

# Introdução

Encriptação: transforma texto simples em texto cifrado usando uma chave

$$c \leftarrow E(K, p) : \text{a encriptação é aleatória (normalmente)}$$
$$p \leftarrow D(K, c) : \text{a desencriptação é determinística}$$

Cifra de César:  $\gg 3$ ,  $K=3$  GENERALIZANDO  $\gg K$

Cifra de Substituição:  $\alpha \rightarrow S$ ,  $\beta \rightarrow J$ , ...

• O espaço de chave é  $26! \approx 2^{88}$

• Não vulnerável a força-bruta, mas vulnerável a ataques de frequência de letras

Cifra de Vigenère:  $\alpha \gg x$ ,  $\beta \gg y$ ,  $\gamma \gg z$ , ...,  $K = (x, y, z)$

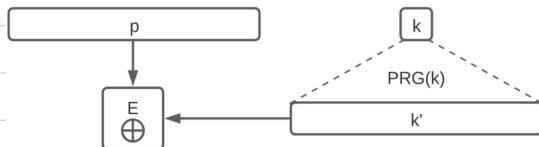
1. Escolher chave em  $\mathbb{Z}_{26}^l$ , sendo  $l$  o tamanho da chave

2. Para encriptar, fazer "shift" da letra índice- $i$  por  $K_i \bmod l$

"One-Time Pad": chave aleatória  $K \leftarrow \{0, 1\}^{(m)}$  (m) COMPRIMENTO DA MENSAGEM

$$\begin{aligned} \text{Encriptar: } c &= m \oplus K \\ \text{Desencriptar: } p &= c \oplus K \end{aligned} \quad \left. \begin{array}{c} \\ \end{array} \right\} \text{XOR}$$

• É perfeitamente seguro, desde que as chaves não sejam usadas numa vez porque  $(m_1 \oplus K) \oplus (m_2 \oplus K) = m_1 \oplus m_2$ , mas a chave tem de ser do mesmo tamanho da mensagem



**"Perfeitamente Seguro"**: desfrar nem a chave é tão difícil como adivinhar a mensagem

**Segurança**: definida em duas dimensões

**Modelo de Ataque**: poder do atacante

- poder computacional
- capacidade de colecionar outputs
- capacidade de influenciar inputs

**Objetivos de Segurança**: o que o atacante não deve poder fazer — as circunstâncias que constituem um ataque

A análise de segurança é feita com base num modelo de segurança — uma experiência corre um adversário de acordo com o seu modelo e depois verifica o objetivo de segurança

**Segurança Semântica**: incapacidade de distinguir o texto cifrado

↓ como PROVAR?

**Experiência**: execução algorítmica do modelo de segurança

1. O challenger escolhe uma chave aleatória  $K$
2. O atacante escolhe duas mensagens  $(m_0, m_1)$  tais que  $|m_0| = |m_1|$
3. O "challenger" escolhe um bit aleatório  $b$  e faz  $c \leftarrow E(K, m_b)$
4. O atacante recebe o desafio  $c$  e faz output de  $b'$
5. O atacante ganha se  $b = b'$

A encriptação está quebrada se o atacante ganha a experiência com invasamento visível desde  $1/2$

## Como provar?

→ Observe-se que, sem conhecimento da chave,  $c$  segue uma distribuição uniforme

1. Considere-se que o tamanho das mensagens é 1
2. Se a mensagem encriptada é  $t_1$ , então  $c$  é  $p \oplus 0$  ou  $p \oplus 1$
3. A chave é aleatoriamente amostrada, então os resultados não equiprováveis
4. Isto verifica-se para qualquer tamanho de mensagem

## Como quebra o "One-Time Pad"?

→ Considere-se que a mesma chave é usada mais do que uma vez

1. O adversário recebe  $c = m_1 \oplus K$
2. O adversário pede para encriptar  $m_0 \oplus m_1$ :  $c' = m_0 \oplus m_1 \oplus K$
3. O adversário pode fazer  $c \oplus c' = m_1 \oplus K \oplus m_0 \oplus m_1 \oplus K = m_1 \oplus m_0 \oplus m_1$
4. Se  $b = 0$ , então  $c \oplus c' = m_0 \oplus m_0 \oplus m_1 = m_1$
5. Verificar se  $c \oplus c' = m_1$  e retornar 0, ou, caso contrário, retornar 1
6. Vitória com probabilidade 1

## Modelos de Ataque:

- POSS**
1. "ciphertext-only": não vê o texto cifrado → desafio
  2. "Known-message": vê alguns pares de texto limpo e texto cifrado
  3. "chosen-plaintext": vê textos cifrados para textos limpos escolhidos
  4. "chosen-ciphertext": vê textos limpos para textos cifrados escolhidos

Princípio de Kerckhoffs: todos os detalhes da operação de um sistema criptográfico são públicos — o único segredo é a chave

→ Nunca usar a zona própria criptografia!

# Aleatoriedade e Segurança Criptográfica

**Criptografia Simétrica:** chave gerada de forma uniformemente aleatória ou derivada usando uma função de Derivação de Chave.

**Criptografia Assimétrica:** par de chaves gerado a partir de um algoritmo de geração de chaves pelo detentor da chave privada, que publica a chave pública.

