Gerenciamento de Chaves Simétricas

Capítulo 3

Introdução

- A criptografia de chave simétrica pode manter seguro seus segredos,
- mas pelo fato de **precisarmos das chaves para recuperar os dados criptografados**, devemos mantê-las seguras.

Introdução

- **O processo para manter todas as chaves seguras e disponíveis** é conhecido como **gerenciamento de chaves**.

- Este capítulo é sobre **gerenciamento de chaves simétricas**.

Introdução

- Pao-Chi gerou uma chave aleatória ou pseudo-aleatória e a utilizou para **criptografar** os dados.

- Se quiser **decriptografar** os dados, ele deve utilizar a mesma chave.

Introdução

- Então, ele precisa armazená-la em algum lugar seguro, de modo que possa recuperá-la novamente quando precisar.

Introdução

- Soluções para o **armazenamento de chaves** podem ser dispositivos pequenos, projetados para proteger chaves ou senhas.

Introdução

- Ou utilizar **criptografia de chave simétrica** para proteger os megabytes de informação e alguma ou técnica para proteger chaves.

Criptografia baseada em senha

- A chave usada para **criptografar** os megabytes de informação (dados em grande quantidade) é conhecida como ***chave de sessão***.

- Uma ***sessão*** é uma instância de criptografia.

Criptografia baseada em senha

- **Exemplos de sessão:**
  - troca de email
  - uma conexão Web
  - um armazenamento de BD
  - **criptografia de um arquivo** antes de armazená-lo na unidade de disco.

Criptografia baseada em senha

- No caso, podemos supor que Pao-Chi, faz uma sessão para criptografar um arquivo antes de armazená-lo na sua unidade de disco.

Criptografia baseada em senha

- Alguns sistemas geram uma nova chave para cada sessão.
- Outros utilizam a mesma chave em diferentes sessões.

Criptografia baseada em senha

- Um modo de armazenar a **chave de sessão** de forma segura é **encriptá-la usando um algoritmo de chave simétrica**.

Criptografia baseada em senha

- A chave fica criptografada.
- Um invasor precisa quebrar a criptografia para conseguir a **chave de sessão**.

Criptografia baseada em senha

- Então, o processo de **criptografar a própria chave de sessão** precisa de outra chave.

- Há então, a **chave de criptografia de chave** (*key encryption key* – KEK)

