Integridade

Segurança – Engenharia de Software

Ricardo Couto A. da Rocha

Roteiro

- Integridade e Segurança Computacional
- Funções de Hashing (MD5 e SHA-1)
- Assinatura Digital
- Segurança e Ataques

Integridade e Segurança Computacional

Aspectos da integridade de mensagens

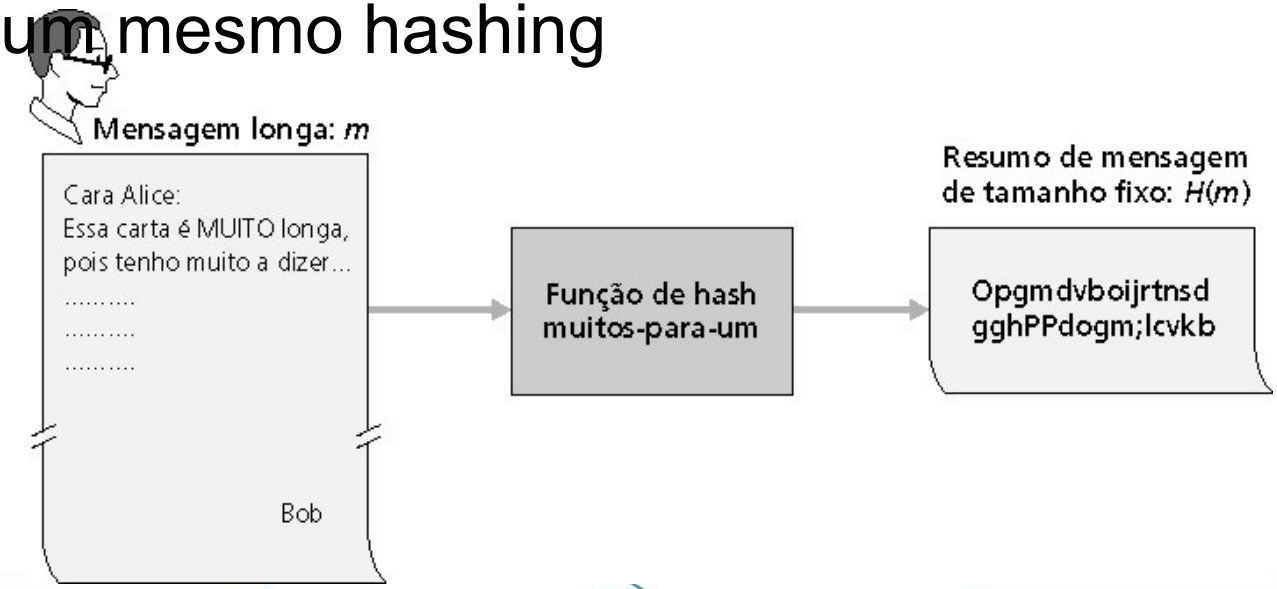
- **Validade/Integridade**: conteúdo da mensagem é correto.
- **Autenticação**: origem da mensagem é verificável e não pode ser forjada
- **Não-repúdio**: remetente da mensagem não pode negar o seu envio

Roteiro

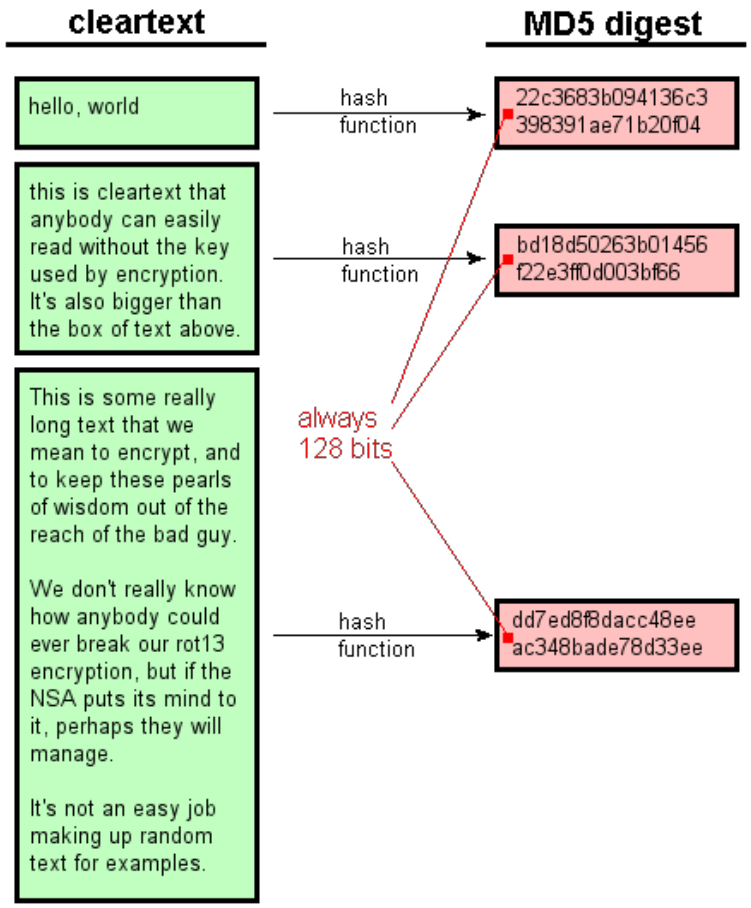
- Integridade e Segurança Computacional
- Funções de Hashing (MD5 e SHA-1)
- Assinatura Digital
- Segurança e Ataques

