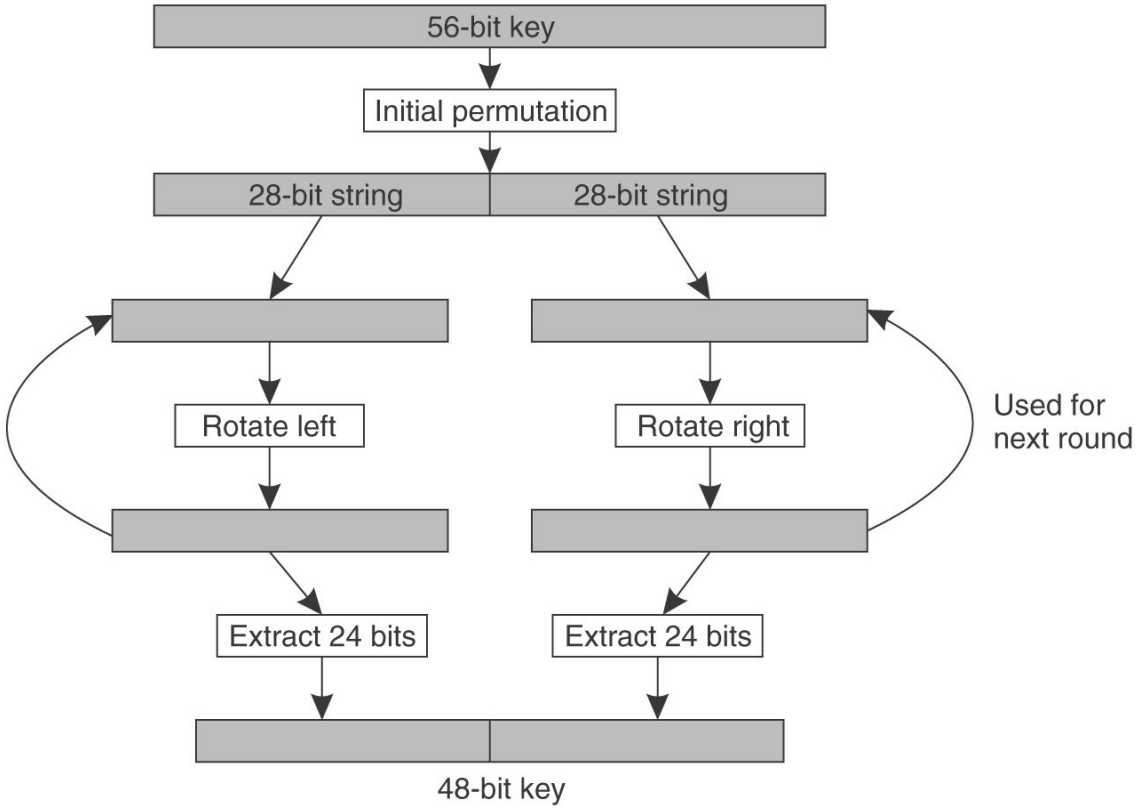
- Os 64 bits são divididos em duas partes de 32 bits cada, L_{i-1} e R_{i-1}.
- A parte à direita é usada como a parte esquerda na próxima rodada $L_i = R_{i-1}$. Também usada como entrada de uma função, conjuntamente com a chave K_i de 48 bits, para produzir um bloco de 32 bits
- O bloco de 32 bits sofre uma operação de ou-exclusivo com o bloco L_{i-1} para produzir o bloco R_i



Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Criptografia Simétrica (cont.)

- A chave K_i de 48 bits para uma rodada i é derivada da chave mestre de 56 bits da seguinte forma:
  - A chave mestre é permutada e dividida em dois blocos de 28 bits cada
  - A cada rodada, cada bloco é rotacionado um ou dois bits para a direita, a partir dos quais 24 bits são extraídos
  - Os 24 bits de cada bloco são então unidos para formar a chave K_i



> Desafio: duas entidades envolvidas na comunicação devem concordar com a mesma chave.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Criptografia Assimétrica

Cada entidade possui duas chaves:

- **Chave pública**: pode ser distribuída;
- **Chave privada**: deve ser protegida.

Com criptografia **simétrica**:

- Como A e B combinam K antes de terem um canal seguro?

Com criptografia **assimétrica**:

- A pode usar a chave pública de B. B usa sua chave privada correspondente.
- Mas surge outro problema: Como A sabe que aquela chave pública é realmente de B?

Algoritmo RSA

Chaves pública e privada são escolhidas a partir de números primos muito grandes.