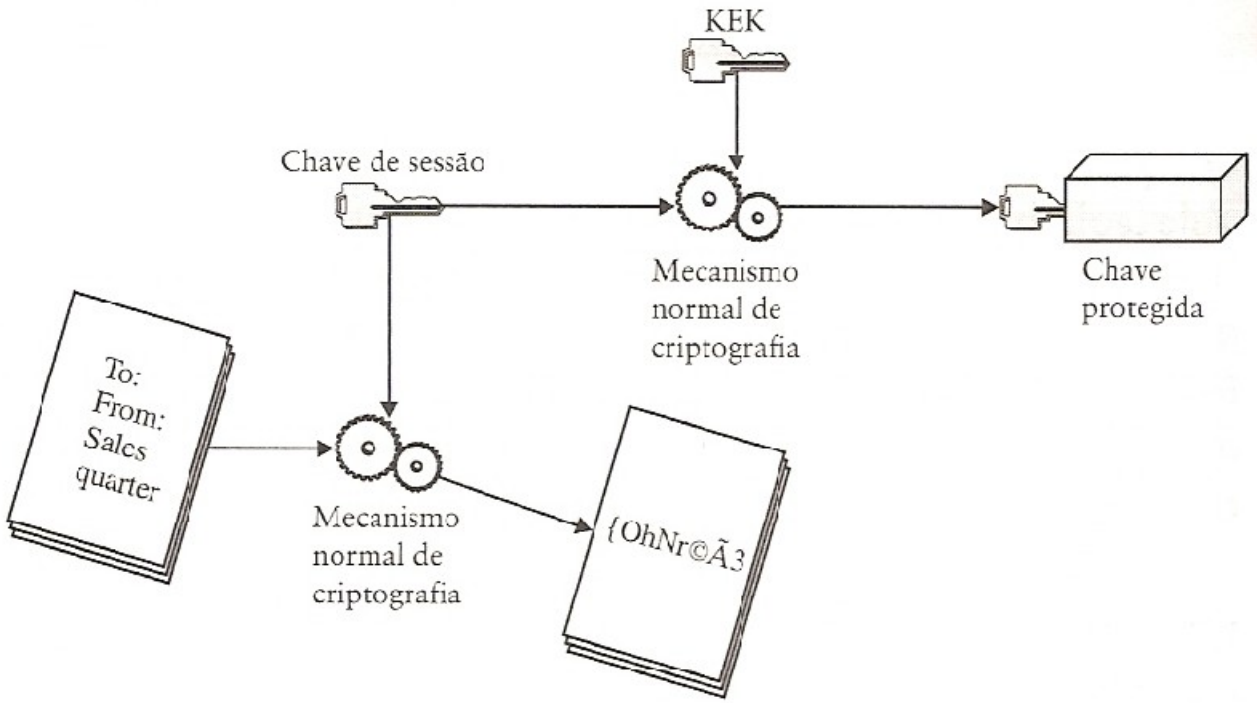
Criptografia baseada em senha

- Para **criptografar**, com a **KEK**, a **chave de sessão**.
- A **chave de sessão**, então, fica protegida e é armazenada.

Criptografia baseada em senha

FIGURA 3-1

Umc chave de sessão protege os dados e uma *chave de criptografia da chave* (key encryption key – KEK) protege a chave de sessão



Criptografia baseada em senha

- Pode-se então, pensar que se Pao-Chi utilizar uma KEK, ele agora tem que armazená-la e protegê-la.

- Na verdade, **ele não precisa armazenar a KEK**. Assim, **ele não precisa protegê-la**.

Criptografia baseada em senha

- **KEK não precisa ser protegida.**
- Ao se precisar **critografar uma chave de sessão**, a KEK é gerada, utilizada e descartada.
- Para **decriptografar a chave de sessão**, KEK é novamente gerada, utilizada e descartada.

Criptografia baseada em senha

- Como a criptografia da chave de sessão é simétrica, a mesma KEK precisa existir depois de descartada.

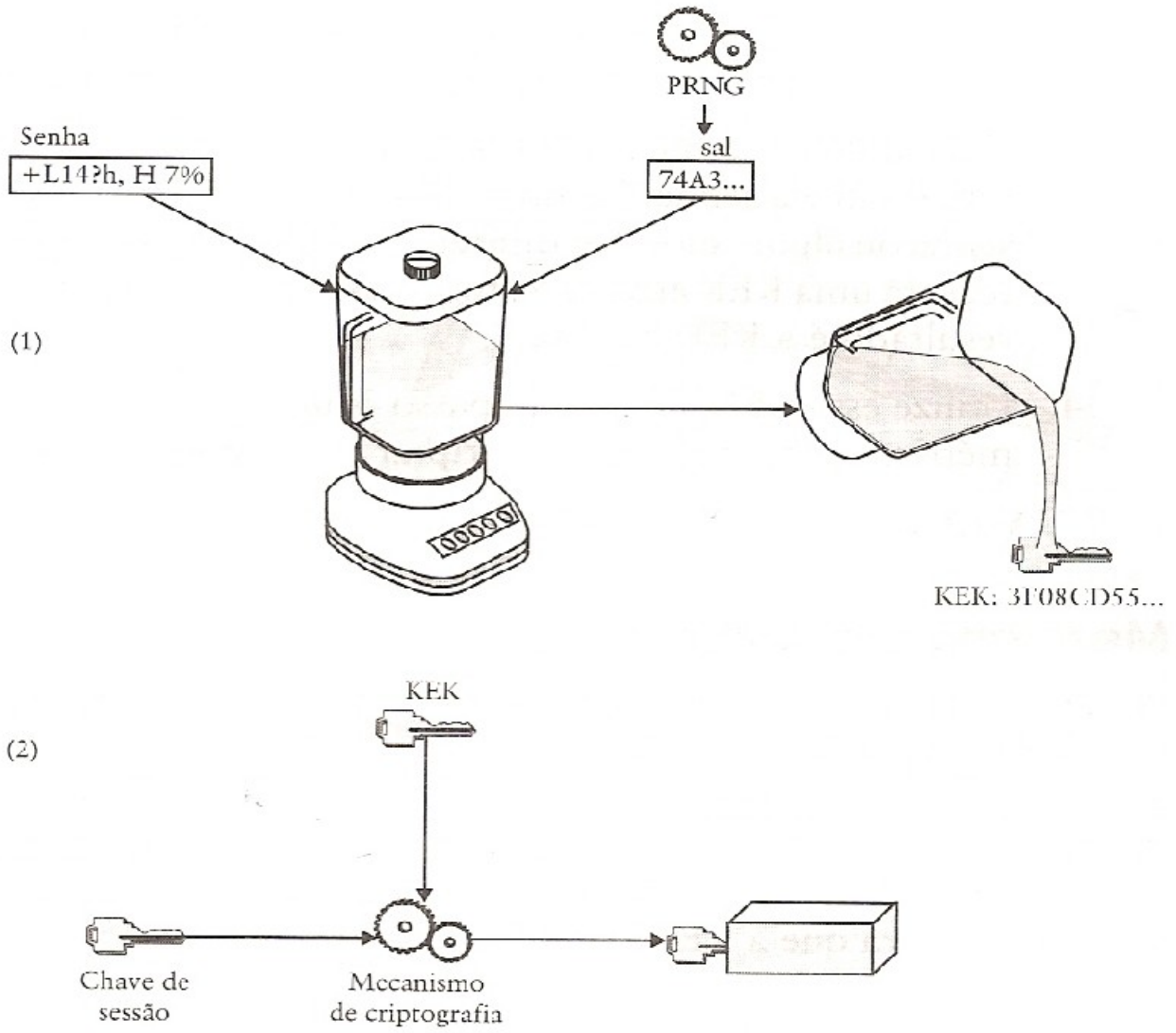
- A geração pela segunda vez, produzindo o mesmo valor anterior é possível porque **KEK está baseada em senha**.