As chaves são normalmente o material mais suscetível de um sistema seguro  
COMO  $\downarrow$  ARMazenar?  
manterem confidencial ou "embrulhar" as chaves

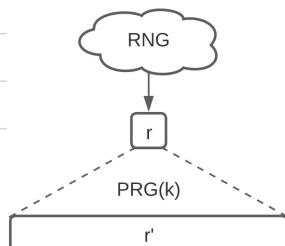
**Aleatoriedade:** propriedade do processo de geração/amostragem

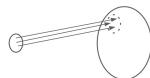
**Distribuição Uniforme:**  $\forall s^* \in S, \Pr[s = s^* : s \leftarrow \$_U] = 1/|S|$   
 $\uparrow$  MAXIMIZADA POR

**Entropia:** mede a incerteza em relação a um resultado da amostragem  
 $H(X) = \sum_{s^* \in S} -\Pr[s^*] \times \log_b (\Pr[s^*])$  = número de bits de incerteza

**Geradores Aleatórios de Números:**

1. Uma fonte de entropia - um processo natural que amosta  $\lambda$  bits a partir de uma distribuição de alta entropia
2.  $\lambda > 2$ , sendo  $\lambda$  a entropia armada
3. Extatores aleatórios (funções de hash) combinam esses bits para 2 bits
4. Os bits resultantes são armazenados como uniformes





Geradores Pseudo-Aleatórios: funções PRG :  $\{0,1\}^\lambda \rightarrow \{0,1\}^l$ ,  $l \gg \lambda$

Um PRG é um algoritmo determinista que recebe uma sequência de bits uniforme de tamanho  $\lambda$  para gerar uma string de bits longa e aparentemente aleatória, de tamanho  $l \gg \lambda$

Segurança: um atacante deve ser incapaz de distinguir outputs de geradores pseudo-aleatórios e strings verdadeiramente aleatórias

Experiência: considere-se um esquema de cifra e um atacante que não sabe  $K$

1. O atacante escolhe inputs não repetidos  $X_i$  e obtém:
  - a.  $Y_i$  escolhido de forma uniformemente aleatória, se  $b = 1$
  - b.  $Y_i = E(K, X_i)$ , se  $b = 0$
2. O atacante advinha  $b'$  e ganha se  $b = b'$

Vantagem:  $E = \left| P_r[b' = 1 | b = 1] - P_r[b' = 1 | b = 0] \right| > 0$

Segurança ( $t, \epsilon$ ): para um modelo de ataque bem definido, qualquer atacante deve correr, no máximo,  $t$  passos e ter, no máximo,  $\epsilon$  probabilidade de sucesso (vantagem)

$t$  é um limite inferior do trabalho necessário para quebrar o esquema

Para um espaço de chaves de  $2^K$ : a segurança é  $(t, t/2^K)$

Segurança de  $n$  bits: o melhor ataque para quebrar o esquema requer  $2^n$  passos

chaves de  $n$  bits não podem oferecer mais do que segurança de  $n$  bits

chaves de  $l$  bits podem levar a segurança de  $n$  bits,  $n \ll l$ , se o melhor ataque for mais eficiente do que força-bruta

CURTO

$2^{80}$

IDEAL

$$2^{128} \rightarrow (t, \varepsilon) = (2^{88}, 2^{-40})$$

LONGO

$2^{256}$

Nos SDs modernos, a geração de aleatórios é "stateful" — os PRG mantêm um estado e o SD mistura o output da fonte de entropia com ele

Resistência a Backtracking: se um adversário corromper o estado do PRG, aleatoriedade passada não deve ser comprometida — "forward secrecy"

Resistência a Predição: se um adversário corromper o estado do PRG, o SO adiciona uma camada extra (oculta) de entropia ao estado e output futuro deve voltar a parecer aleatório novamente

Objetivo de Segurança Criptográfica: "dados  $n$  bits de input a um PRG, um adversário consegue adicionar o bit  $n+1$ ?"

Segurança Heurística: analistas criptográficos tentam constantemente quebrar esquemas e refutar a segurança de  $n$  bits  
Ex: AES

Segurança Demonstrável: prova matemática — quebrar um esquema implica resolver um problema matemático difícil ou quebrar outro esquema

Assunção: problema matemático  $P$  não pode ser eficientemente resolvido

Objetivo: quebrar o esquema  $C$  não pode ser eficientemente feito

Método: construir uma redução

1. Tome-se qualquer atacante (hipotético)  $A$  que quebra  $C$
2. construir uma redução (concreta)  $B^A$  —  $B$  usa  $A$  como subrotina
3. mostrar que  $B$  resolve  $P$  quando  $A$  existe

$C$  não é seguro por si próprio, mas nem tanto quanto como a dificuldade de  $P$

# Cifras por Blocos

Uma cifra por blocos é definida por dois algoritmos determinísticos

$$\text{Encriptação: } E(K, p) = p \xrightarrow{K} c, \quad K \in \{0,1\}^2, \quad p \in \{0,1\}^B, \quad c \in \{0,1\}^B$$
$$\text{Desencriptação: } D(K, c) = c \xrightarrow{K} p, \quad K \in \{0,1\}^2, \quad c \in \{0,1\}^B, \quad p \in \{0,1\}^B$$

Uma cifra por blocos é invertível —  $K$  define uma permutação

Uma cifra por blocos deve ser uma permutação pseudo-aleatória (PRP)