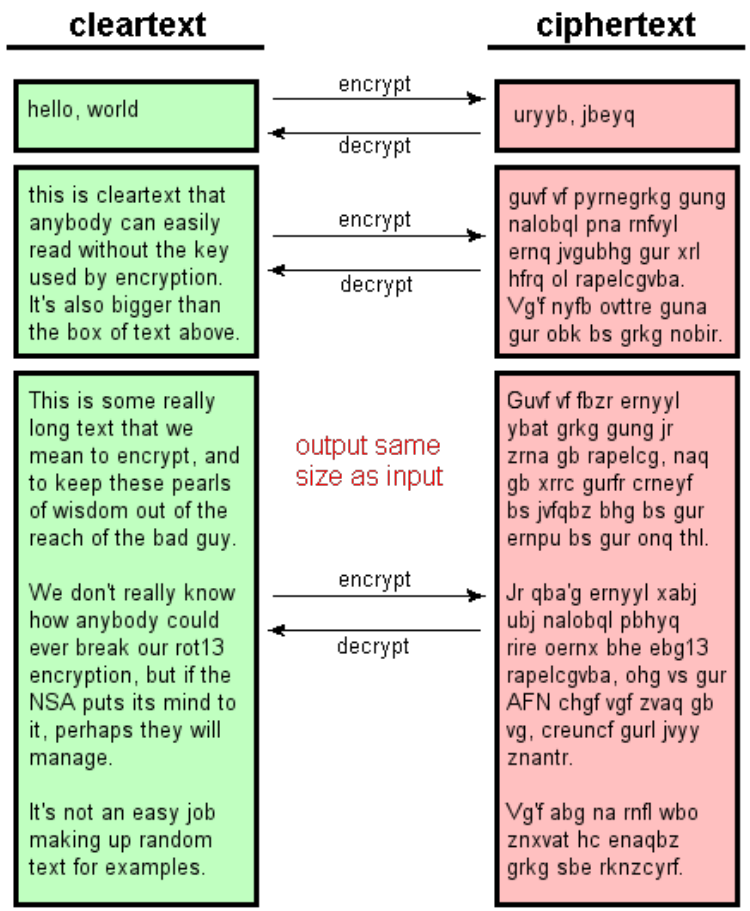
Funções Hash

- Produzem um resultado de tamanho fixo
- Relação muitos-para-um
- Computacionalmente caro obter uma mensagem com um mesmo hashing



Hashes vs. Encriptação





Steve Friedl. **An Illustrated Guide to Cryptographic Hashes**

Requisitos de uma Função Hash

Para função hash MD, também chamada de message digest

1. Dado ***P***, deve ser fácil computar ***MD(P)***
2. Dado ***MD(P)***, é *efetivamente impossível* encontrar ***P***
3. Dado ***P***, *ninguém pode* encontrar ***P'*** tal que ***MD(P')=MD(P)*** → chamado de **colisão**
4. Uma mudança em 1 bit de ***P*** chega um ***MD(P)*** muito diferente.

Exemplo

Mensagem 1

This is a very small file with a few characters

Mensagem 2

this is a very small file with a few characters

Diferença binária

- 1 bit (!): **t** (***01110100***) e **T** (***01010100***)

Hashes (MD5)

75cdbfeb70a06d42210938da88c42991 mensagem1

6fbe37f1eea0f802bd792ea885cd03e2 mensagem2

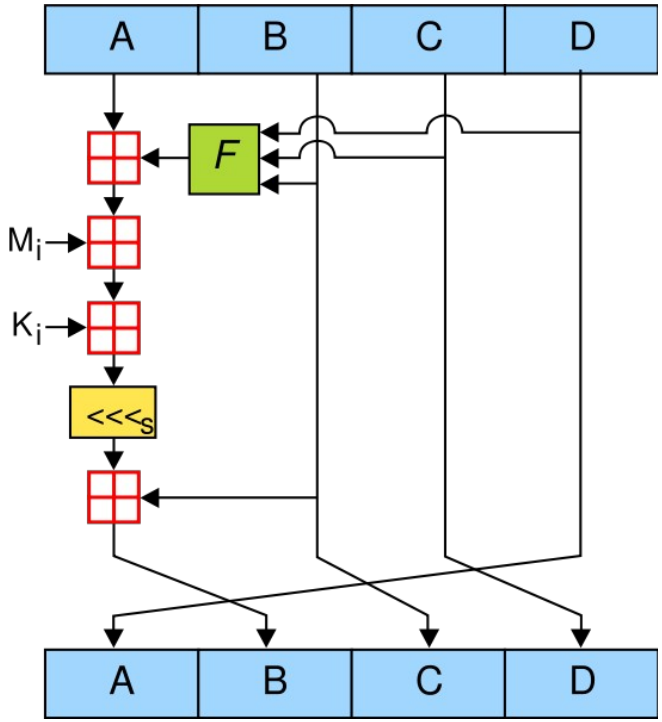
Steve Friedl. **An Illustrated Guide to Cryptographic Hashes**

MD5

MD5 proposto por Rivest (1991)

Gera um digest de 128 bits (ou 16 bytes)