Criptografia baseada em senha

- Pao-Chi usa um **RNG** ou um **PRNG** para **gerar uma chave de sessão**.
- Pao-Chi utiliza a **criptografia baseda em senha** (password-based encryption - **PBE**) para **construir uma KEK**.

FIGURA 3-2

Na *criptografia baseada em senha* (password-based encryption – PBE), (1) mescle a senha e o salt para formar uma KEK e então (2) utilize-a para encriptar a chave de sessão. Para decriptar os dados, utilize a mesma senha e o mesmo salt



Criptografia baseada em senha

- A **criptografia baseada em senha** para construir uma **KEK** é conseguida, fazendo-se um "resumo de mensagem", através de uma função *hash*, a partir de uma **senha** pré-estabelecida, e um "**salt**".

Criptografia baseada em senha

- **PBE** é uma maneira possível para **proteger chaves criptográficas**.
- Outra maneira: **armazenamento de chave baseado em hardware**.

Criptografia baseada em senha

- A **senha** é pré-estabelecida e pode ser descoberta.

- Se a **senha** for descoberta, a KEK, certamente será descoberta, e consequentemente, a **chave de sessão** pode ser decriptografada, sendo então, também, descoberta.

Criptografia baseada em senha

- Se a **chave de sessão** for descoberta ... ...
- Mega-bytes de informação serão descobertos.

Criptografia baseada em senha

- Uma **senha** é composta de caracteres associados às teclas de um teclado, o que não é suficientemente aleatório.

- Assim, uma **senha** não tem muita **entropia** (medida de aleatoriedade).

Criptografia baseada em senha

- Um invasor poderia construir um **dicionário de senhas**.
- Logo, não é bom usar uma senha **como a única coisa** para gerar uma KEK.

Criptografia baseada em senha

- Então, gerar um número PRNG, que será chamado ***salt***, que existe para evitar pré-computações de um invasor que deseje criar um **dicionário de senhas** comuns e **chaves KEK associadas**.

Criptografia baseada em senha

- Evitando-se a construção do dicionário, **evita-se um ataque de dicionário**.

- Utilizando-se um **algoritmo de *hash* (mistura)** (em Comp. *merge* = intercalar) na senha e no *salt*, assegura-se que **uma KEK seja mais aleatória do que a senha.**