Experiência: amostra uniformemente aleatória,  $K \in \{0,1\}^2$ ,  $\pi: \{0,1\}^B \Rightarrow \{0,1\}^B$

1. O atacante pode pedir encriptações

a. A experiência retorna  $E(K, p)$ , se  $b=0$

b. A experiência retorna  $\pi(p)$ , se  $b=1$

2. O faz output de  $b'$  e ganha se  $b=b'$

Vantagem:  $|\Pr[b=b'] - 1/2|$

O esquema deve ser indistinguível de uma permutação aleatória

$$\pi: \{0,1\}^B \Rightarrow \{0,1\}^B$$

Permutação Aleatória: Tabela com  $2^B$  entradas, indexada pelo texto limpo  $p$ , em que cada entrada contém um texto cifrado correspondente  $C$  amostrado de forma uniformemente aleatória, sem repetições — é invertível

$B$  é o tamanho do bloco —  $B \approx \lambda$

$$\rightarrow \text{DES: } B = 64$$

$$\rightarrow \text{AES: } B = 128$$

Ronda: algoritmo simples e eficiente, não tão seguro como a cifra por blocos

A cifra por blocos itera o algoritmo de ronda n vezes

Cada ronda toma uma chave diferente, derivada da chave da cifra por blocos

Horário de Chave: sequência de chaves de ronda

$$c \leftarrow E(k, p) = R_3(k_3, R_2(k_2, R_1(k_1, p)))$$

$$p \leftarrow D(k, c) = R_1^{-1}(k_1, R_2^{-1}(k_2, R_3^{-1}(k_3, c)))$$

Redes de Substituição-Permutação: função de ronda - tem de ser invertível

Substituição: caixas S são pequenas tabelas de pesquisa (4-8 bits) projetadas para introduzir não-linearidade na função de ronda - confusão

Permutação: transformações ao nível dos bits ou funções algébricas que introduzem dependências por todo o bloco - difusão

Redes de Feistel: a função de ronda processa metade do bloco

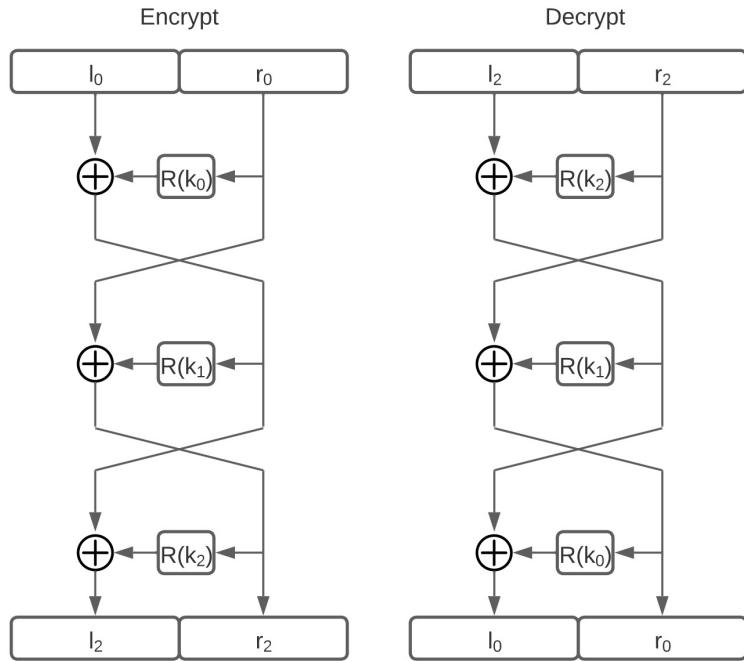
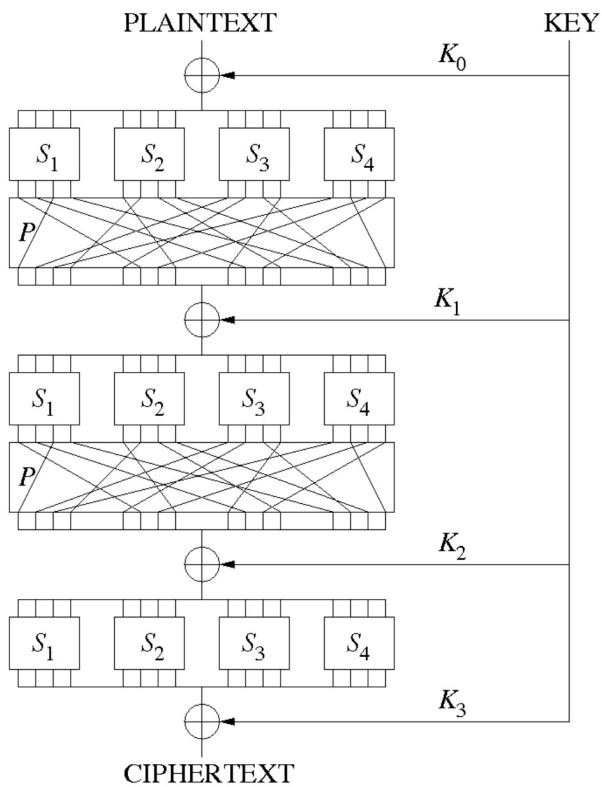
Input:  $(l, r)$