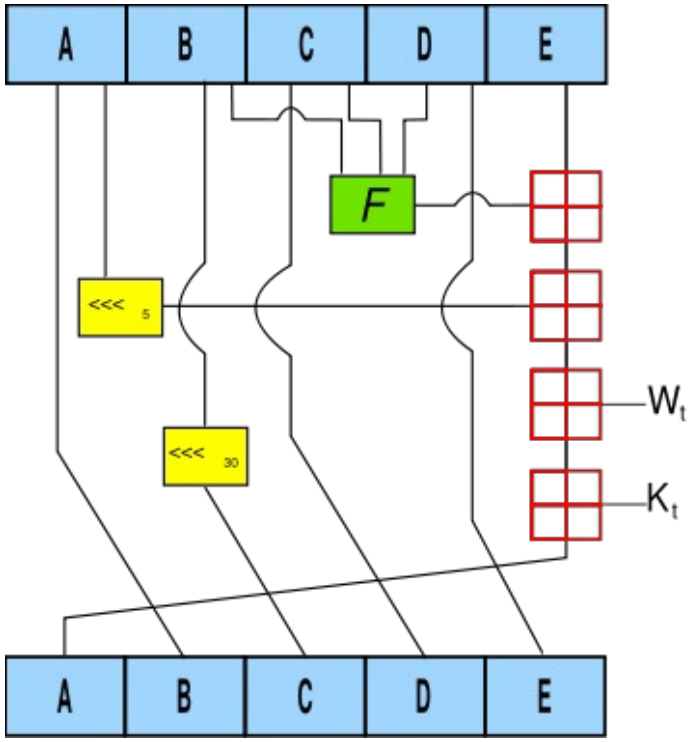
Ainda intensivamente usado **(mas não é mais considerado seguro!)**



1 rodada do MD5 para 128 bits
Repetida 4 vezes até completar o bloco de 512 bits
Rodada é repetida 16 vezes (há quatro funções F)

SHA-1

- SHA-1 proposto pelo NIST (1994)
- Gera digests de 160 bits (ou 20 bytes)
- Mais seguro que MD5 e o substituiu nas aplicações críticas

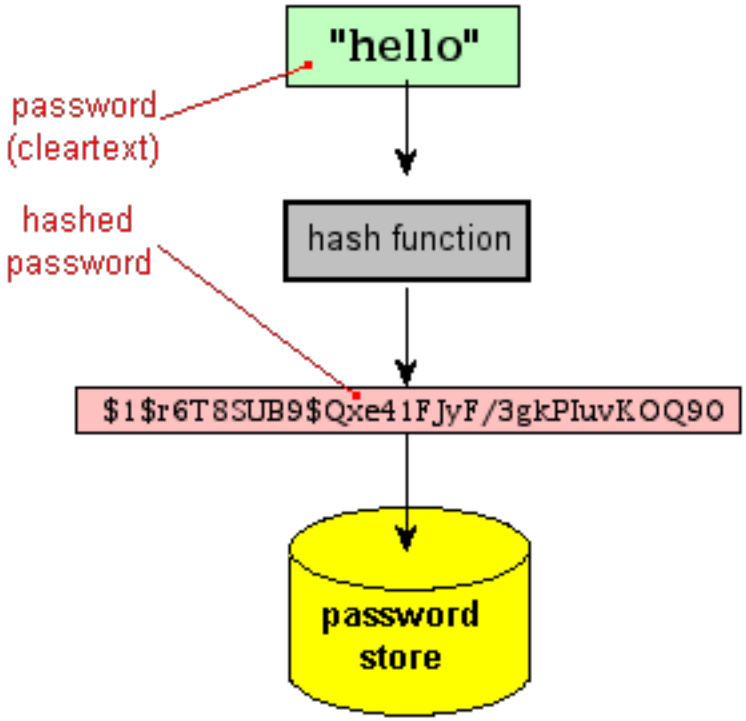


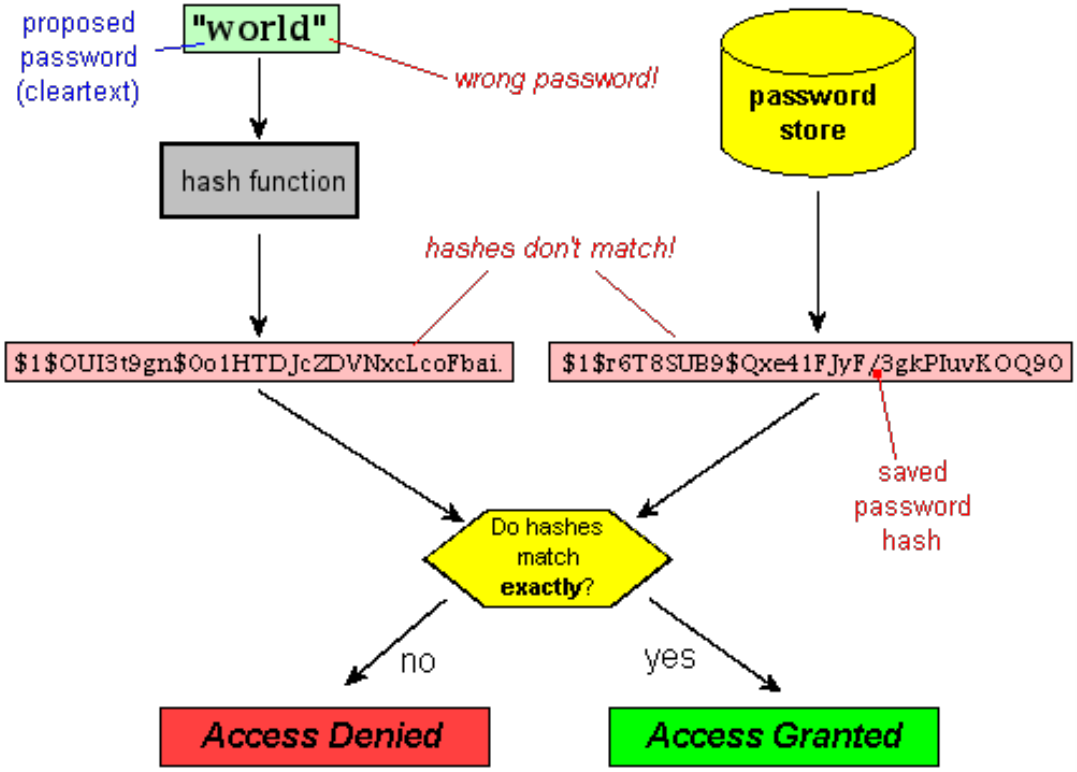
1 rodada do SHA-1 para 160 bits
Rodada é repetida 80 vezes
(há quatro funções F)