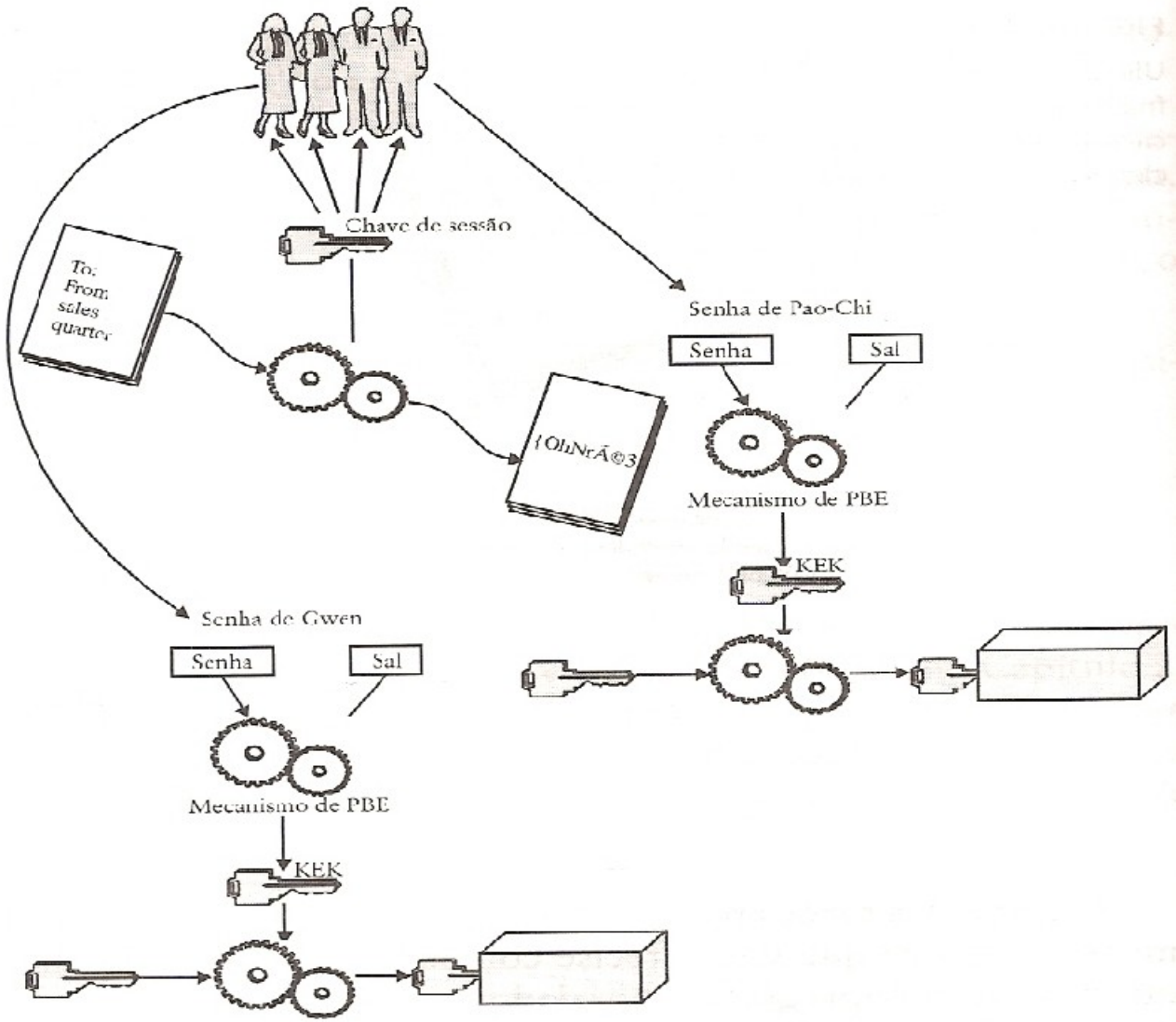
Criptografia baseada em senha

- Um ***salt*** não é secreto. Evita apenas pré-computações. Mesmo não sendo secreto ele realiza sua função.

- Se fosse secreto, como seria recuperado?

- Um ***salt*** não adiciona segurança. Apenas **evita ataque de dicionário**.

FIGURA 3-4
Utilizando uma chave de sessão para dados em grande quantidade e protegendo-os com uma PBE, os usuários não precisam compartilhar as senhas



Razões para usar duas chaves

- **(1)**

- **Compartilhar os dados com outras pessoas** e mantendo criptografados os dados armazenados.

Comentando (1)

- Neste caso, é gerada uma chave de sessão e todo mundo obtém uma cópia dela (COMO ?).

- Todo mundo, então, protege sua cópia, utilizando uma PBE.

Comentando (1)

- Cada um tem sua senha, gera um *salt* e com um determinado algoritmo de *hash*, uma KEK é gerada.