Output:  $(r \oplus R(k_i, l), l)$  função de ronda

A desencriptação é idêntica à encriptação, mas com o horário de chave invertido

A função de ronda pode ser uma função previsível-aleatória (PRF), que não é necessariamente invertível e cujo tamanho do input pode ser diferente do tamanho do output

Experiência:  $\approx$  PRP, mas escolher função  $f$  em vez de permutação  $\pi$



# AES

DES: chaves de 56 bits - segurança de 56 bits

↓  
3DES:  $E(K_1, D(K_2, E(K_3, p)))$  - segurança de 112 bits

↓  
AES: Advanced Encryption Scheme

Porque não 2DES? Meet-in-the-Middle - segurança de 57 bits

Porque não EEE? retro-compatibilidade -  $E_K D_K E_K = E_K$

→  $B = 128$  - estado interno com array  $4 \times 4$  de 16 bits

→  $\lambda \in \{128, 192, 256\}$

AddRoundKey:  $\oplus$  com o estado

SubBytes: substitui cada byte usando a Tabela de pesquisa (caixa S)

ShiftRows: as linhas das matrizes não "shifted" 0...3 posições

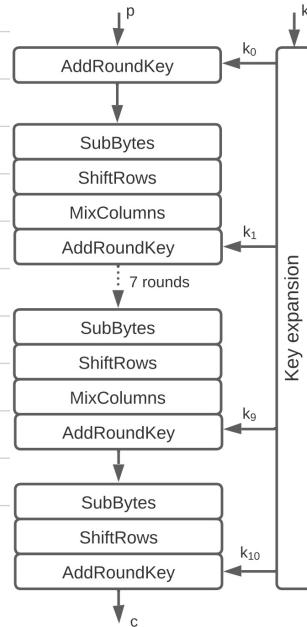
MixColumns: transforma as colunas

diffícil de implementar; canais colaterais

rápido; instruções nativas

Assume-se que o AES é uma PRP

Dá segurança demonstrável e esquemas de encriptação simétrica muito eficientes



# Encriptação Simétrica

- As cifras por blocos são uma primitiva
- Os enquadres de encriptação são construídos a partir de cifras por blocos
- A encriptação prova-se segura assumindo uma cifa de blocos como PRP

Chave: normalmente uniformemente amostrada em  $\{0,1\}^2$

Encriptação: algoritmo probabilístico  $- c \leftarrow E(k, m)$

Desencriptação: algoritmo determinístico  $- m/l \leftarrow D(k, c)$

## Segurança Semântica:

- Experiência amostra  $k$  e  $b$  de forma uniformemente aleatória
- Atacante pode pedir encriptações de mensagens escolhidas
- Atacante faz output de  $(m_0, m_1)$ ,  $|m_0| = |m_1|$
- Atacante obtem  $c \leftarrow E(k, m_b)$
- Atacante faz output de  $b'$  e ganha se  $b = b'$

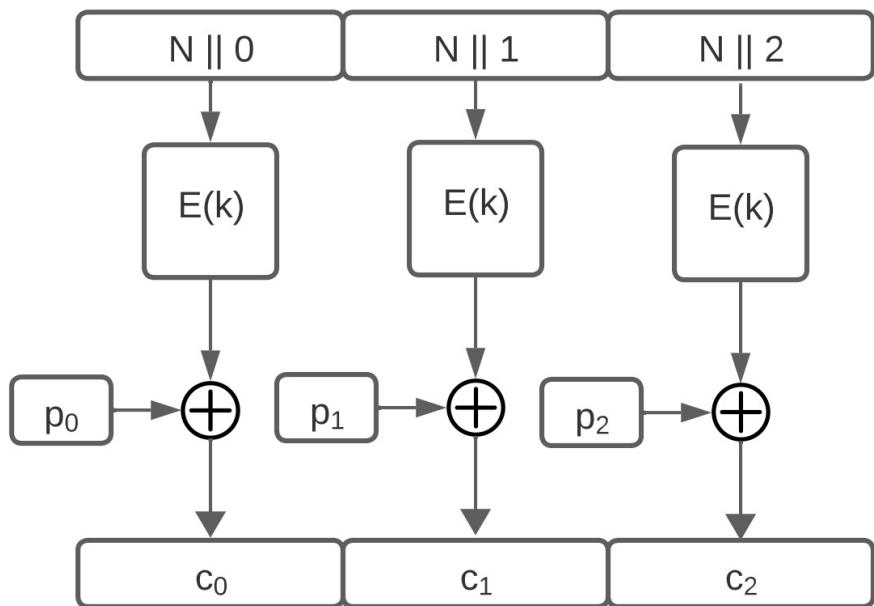
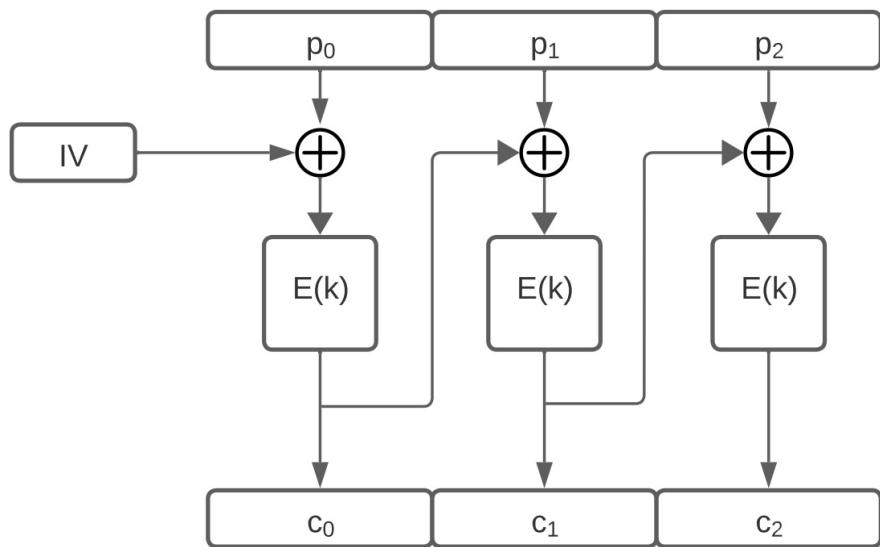
Vantagem:  $|\Pr[b = b'] - 1/2|$