Chaves são geradas em quatro passos:

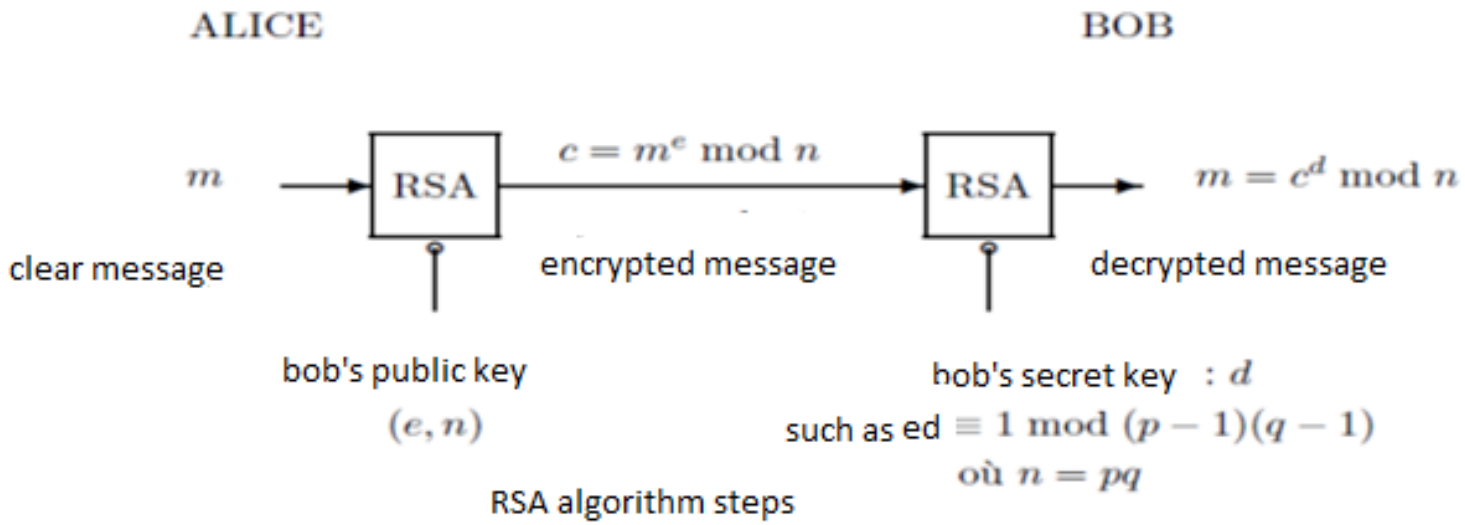
1. Escolher dois números primos grandes, p e q
2. Calcular $n = pq$ e $z = (p-1)(q-1)$
3. Escolher um número e, menor do que n que não tenha fatores comuns (exceto o 1) com z
   - e e z são considerados números primos entre si
4. Achar um número d, tal que $ed = 1 \bmod z$ seja divisível exatamente por z
   chave pública $K+$

- Chave **pública** $K+$: par (n, e)
- Chave **privada** $K-$: par (n, d)

> Geralmente utiliza-se 768 bits para aplicações comuns e 1024 bits para aplicações comerciais.

Algoritmo RSA (cont.)

- Cifragem: $c = m^e \bmod n$
- Decifragem: $m = c^d \bmod n$



> Fonte da imagem: Comparative Study of Encryption Algorithms Applied to the IOT.
> Na prática, RSA não deve ser usado "puro". Usa-se padding e construções seguras, como RSA-OAEP para cifragem e RSA-PSS para assinatura.

Algoritmo RSA: Exemplo

- $p = 3$
- $q = 11$
- $n = p \times q = 33$
- $z = (p - 1) \times (q - 1) = 20$
- Encontre um número primo relativo a $z \rightarrow d = 7$
- Encontre e: $7 \times e \equiv 1 \bmod 20 \rightarrow e = 3$
- $C \equiv P^3 \bmod 33$
- $P \equiv C^7 \bmod 33$

Plaintext (P)			Ciphertext (C)		After decryption	
Symbolic	Numeric	P^3	P^3 (mod 33)	C^7	C^7 (mod 33)	Symbolic
S	19	6859	28	13492928512	19	S
U	21	9261	21	1801088541	21	U
Z	26	17576	20	1280000000	26	Z
A	01	1	1	1	01	A
N	14	2744	5	78125	14	N
N	14	2744	5	78125	14	N
E	05	125	26	8031810176	05	E
Sender's computation				Receiver's computation		