- Essa KEK é diferente, para cada pessoa. A chave de sessão é criptografada e armazenada junto com o *salt*. (Fig.3-6)

Comentando (1)

- **A KEK é descartada**.
- Para **descriptografar** a chave de sessão, cada pessoa tem que ter os mesmos elementos (**sua senha**, o **salt armazenado** e o **algoritmo de *hash***)

Comentando (1)

- Com **sua senha** e o **salt armazenado**, calcule o hash para obter o que presumivelmente será a sua KEK inicial.
- A chave de sessão pode, então, ser decriptografada.

Comentando (1)

- Conclusão:

- **Utilizando uma chave de sessão** para dados em grande quantidade e protegendo-os com uma PBE, **os usuários não precisam compartilhar senha**.

Razões para usar duas chaves

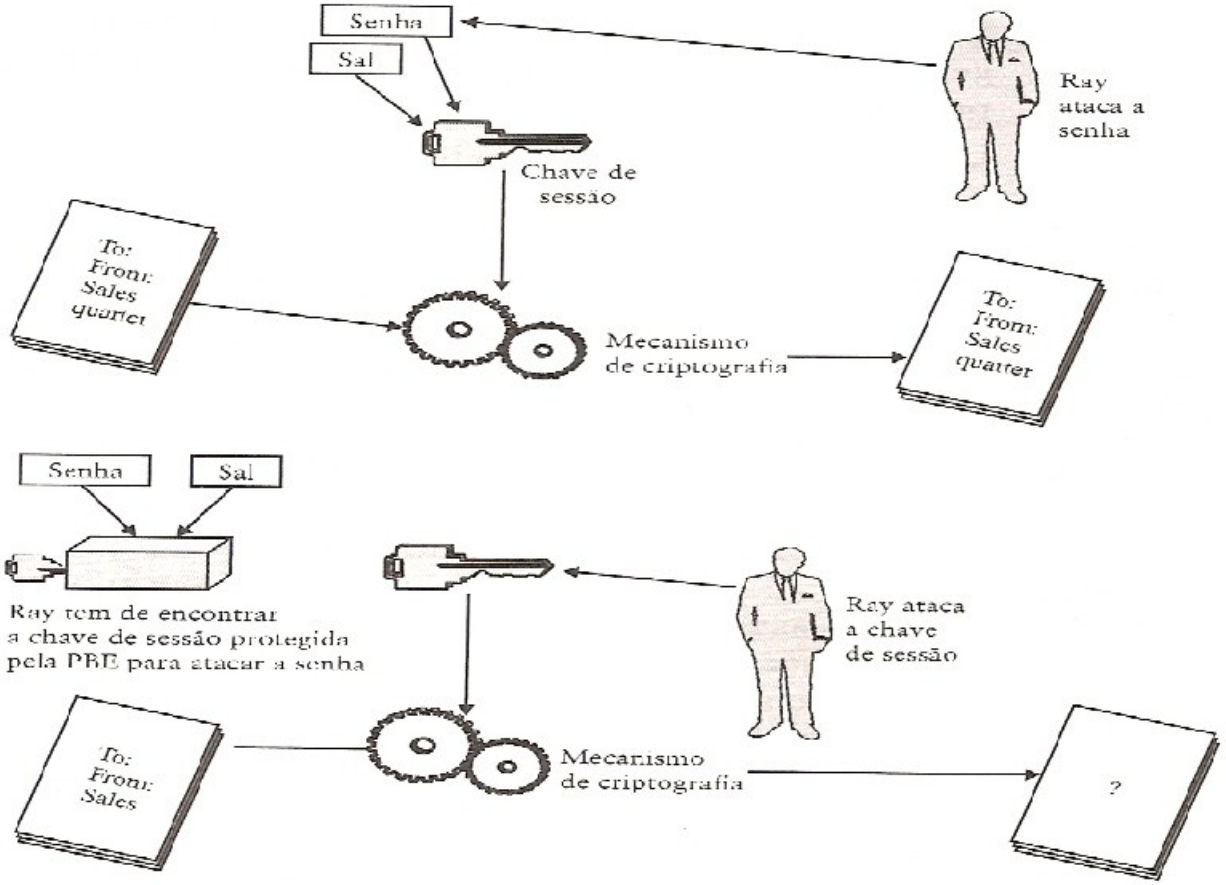
- **(2)**
- **É <u>mais fácil quebrar uma senha</u> do que uma chave**.
- Talvez um invasor tenha acesso mais fácil aos dados criptografados do que à chave criptografada.

Comentando (2)

- Se Pao-Chi utilizar **PBE para proteger os dados em grande quantidade**, Ray, um invasor, pode atacar e quebrar a senha (**PBE é baseado somente na senha**) para recuperar os dados.