Electronic-Code-Book (ECB): encriptar independentemente cada bloco  $- c_i \leftarrow E(k, p_i)$   
INSEGUR → pedir  $m_0$  e obter  $c^*$ ; enviar  $(m_0, m_1)$  e obter  $c$ ;  $b' = 0$  se e só se  $c = c^*$

Cipher Block Chaining (CBC): vetor de inicialização e dependência entre blocos  
SEGUR → duas encriptações do mesmo texto parecem independentes

Padding: adicionado sempre - adicionar  $K - |M|$  bytes com valor  $K - |M|$  (múltiplo de B)

Counter Block Mode (CTR): encriptação "stateful", pré-processável, paralelizável



# Cifras por Fluxo

**Cifras por Fluxo:** Geradores Aleatórios Determinísticos de Números (DRNG)  
 ↳ recebem uma semente conhecida e geram aleatoriedade

$$(k, n) \Rightarrow KS$$

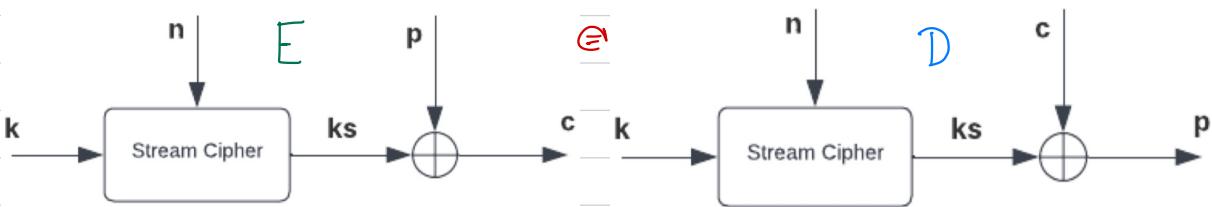
K: chave - segredo - 128/256 bits

n: "nonce" - público - 64/128 bits

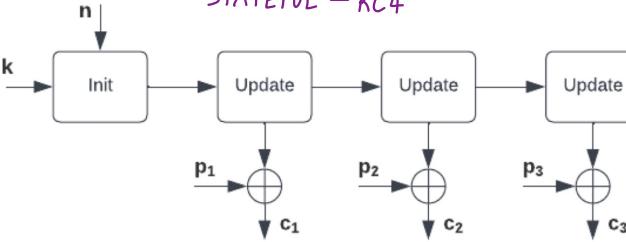
! Repetir o "nonce" revela duplicados!

KS: "Keystream"

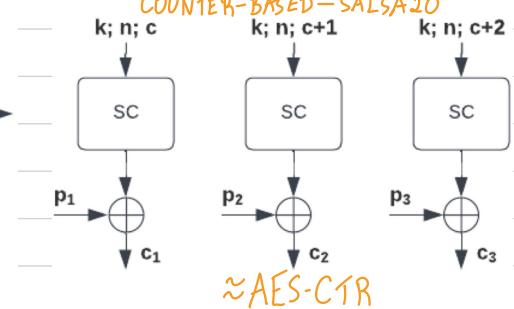
$$c \leftarrow KS \oplus m$$



STATEFUL - RC4



COUNTER-BASED-SALSA20



Cifras baseadas em Hardware: circuitos eletrônicos - lidam com bits

Cifras baseadas em Software: algoritmos - lidam com palavras - 32/64 bits

## REGISTOS FEEDBACK SHIFT

→ array de bits

+

→ função de feedback -  $f(R)$       Ex: XOR

$R_0$ : estado inicial



$R_1$ : estado seguinte

$$R_1 = R_0 \ll 1 \parallel f(R_0)$$

Output: bit mais à esquerda de  $R_0$

- Todos os FSR têm um período — maior período  $\Rightarrow$  mais seguro
- O estado  $0^n$  passa sempre para  $0^n$

Registros Feedback Shift Lineares (LFSR): a função de feedback é linear

$$\rightarrow F(R) = 1 + X + X^2 + \dots + X^n$$

↳  $X^i$  só é incluído se a função faz XOR do bit índice- $i$

- O período é máximo se o polinómio é primitivo (não redutível)

Qual é o período máximo de um LFSR de  $n$  bits?  $2^n - 1$

↳  $0 \dots \rightarrow 0 \dots$

• Usar diretamente um LFSR como cifra por fluxo é inseguro porque um atacante só precisa de  $n$  bits de output para recuperar o estado inicial e o feedback polinomial e, assim, adivinhar futuros "Keystreams"