Aplicações

- Integridade de mensagens
- Verificação de senhas
- Verificação de integridade de arquivos (grandes ou críticos)
- Códigos de verificação/autenticação (como tokens de sessão, MAC)
- Parte do algoritmo de geradores de números aleatórios
- Identificação de arquivos
  - Algoritmos de índices baseados em hashing em P2P

Úteis para Armazenar Senhas





Roteiro

- Integridade e Segurança Computacional
- Funções de Hashing (MD5 e SHA-1)
- Assinatura Digital
- Segurança e Ataques

Assinatura Digital

Mecanismo que demonstra a autenticidade de uma mensagem ou documento digital.

Requisitos

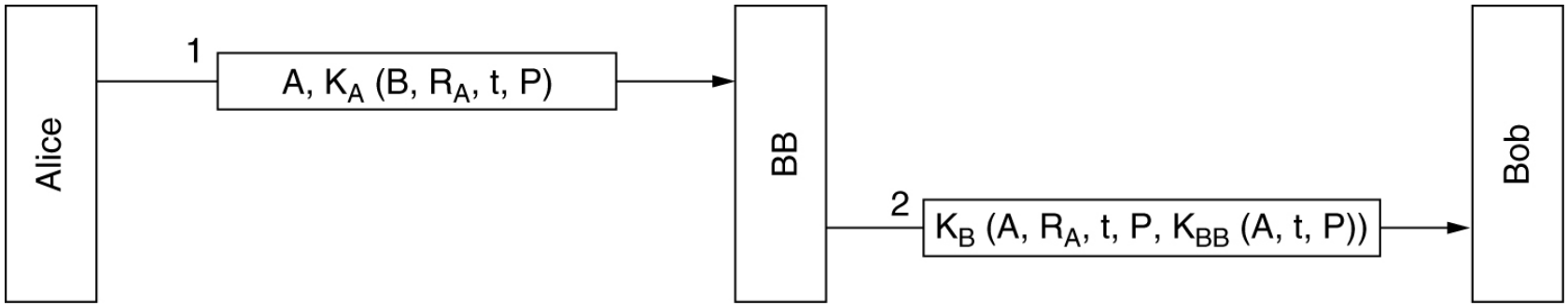
- Deve garantir a origem da mensagem
- Deve garantir o não-repúdio
- Deve garantir que a mensagem não foi modificada (integridade)
- Deve impedir ataques de replay
- Autenticidade deve poder ser verificada em qualquer instante (autenticidade não expira)

Assinatura Digital usando Chaves Secreta

Mensagem: **P**

Alice: chave $\mathbf{K_A}$

Bob: chave $\mathbf{K_B}$



Assinatura Digital usando Chaves Secretas

Problemas:

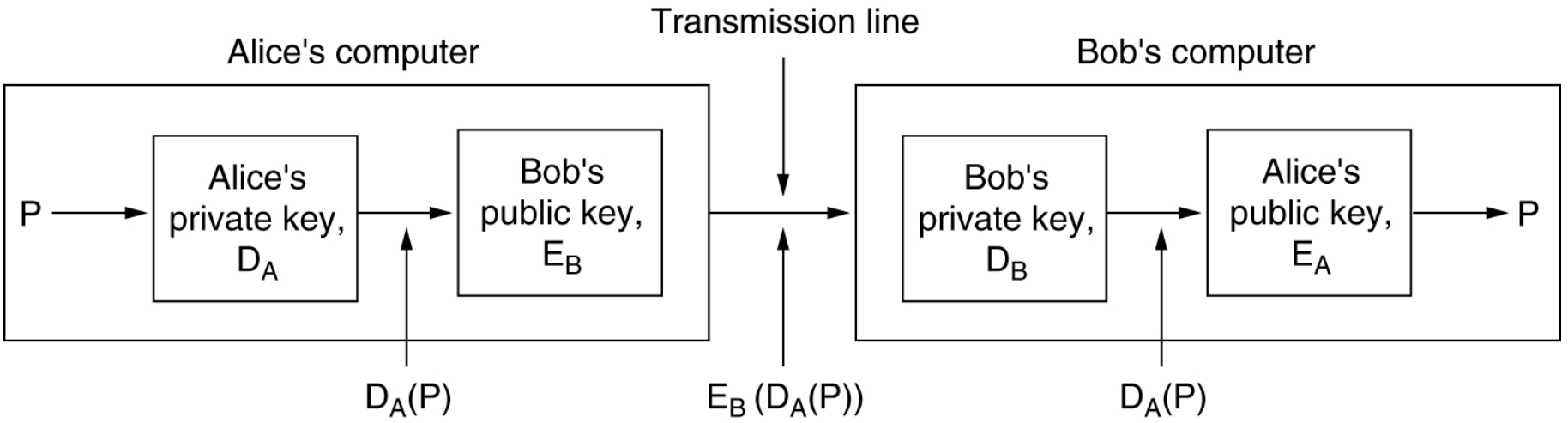
- Exige um elemento intermediário (BB) central e confiável ➔ inviável na Internet
- Chaves privadas deve ser fixas o que gera o problema de revogação de chaves (deve ser feito por BB)
- BB lê todas as mensagens