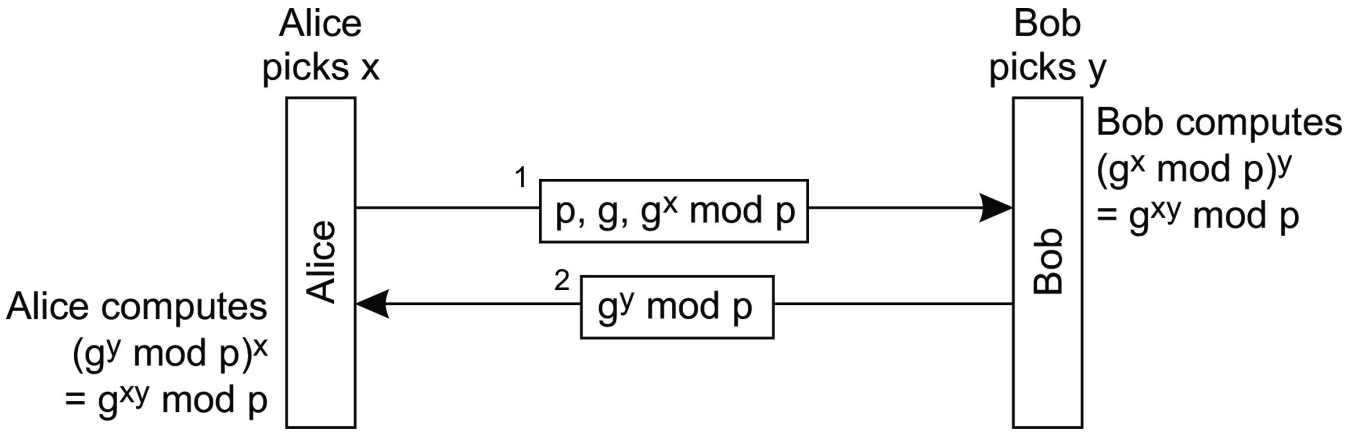
> Fonte da imagem: Prof. Louise E. Moser @UCSB

Troca de chaves: ideia de Diffie-Hellman

Objetivo: Duas partes combinam uma chave secreta mesmo sobre um canal público.

Ideia geral:

- A envia contribuição pública
- B envia contribuição pública
- Ambos calculam o mesmo segredo compartilhado



Atacante observa as mensagens públicas.

- Mas não deve conseguir calcular o segredo.
- Em sistemas modernos, usam-se variantes efêmeras para prover sigilo futuro.

> Fonte da Imagem: Van Steen e Tanenbaum. Distributed Systems (livro), 4a edição.
> Veja mais informações em Wikipedia.

Autenticação: provar identidade

Em protocolos distribuídos, autenticação deve ocorrer antes da troca de informações sensíveis.

Perguntas fundamentais:

- Quem é a outra parte?
- A mensagem é recente?
- A chave pertence mesmo à entidade esperada?
- O protocolo resiste a repetição e reflexão?

Autenticação: Técnicas Simples

Declarar identidade: A → B: "Sou A"

- **Problema**: Atacante → B: "Sou A"
- Não há evidência criptográfica.
- A identidade declarada não prova posse de segredo, chave privada ou credencial confiável.

Verificar IP: B aceita mensagens vindas do IP de A

- **Problemas**: endereços podem ser forjados em certos cenários; IP identifica origem de rede, não necessariamente a entidade; não protege conteúdo da mensagem.

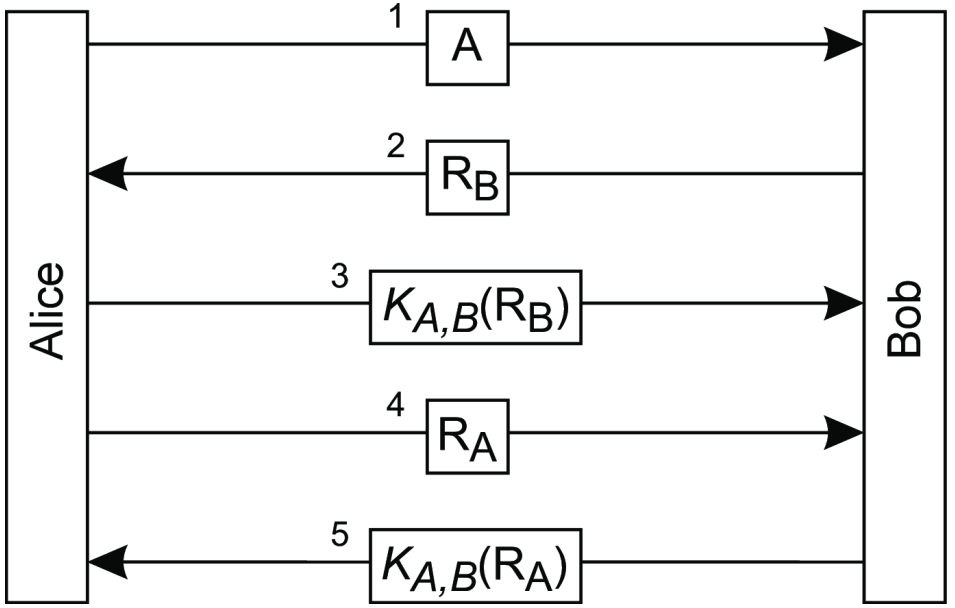
Enviar senha na rede: A → B: usuário, senha

- **Problemas**: senha pode ser interceptada; senha pode ser reutilizada pelo atacante;
- **Melhoria**: nunca enviar o segredo diretamente; provar posse do segredo.