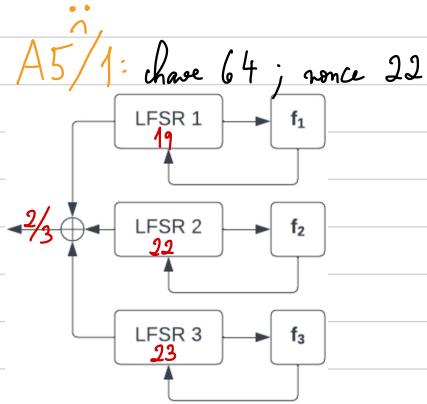
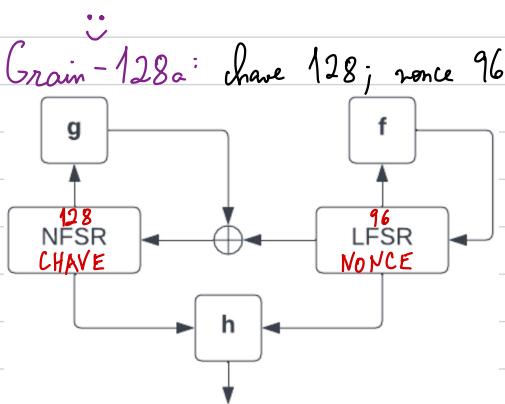
**Registros Feedback Shift Filtados:** escondem a linearidade filtrando os outputs através de uma função não-linear, mas ainda são vulneráveis a ataques subtis

**Registros Feedback Shift Não-Lineares (NFSR):** a função de feedback é não-linear

→ XOR (+); AND (x); OR

- NFSR não criptograficamente mais fortes do que LFSR
- não há forma de calcular nem garantir o período em menos de  $2^n$  passos

## CIFRAS POR FLUXO ORIENTADAS AO HARDWARE



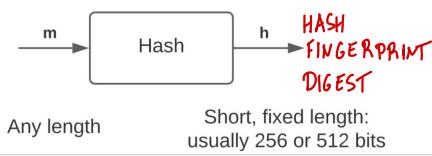
## CIFRAS POR FLUXO ORIENTADAS AO SOFTWARE

RC4: com chave de 128 bits e nem "nonce", nem XORs, nem ANDs, ...  
não hochas - falha por falta de "nonce" e desvaneamento estatístico na geração de "Keystreams".  
Ex: VEP; TLS

Salsa20: texto 512; chave 256; "nonce" 64; contadores 64; palavras 128



# Funções de Hash



Função de Hash Criptográfica:

- 1. Não Injetiva
- 2. Bem Distribuída
- 3. Imprevisível
- 4. Determinística
- 5. Pública

- Valores de hash conseguem identificar valores de input arbitrariamente grandes
- Assinar  $H(m)$  é tão seguro como assinar  $m$

S: conjunto de pré-imagens - domínio

R: conjunto de imagens - "range"

$$\rightarrow |S| \gg |R|$$

Propriedade 1 - Resistência a Pré-Imagens: é difícil encontrar o input que produz um dado valor de hash

Experiência: ao atacante é dado um valor  $y \in R$  e ele adivinha  $x \in S$   
 → ganha se  $h(x) = y$

Ataque: pesquisar por todas as pré-imagens possíveis (força-bruta) -  $+ 2^n$  op.

Propriedade 2 - Resistência a Colisões: é difícil encontrar  $m_0$  e  $m_1$  tais que  $h(m_0) = h(m_1)$

Ataque: computar valores como num ataque de força-bruta, armazená-los indexados pelo valor da imagem e procurar cada nova imagem até encontrar uma colisão -  $\sqrt{2^n}$

→ O output da hash deve ser o dobro do parâmetro de segurança, isto é, segurança de  $n$  bits → hashes de  $2n$  bits

## Como construir funções de hash?

Construção Merkle-Damgard: confia numa função de compressão de  $m+n$  para  $n$  bits para construir uma função de hash - output de comprimento  $n$   
Ex: MD4; MD5; SHA1; SHA256; SHA512

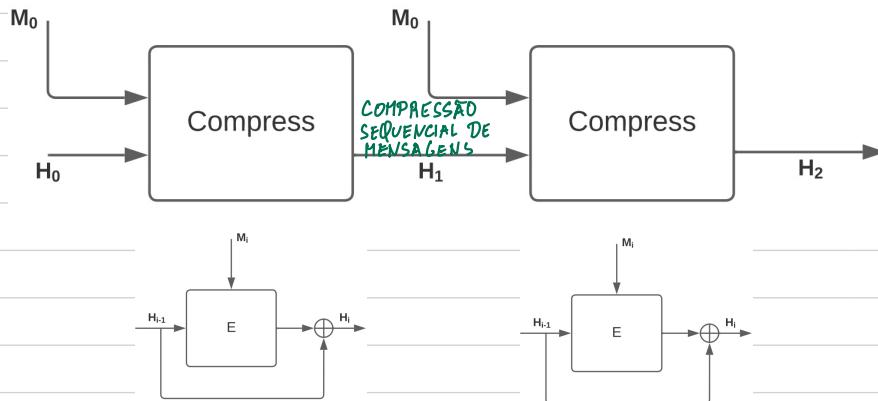
→  $H_0$  é o valor inicial: constante e público