Assinatura Digital usando Chaves Pública

Mensagem: **P**

Alice: pública E_A, privada D_A

Bob: pública E_B, privada D_B

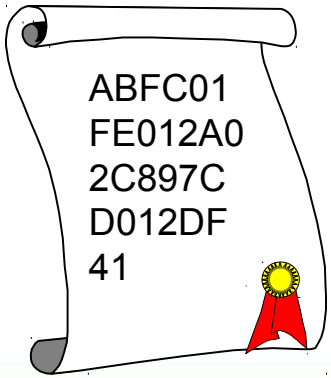


Assinatura Digital usando Chaves Públicas

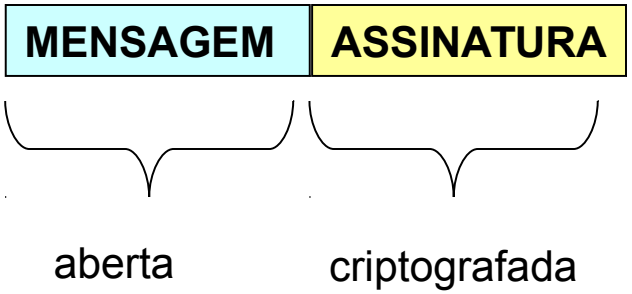
Problemas comuns

- Chaves públicas devem ser distribuídas com segurança (requisito do mecanismo)
- Mecanismo custoso (dupla criptografia) e não se pode adotar a criptografia simétrica
- Toda a mensagem é criptografada
  - Exige que **toda** a mensagem criptografada seja armazenada (**Por que?**)
  - Validação pode ser custosa

Assinatura Digital usando Hashing



Mensagem com Assinatura Digital



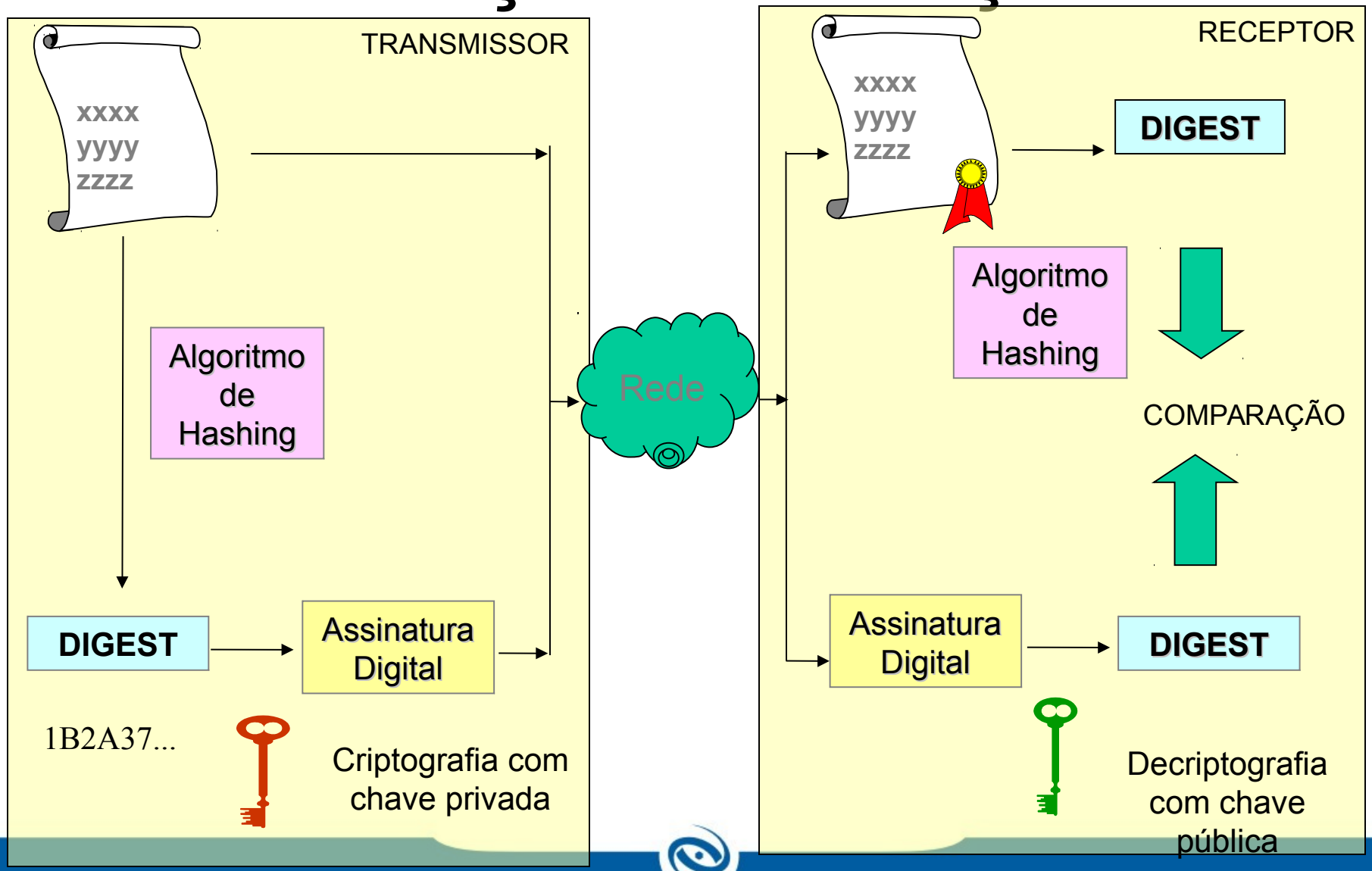
Assinatura Digital com Hashing e Criptografia Assimétrica