Protocolo Desafio-Resposta com Segredo Compartilhado

A e B compartilham $K_{A,B}$ (segredo, senha compartilhada).

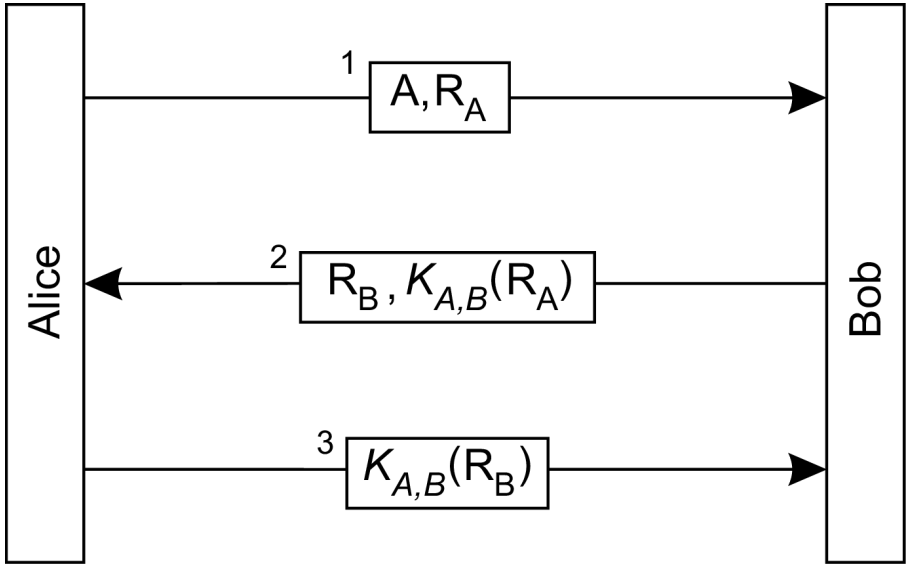
- As entidades envolvidas desafiam-se mutuamente a responder corretamente a um desafio.
- Desafio (e.g., um número) é criptografado e enviado para a entidade desafiante.
  - Se resposta correta, identidade confirmada.



> Fonte da Imagem: Van Steen e Tanenbaum. Distributed Systems (livro), 4a edição.

Protocolo Desafio-Resposta com Segredo Compartilhado

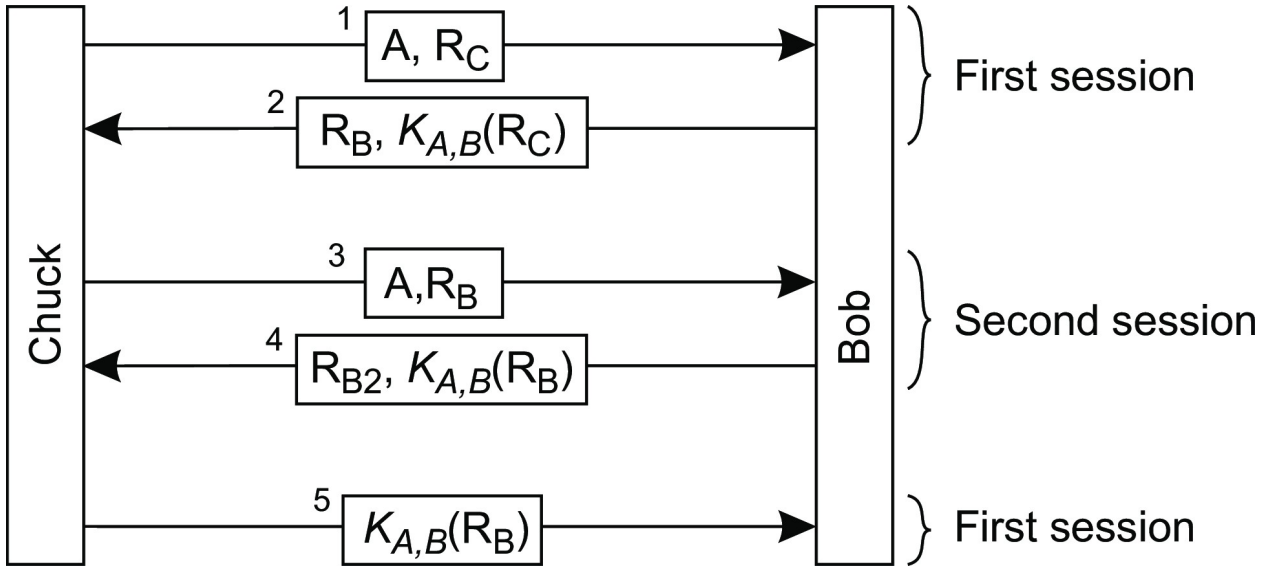
Otimização para reduzir o número de mensagens trocadas?