Comentando (2)

FIGURA 3-5
Se Pao Chi utilizar PBE para proteger os dados em grande quantidade, Ray pode recuperá-los quebrando a senha. Se Pao-Chi utilizar PBE para proteger a chave de sessão, Ray devo encontrar a chave encriptada



Comentando (2)

- Se Pao-Chi utilizar **PBE para proteger a chave de sessão**, Ray, o invasor, atacará e encontrará a chave de sessão criptografada, o que faz com que ele tenha de quebrar essa criptografia para chegar na chave de sessão.

Conveniência de Programação

- Um programa de PBE fará seu trabalho, mesmo com uma senha errada.

- Se uma senha errada foi inserida, o programa não tem como saber se a senha não é a correta.

Conveniência de Programação

- Ele mesclaria a senha incorreta com o salt e geraria uma KEK incorreta.
- Mas, o programa PBE não teria como saber isso.
- Ele usaria esta KEK para decriptografar a chave de sessão. Algum valor sairia como resultado.

Conveniência de Programação

- Seria uma chave errada.
- Essa suposta chave de sessão seria usada para decriptografar o texto cifrado.
- Os dados resultantes não teriam sentido.

Conveniência de Programação

- Somente neste ponto é que seria possível que algo saiu errado.
- Todo os dados em grande quantidade tiveram de ser descriptografados, antes de se descobrir o erro.

Conveniência de Programação

- Assim, é conveniente se, ao se inserir uma senha, houvesse alguma maneira para se saber se tal senha é correta ou não.

Conveniência de Programação

- Antecipando algo que saiu errado ...
- Uma solução é utilizar a KEK para **criptografar a chave de sessão**, juntamente com **algum valor reconhecível**, como o ***salt***.