- Uso do par de chaves na troca da assinatura
- Receptor verifica integridade e autenticidade da mensagem
- Garante o não-repúdio e verificação por terceiros

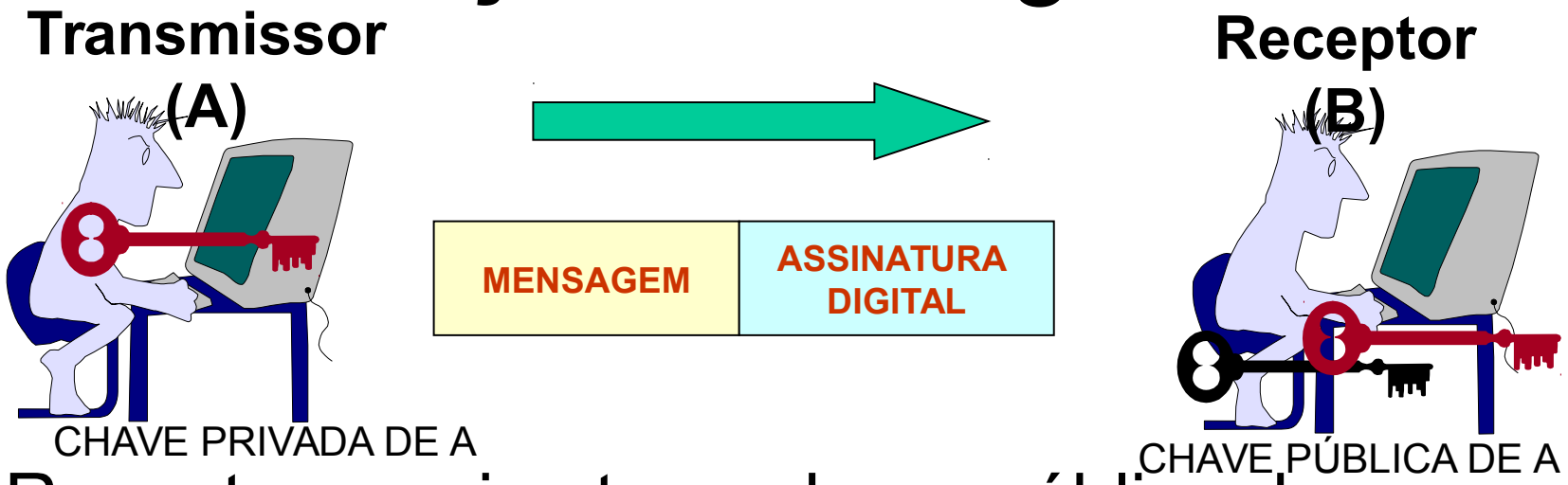
Assinatura Digital com Hashing e Criptografia Assimétrica

- Utiliza um par de chaves para implementar assinatura (privada/pública)
- Permite verificar a integridade da informação
- Uso das chaves diferente da criptografia assimétrica
  - Encriptação com chave privada e decriptação com chave pública
  - Não garante a confidencialidade!

Geração e Validação



Verificação da Integridade



- Receptor precisa ter a chave pública do transmissor
- Como distribuir e validar chaves públicas?
  - Certificados digitais e infraestrutura chave pública

Características

- Texto do documento assinado pode ser aberto ➔ usabilidade do mecanismo
- Assinatura deve conter propriedades do documento, e não apenas o seu conteúdo
  - Timestamps

Autenticação com Chave Secreta Compartilhada

HMAC: *keyed-Hash Message Authentication Code*

- Mecanismo de autenticação de mensagens que permite verificar simultaneamente a integridade de uma mensagem e a sua autenticidade
- Utiliza algoritmos de hashing: HMAC-MD5 e HMAC-SHA-1
- Utilizado em diversos protocolos como IPSec e TLS.
- Padronizado no RFC 2104

Autenticação com Chave Secreta Compartilhada

HMAC é um hashing calculado da seguinte forma*

`HMAC = Hash(K + Hash(K + dado))`

- **K** é a chave secreta
- **dado** é o dado sendo transmitido entre os peers junto com o HMAC
- **+** significa concatenar as informações

* *desconsiderando algumas constantes usadas no hashing*

Roteiro

- Integridade e Segurança Computacional
- Funções de Hashing (MD5 e SHA-1)
- Assinatura Digital
- Segurança e Ataques