Fonte das Imagens: Van Steen e Tanenbaum. Distributed Systems (livro), 4a edição.

Protocolo Desafio-Resposta com Segredo Compartilhado

Otimização para reduzir o número de mensagens trocadas?



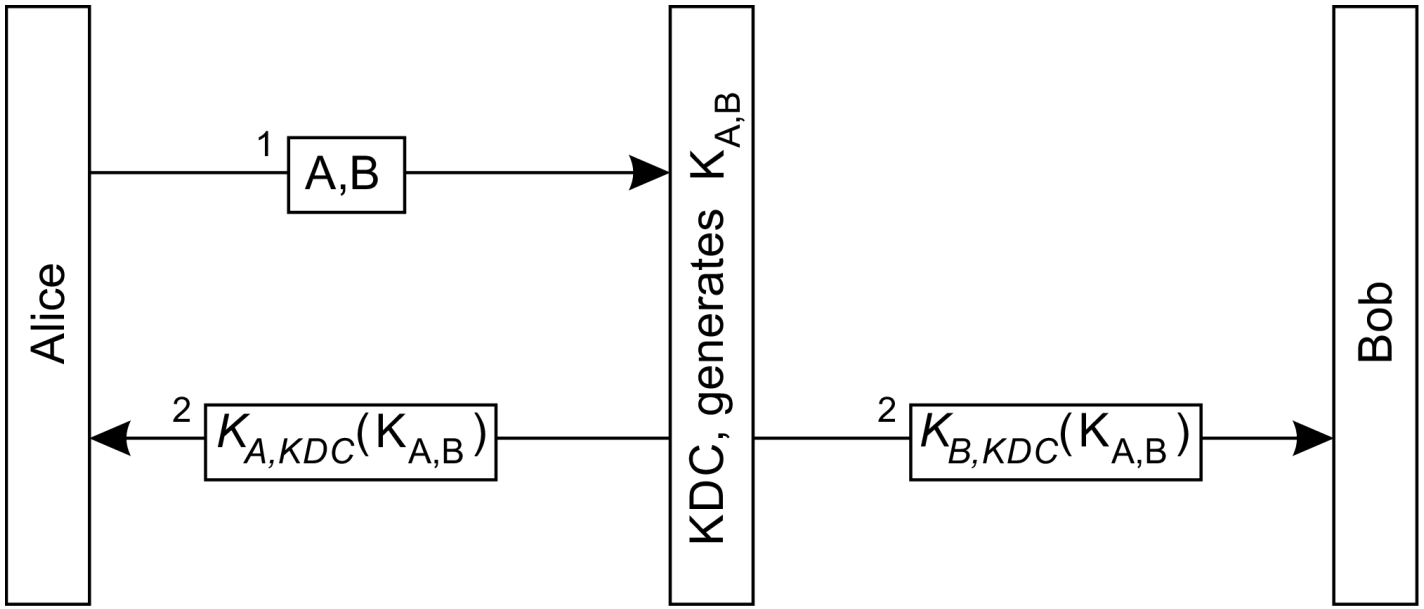
Nova abordagem sofre com ataque de reflexão.

- **Causa típica**: mesmo segredo; ausência de papéis explícitos no protocolo.

> Fonte das Imagens: Van Steen e Tanenbaum. Distributed Systems (livro), 4a edição.

Key Distribution Center (KDC)

Um **KDC** compartilha uma chave secreta com cada entidade.