Criptografando dados em grande quantidade

1. Gerar uma chave de sessão aleatória ou pseudo-aleatória.

2. Utilize essa **chave de sessão para criptografar os dados**.

3. Insira uma senha, gere um *salt.* Mescle os dois para obter uma KEK.

Criptografando dados em grande quantidade

4. Criptografe a **chave de sessão e o salt** utilizando a KEK.

5. Armazene os **dados criptografados** com o ***salt***.

6. Armazene a chave de sessão e o salt, criptografados com KEK.

Decriptografando dados em grande quantidade

1. Colete o salt e a senha. Mescle os dois para obter o que se presume ser a KEK.

2. Utilizando a KEK, descriptografe a **chave de sessão junto com o salt**.

Descriptografando dados em grande quantidade

3. Verifique o salt descriptografado. É o correto ?

4. Se for o *salt* correto, utilize a chave de sessão para descriptografar os dados.

Descriptografando dados em grande quantidade

5. Se não for o *salt* correto, o usuário provavelmente inseriu a senha errada.

Boas senhas

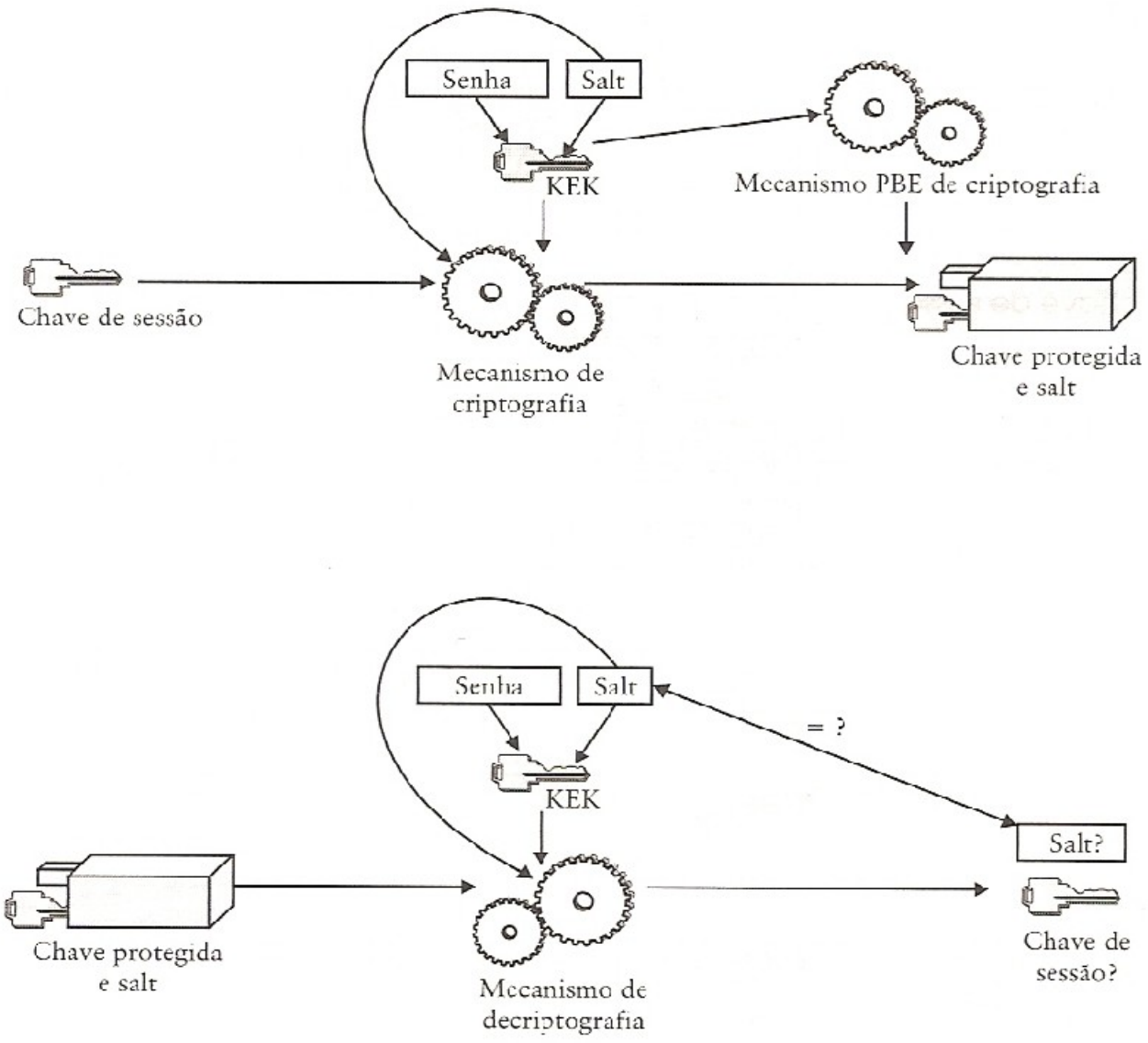
- Ao escolher uma senha, seu objetivo é escolher uma que resista a um ataque de força bruta.

- Exemplo:
  14G:c*%3<wM*-16]Bnp?~d86

- Se estivermos **usando PBE**, **precisamos de uma boa senha**.

FIGURA 3-6

Utilizo uma KEK para encriptar a chave de sessão junto com um valor reconhecível como o salt. Inserir a senha errada produz uma combinação errada de KEK/salt



Geradores de senha

- Existem programas que geram senhas.
- **Funcionam como PRNG**, mas produzem **pressionamentos de teclas** em vez de números.
- Podem especificar a duração da senha.

Geradores de senha

- Ver em JavaScript Source
  http://javascript.internet.com/

  tiFXFCZcZ6
  K6($xV]!hl
  M?a84z9Wg

- Mas, certifique-se de que o programa é confiável.

Exemplos práticos

- Na prática, como as empresas protegem suas chaves ?
- Uma classe de produtos para proteção de chaves de sessão são aplicativos de criptografia de arquivo.

Exemplos práticos

- **Arquivos** são criptografados na unidade de disco, utilizando **criptografia de chave simétrica**.

- A proteção das chaves criptográficas em grande quantidade pode ser feita de várias maneiras.

Aplicativos para criptografia de arquivo

- PGP (Pretty Good Privacy)
  - **criptografia de arquivo por PBE**
  - "envelopamento" para **criptografia avançada de arquivo**.

- Outros (freeware, shareware, produtos convencionais)

Aplicativos para criptografia de arquivo

- Keon Desktop

  A **RSA Security** produz uma família de produtos chamada de **Keon**.

Armazenamento de chaves em HW

- **PBE** é uma maneira possível para **proteger chaves criptográficas**.

- Outra maneira: **armazenamento em um dispositivo de HW**:
  - tokens
  - aceleradores de criptografia

Tokens

- Um cartão plástico “inteligente”
- Uma chave plástica
- Um pequeno anexo da porta USB
- Um anel de dedo

Tokens

- Contém um pequeno chip com um processador, um tipo de sistema operacional apropriado e recursos limitados de I/O, memória e espaço de armazenamento em disco.