→  $M$  é partido em blocos de tamanho  $m$

Padding: sempre acrescentado - mensagem  $10^*$  ... COMPRIMENTO FIM

O resultado da compressão é resistente a colisões, pelo que toda a construção é resistente a colisões

Função de Compressão: construção de Davis-Meyer - cifra por blocos



MD5: quebrada - output de 128 bits

SHA0: quebrada - output de 160 bits

SHA1: quebrada - output de 160 bits - 5 adições DM de 32 bits

SHA2: usável - output de X bits - SHA224 ≈ SHA256 ; SHA384 ≈ SHAS12

Construção Esponja: usa uma permutação de  $l$  bits para construir uma função de hash — output de comprimento arbitrário

Ex: SHA3

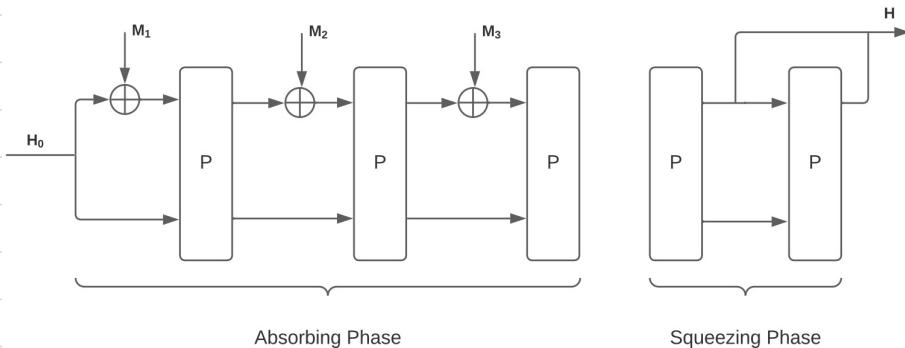
→ O estado é do mesmo tamanho  $w$  que o input da permutação

1. Absorver: permuta a mensagem

- valor inicial fixo  $h_0$ , acumula gradualmente a mensagem no estado
- a mensagem é partida em blocos de tamanho  $r$  (rácio)
- o bloco é menor do que o tamanho do estado
- é feito XOR do bloco no estado
- a permutação é recomputada

2. Espremer: busca o output

- processo dual constrói iterativamente o output
- o output é construído bloco por bloco
- a permutação é computada sobre o estado interno
- a parte do estado com o tamanho do bloco é acumulada no output



SHA3: resável — output variável — permutação de  $l = 1600$  bits — funções SHAKE if off ~

# MACs

MAC: Código de Autenticação de Mensagem - autenticação simétrica

$$t \leftarrow \text{MAC}(k, m)$$

- $t$  garante que  $m$  foi produzido por alguém que conhecia  $k$
- implica que a mensagem  $m$  não foi alterada desde a sua criação

1. Ambas as partes querem autenticação e integridade das mensagens
2. Alguma forma de configurar/concordar em estabelecer a chave  $K$
3. O emissor computa  $t \leftarrow \text{MAC}(k, m)$  e envia  $(m, t)$
4. O receptor obtém  $(m, t)$  e recomputa  $t' \leftarrow \text{MAC}(k, m)$
5. Se  $t \neq t'$ , então a mensagem é rejeitada

- A aceitação significa que  $m$  foi produzido com o conhecimento de  $k$
- MACs não dão confidencialidade, mas nem integridade - São ortogonais à encriptação
- Como não é possível computar  $t$  sem  $k$ , então o adversário não pode alterar a mensagem nem conjugar novas mensagens

## Segurança UF-CMA

Objetivo: infogabilidade

Poder do Adversário: ataques de escolha da mensagem

Experiência:

1. Gera uma chave  $K$
2. O adversário (adaptativamente) envia  $m$  para obter  $t \leftarrow \text{MAC}(m, k)$
3. Eventualmente, o atacante faz output de  $(m^*, t^*)$

Vitória: se  $t^* = \text{MAC}(k, m^*)$  e  $t^*$  não foi produzido pela experiência

Problema: O MAC por si só não protege contra ataques de repetição

Solução: Impor que as mensagens nunca se repitam na rede - números de sequência

Como construir MACs? A partir de funções de hash e chaves por blocos

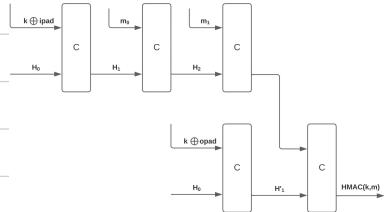
Construção + Simples: prefixar a chave -  $\text{MAC}(K, m) = H(K \parallel M)$

PROBLEMA

MD origina um MAC inseguro porque, dado  $(m, t)$ , o atacante faz output de  $H(K \parallel M \parallel \text{pad} \parallel M')$  que pode ser computado só a partir de  $t'$  e  $m'$

ATAQUE DE EXTENSÃO DE COMPRIMENTO

Quando instanciado com a construção MD, a função de compressão é uma PRF e  $\text{HMAC} = H((K \oplus \text{opad}) \parallel (K \oplus \text{ipad}) \parallel m)$