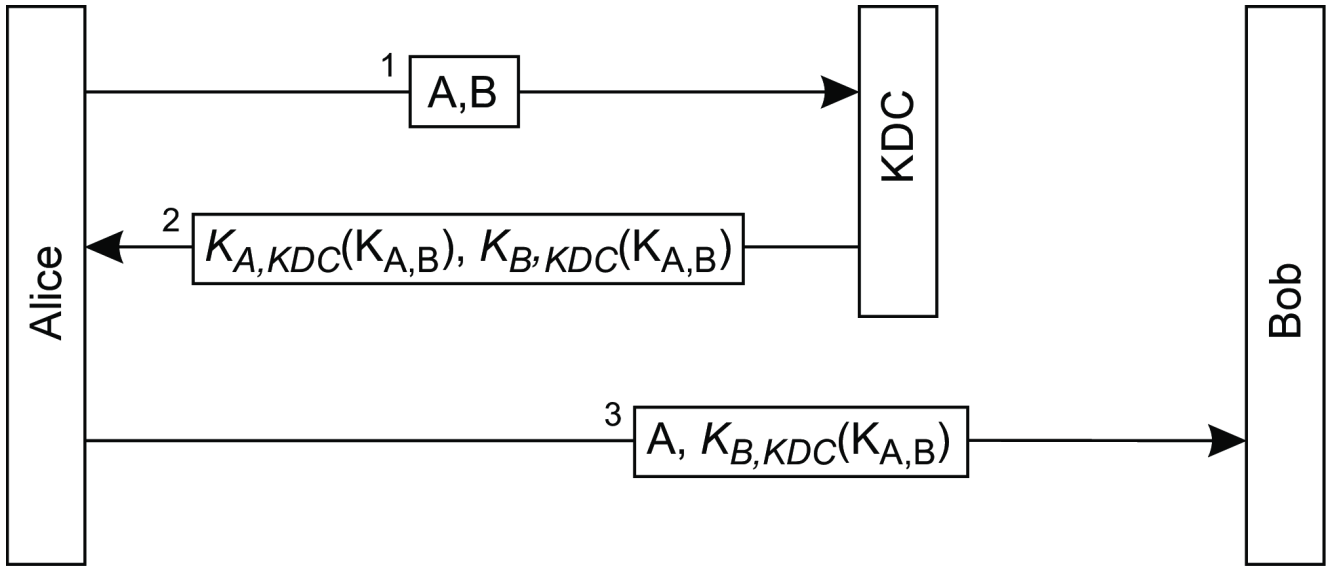
O KDC ajuda a criar uma chave de sessão $K_{A,B}$.

- o KDC é uma autoridade central que emite credenciais temporárias.

Fonte da Imagem: Van Steen e Tanenbaum. Distributed Systems (livro), 4a edição.

KDC com ticket

- **A** informa ao KDC que quer se comunicar com **B**
- O KDC envia uma nova chave secreta, compartilhada por **A** e **B**.
- **A** informa **B** que quer se comunicar e qual a chave a ser utilizada.



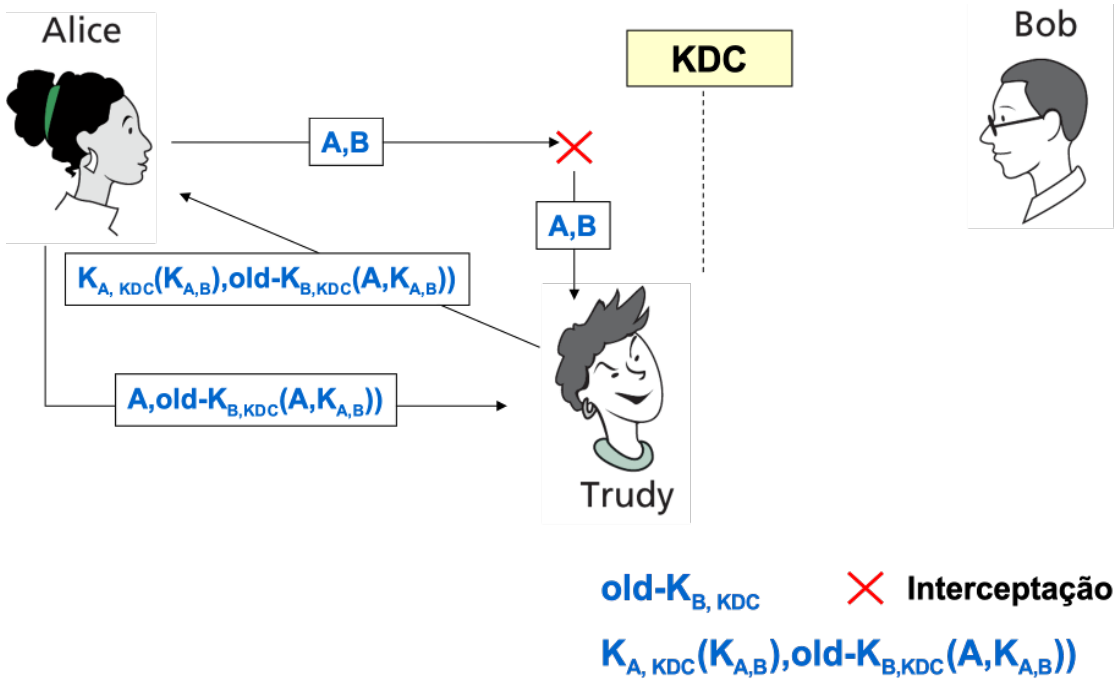
Note que o ticket é protegido com a chave que B compartilha com o KDC. Assim, A não consegue alterar o ticket sem ser detectado. Fonte da Imagem: Van Steen e Tanenbaum. Distributed Systems (livro), 4a edição.

Problema: Replay com mensagens antigas

Se um atacante captura um ticket antigo, B pode aceitar uma sessão antiga se o protocolo não verificar frescor.