Por que usar Hashing e não só Encriptação

- Velocidade
- Restrições impostas por lei
- Aplicação
  - Disseminação de mensagens
  - Usabilidade
- Operação do sistema
  - Em peer sobrecarregado, encriptação pode ser crítica
- Checksum de código
  - Ex: Autenticode da Microsoft

Segurança de Assinaturas Digitais com Hashing

- Desconsiderando aspectos do próprio mecanismo de criptografia
  - Segurança da assinatura depende da segurança do algoritmo de hashing
- Segurança do hashing
  - Propriedade *oneway*
  - Resistência a colisões → é difícil encontrar outro documento que produza o mesmo hashing

Segurança de Assinaturas Digitais com Hashing

Custo teórico de geração colisões de hashings (por força bruta)

- MD5: $2^{128/2}$ Efetivo MD5: 2^{21} (pouco seguro!)
- SHA-1: $2^{160/2}$ Efetivo SHA-1: 2^{51} (ficando insuficiente)

Exemplo de aplicação onde o ataque do aniversário pode ser usado

Ataques a Funções de Hashing

Aplicação 1: Armazenamento de senhas

- Objetivo do Ataque: encontrar qualquer colisão de um hashing específico

Aplicação 2: Integridade de mensagens

- Objetivo do Ataque: construir um conteúdo ***em particular*** que produza uma colisão com um conteúdo conhecido

Ataques no Armazenamento de Senhas baseado em Hashing

- Dado um arquivo de senhas em particular, deve-se encontrar uma mensagem qualquer que produza uma colisão com os hashing indicados no arquivo
- Abordagem 1: Força Bruta
  - Utilizar um gerador de senhas e testar os hashing das senhas geradas com os armazenados no arquivo
  - Ataque muito lento → efetivo apenas com senhas previsíveis (ex: combinação de palavras de um dicionário)

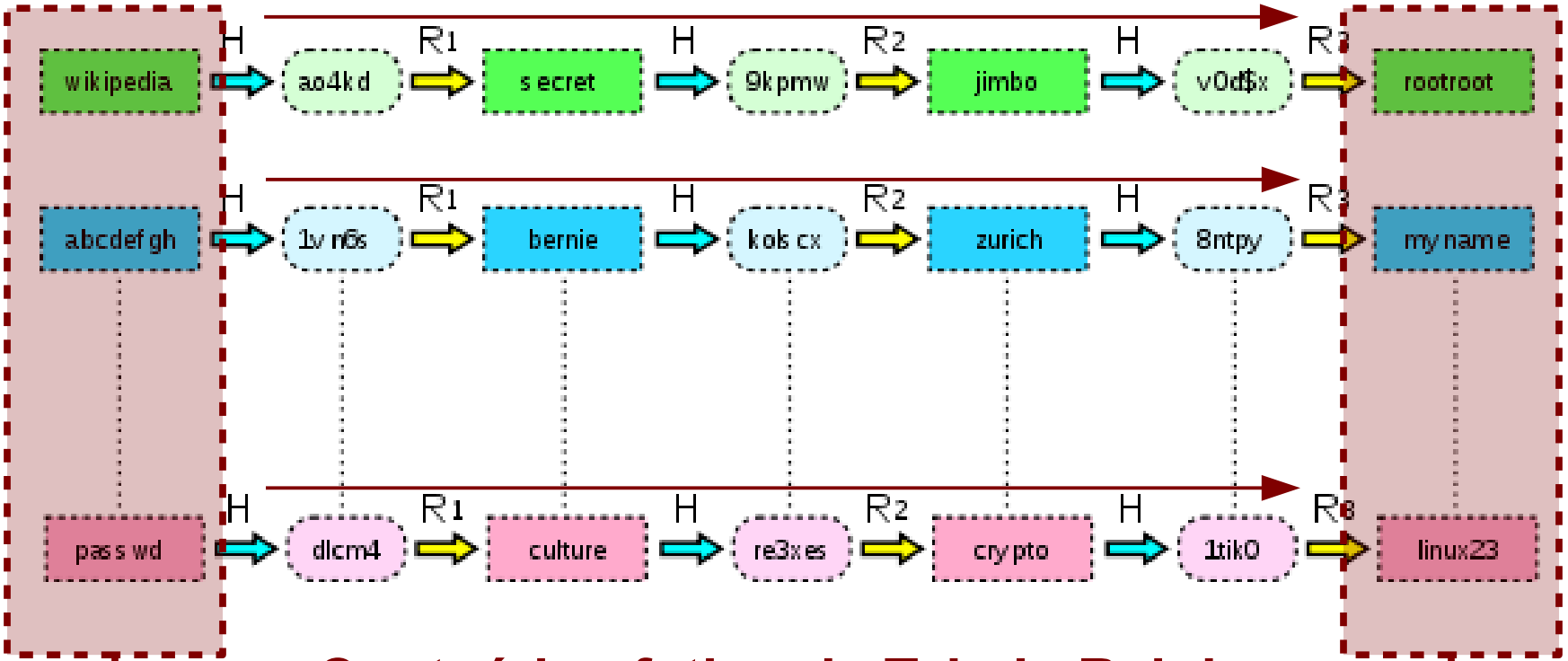
Ataques no Armazenamento de Senhas baseado em Hashing