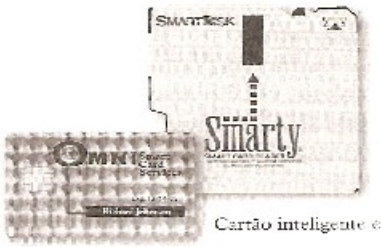
Alguns Tokens

FIGURA 3-7
Alguns tokens

Anel Java

IKey 2000

Datakey



Cartão inteligente e leitor Smarty

Cartão inteligente RSA SecurID 3100

Tokens

- A vantagem de se utilizar tokens é que **um invasor não tem acesso a eles**.

- Quando precisar utilizar uma chave simétrica, transfere-se do token para o computador, que a utiliza para encriptar ou decriptar os dados.

Cartões inteligentes

- É simplesmente um cartão de plástico, que contém um microprocessador.
- ISO publicou vários padrões descrevendo as características físicas.
- A idéia é todos os cartões inteligentes serão utilizados com uma ampla gama de leitores.

Tokens USB

- USB (Universal Serial Bus) é um padrão da indústria para conectar dispositivos Plug-and-Play.
- Não é necessário reinicializar o SO depois de inserir o dispositivo.
- Desde que o software esteja instalado, simplesmente conecta-se o dispositivo e trabalha-se com ele.

Tokens USB

- Têm um pouco mais de poder de computação e espaço de armazenamento do que os cartões inteligentes.

Tokens como dispositivos de armazenamento de senha

- Tokens podem armazenar senhas, de várias contas aos quais uma pessoa possa se conectar.
- Utilizar um token para gerar grandes senhas aleatórias e para armazenar essas senhas.
- Não é preciso lembrar da senha.

Tokens como dispositivos de armazenamento de senha

- Quando precisar conectar-se a uma conta, conecta-se o token e faz-se com que ele envie a senha.
- O acesso ao token é por senha.
- Utilizar um token ajuda a prevenir de um ataque remoto.

Aceleradores de criptografia

- Dispositivos de criptografia baseados em hardware.
- Chips especializados em criptografia.
- Mais rápidos que microprocessadores de uso geral.
- Pode armazenar dados de maneira mais segura que um computador.

Aceleradores de criptografia

- Funcionam o tempo todo conectados, internos ou externos ao computador (PC).
- Com o PC convencional, a unidade de disco é visível ao mundo exterior.
- Invasores podem ler a unidade de disco.

Aceleradores de criptografia

- Mesmo com *firewalls* para informações sigilosas, invasores podem utilizar ferramentas como software de recuperação de dados.

Aceleradores de criptografia

- Aceleradores de criptografia são construídos de tal modo que seu espaço de armazenamento não é visível.

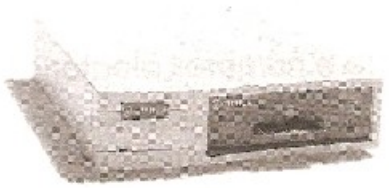
- Se um invasor espionar, abrindo a parte externa, para fisicamente acessar a unidade de disco, o dispositivo apaga a si próprio.

Aceleradores de criptografia

- Gerenciador de chave e acelerador de criptografia **nShield**.
- Cryptoswift PCI E-Commerce Accelerator.
- Luna CA^3
- AXL 300

Aceleradores de Criptografia

FIGURA 3-8
Alguns aceleradores de criptografia

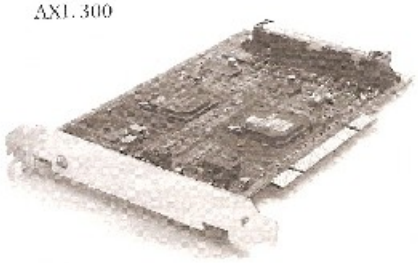


Gerenciador de chave e acelerador de criptografia nShield

Cryptoswift PCI E-Commerce Accelerator

Luna CA³

AXL 300



Aceleradores de criptografia

- A maioria dos aceleradores funcionam conjuntamente com um token. Não operam sem que um token seja inserido, com uma **senha** correta.
- O *token* requer uma chave interna ao acelerador.
- Para encriptar dados, deve-se obter a chave do *token*.

Aceleradores de criptografia

- Com um **acelerador**, envia-se o texto simples ao dispositivo, ele o **criptografa** e retorna o **texto cifrado**.