Soluções:

- **nonces**;
- timestamps;
- validade curta;
- identificador de sessão;
- autenticação adicional de posse da chave de sessão.



Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

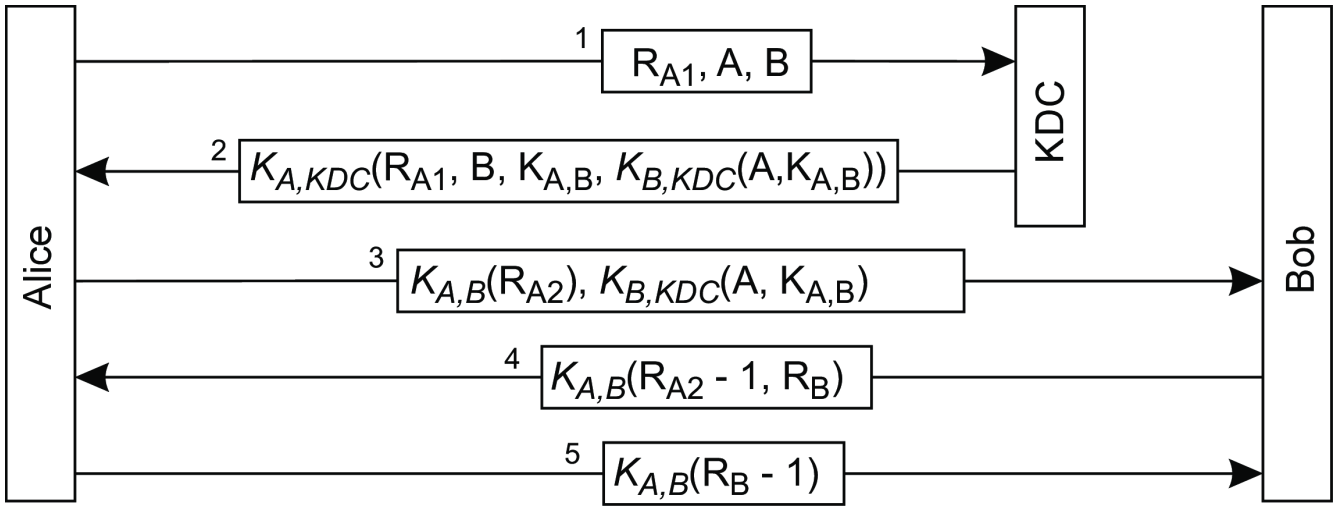
Nonce: um valor usado uma única vez

Objetivos:

- ligar uma resposta a uma requisição específica;
- impedir reuso de mensagens antigas.

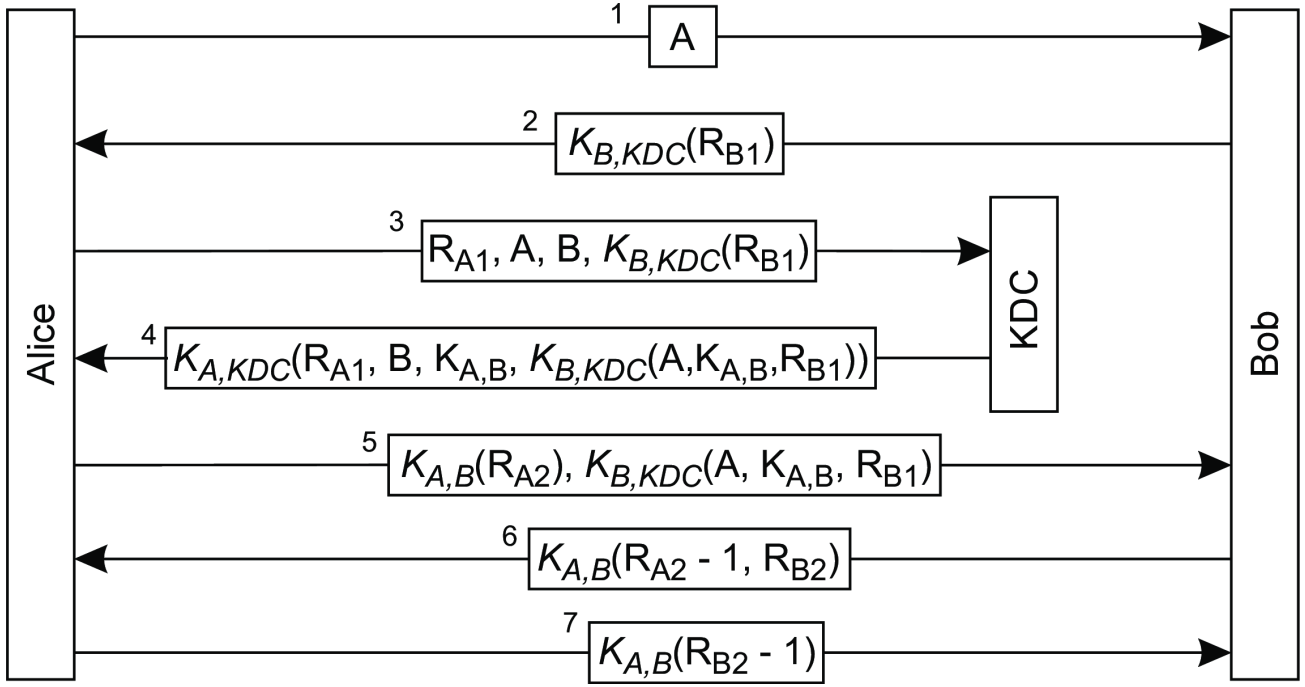
Problema: Se o invasor capturar uma chave $K_{A,B}$ antiga e a mensagem 3, o invasor pode reenviar a mensagem 3 para B se fazendo passar por A.