- Abordagem 2: Uso de dicionários invertidos (hashing → palavra)
  - Problema: tamanho do dicionário
  - 7 bilhões de computadores ($7x10^9$) com um computador de 1 Tb (10^{12} bytes) permite armazenar 2^{68} hashes ($<< 2^{128}$ necessários)
  - Solução: Tabelas Rainbow

Tabelas Rainbow

- Estrutura de dados que armazena cadeias de hashes, a partir de uma função de redução pré-determinada
  - Função hash H(t): $H(t_1) = hashing_1$
  - Função redução R(t): $R(hashing_1) = t_2$
  - $R(H(t')) \neq t'$ (*seria uma contradição ao conceito de hashing se a expressão fosse uma igualdade*)
  - Exemplo: R(t) contém os 8 primeiros caracteres de H(t)
    - HMD5(“12345678”) = 25d55ad283aa400af464c76d713c07ad
    - R(“25d55ad283aa400af464c76d713c07ad”) = “dadaaafc”

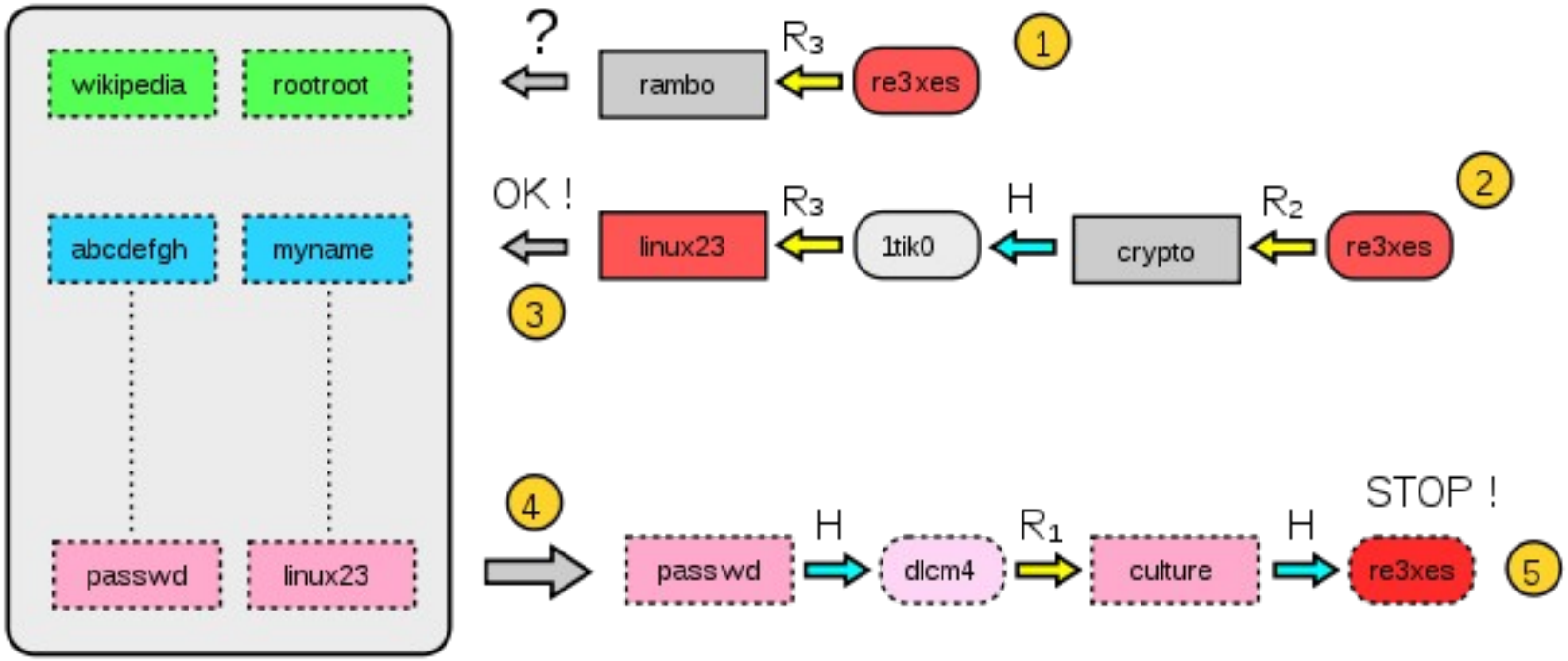
Exemplo de Tabela Rainbow



Conteúdo efetivo da Tabela Rainbow

Fonte: `http://en.wikipedia.org/wiki/Rainbow_table`

Pesquisa na Tabela Rainbow por **re3xes**



Requisitos e Defesas de Tabelas Rainbow

- Requisito para construir Tabela Rainbow
  - Método de geração de hashing precisa ser conhecido, ou a tabela não tem serventia
- Defesa de Tabelas Rainbow
  - Utilização de saltos aleatórios por senha
  - Ex: Armazenar senha *"abredecesamo"* como 4287:MD5("abredecesamo:4287")
- Defesa geral → utilizar funções de hashing lentas (exemplo: executar hashing 100 vezes)

Ataque em Integridade de Mensagens e Assinaturas

- Mais difícil pois encontrar uma colisão não é suficiente
- Necessário encontrar uma mensagem maliciosamente construída (com significado pré-determinado) e que ainda assim produza colisão
- Força bruta (demorado) ou Exploração de vulnerabilidades em funções de hashing (efetivo)

Wang, Xiaoyun, and Hongbo Yu. ***"How to break MD5 and other hash functions."*** Advances in Cryptology–EUROCRYPT 2005 (2005): 561-561