- **Solução**: nonce deve ser gerado por B.



Fonte: Van Steen e Tanenbaum. Distributed Systems (livro), 4a edição.

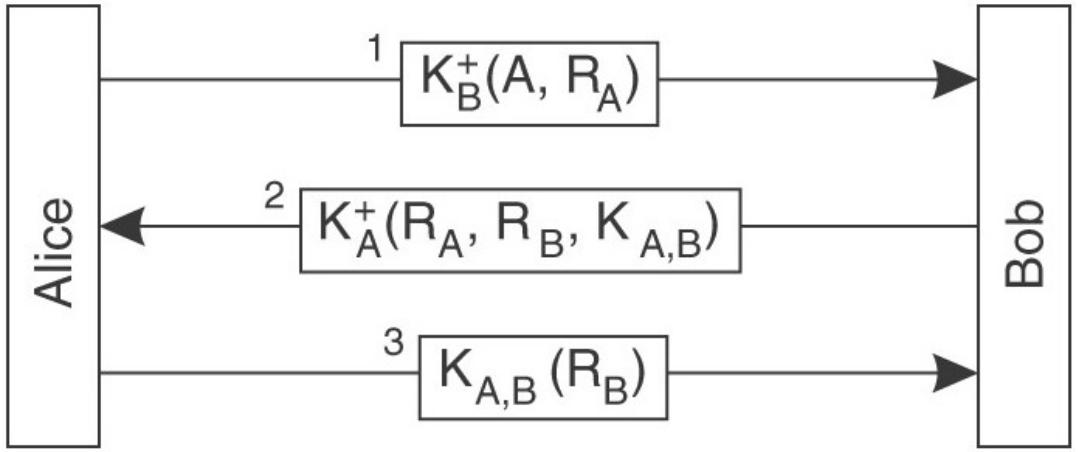
KDC: Proteção contra reuso de chaves



Note que, retornando $R_{A2} - 1$ em #6, **B** prova que conhece $K_{A,B}$.

Autenticação com chave pública

- Protocolo que utiliza as chaves pública e privada de uma entidade para a autenticação.
- Entidade que inicia a autenticação envia um desafio para uma entidade destino
  - Mensagem é criptografada com a chave pública da entidade destino



- Entidade destino decifra a mensagem e retorna o valor do desafio para a entidade origem conjuntamente com seu próprio desafio (uma chave secreta $K_{A,B}$ também é retornada na mensagem)
- Mensagem é criptografada com a chave pública da entidade origem

Public Key Infrastructure (PKI) e Certificados Digitais

Problema: Esta chave pública pertence mesmo a usp.br?

Solução:

- certificado digital vincula identidade a chave pública;
- autoridade certificadora assina o certificado;
- clientes verificam cadeia de confiança.

Em TLS, certificados são fundamentais para autenticar servidores.

> Veja mais em Wikipedia.

Rotação e revogação

Princípios:

- Rotação reduz impacto de vazamentos.
- Revogação remove confiança em chaves comprometidas.

Desafios em sistemas distribuídos:

- múltiplas réplicas;
- caches de certificados;
- clientes offline;
- logs cifrados com chaves antigas;
- compatibilidade durante migração.

Conclusão

- Hash garante resumo, não confidencialidade.
- Assinatura digital permite verificação pública.
- Criptografia simétrica é rápida, mas exige distribuição segura de chaves.
- Criptografia assimétrica ajuda em autenticação, assinatura e acordo inicial de chaves.
- Autenticação exige prova de identidade e frescor.
- Gerenciamento de chaves é parte essencial da segurança.

Leitura Adicional

- Livro Tanenbaum: Capítulo 9 (**importante!**)
  - Criptografia: seção 9.1.3
  - Autenticação: seção 9.2.1
  - Integridade e Confidencialidade: seções 9.2.2 e 9.4.1
- Veja também o material de aula de Jens Lechtenbörger
- Python Security: Exemplos de implementação em Python.