Colisões em MAC levam a forjas! -  $\sqrt{2^n}$  MACs

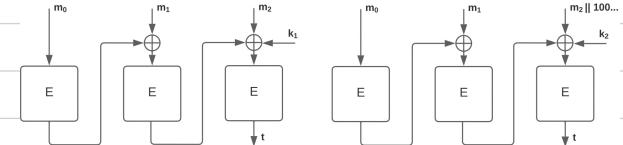
CMAC: modo de operação CBC com IV a zero

a etiqueta é o último bloco de texto cifrado

$$\ell = E(K, 0)$$

$$k_1 = (\ell \ll 1) \oplus (0x00..0087 * \text{LSB}(\ell))$$

$$k_2 = (k_1 \ll 1) \oplus (0x00..0087 * \text{LSB}(k_1))$$



Função de Hash Universal:  $\Pr[UH(K, m_0) = UH(K, m_1)] \leq \epsilon$  - não tem de resistir a colisões

é possível usar como MAC desde que só se autentique uma mensagem!

usar numa PRF para fortalecer a UHF - encriptar o valor da hash universal

$$\rightarrow UH(K_1, m) \oplus \text{PRF}(K_2, \text{nonce})$$

Ex: Wegman-Carter - Poly 1305  $((K_1, K_2), m) = (m_1 K + \dots + m_n K^n \pmod p) + \text{AES}(K_2, \text{nonce})$

# Encriptação Autenticada

Porque?

1. As mensagens têm de ser confidenciais - encriptação
2. As mensagens têm de ser autênticas - MACs
3. As mensagens não devem ser repetidas/omitidas/removidas - AEAD

Encriptação:  $(c, t) \leftarrow \$ AE_{\text{enc}}(k, [n, a]_m)$

Desencriptação:  $m / \perp \leftarrow A\text{Dec}(k, [n, a]_c, t)$

Correção:  $\forall m, m = A\text{Dec}(k, AE_{\text{enc}}(k, [n, a]_m))$

Segurança: o adversário não consegue distinguir duas encriptações de qualquer mensagem nem forjar mensagens novas

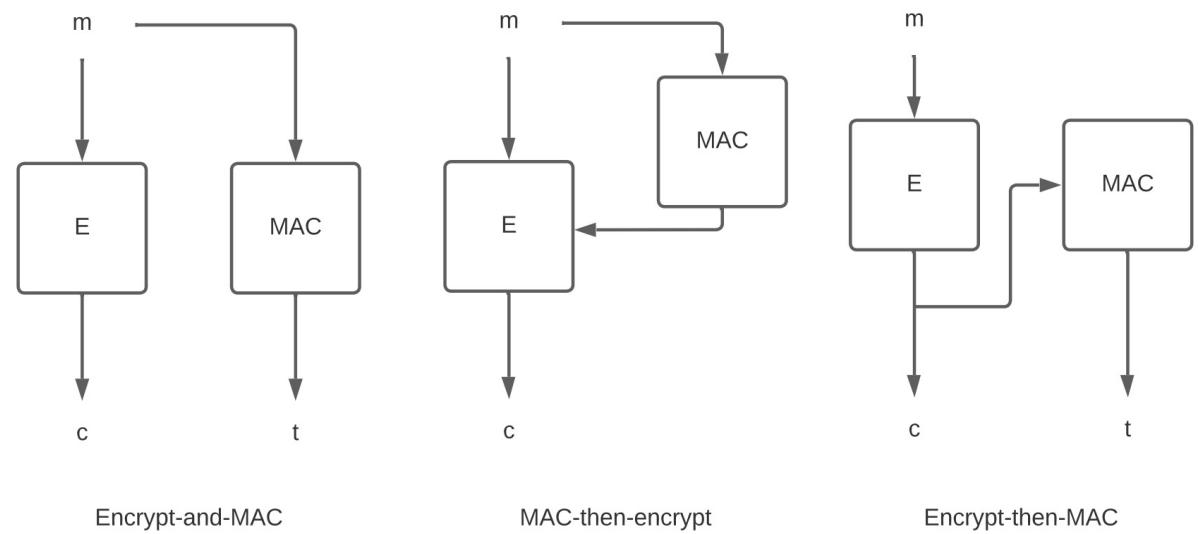
Experiência: IND-CCA

1. A experiência amosta  $k$  e o bit  $b$  de forma uniformemente aleatória
2. O atacante pode pedir encriptações de mensagens escolhidas
3. O atacante pode pedir desencriptações de textos cifrados escolhidos
4. O atacante faz output de  $(m_0, m_1)$  tal que  $|m_0| = |m_1|$
5. O atacante obtém  $c \leftarrow \$ E(k, m_b)$
6. O atacante pode pedir mais encriptações/desencriptações, mas não de  $c$
7. O atacante faz output de  $b'$  e ganha se  $b' = b$

AEAD: O atacante pode escolher os "nonces", nem os repete

Vantagem:  $|\Pr[b = b'] - \frac{1}{2}|$

Infraçabilidade: Sem saber  $k$ , é impossível encontrar um novo  $(a, c, t)$  para o qual  $m \leftarrow A\text{Dec}(k, a, c, t)$ ,  $m \neq \perp$



Encrypt-and-MAC: encriptação e MAC da mensagem  
→ AE com  $K = (K_1, K_2)$  feita com processamento paralelo

$$1. c \leftarrow \$E(K_1, m) \quad 2. t \leftarrow \text{MAC}(K_2, m) \quad 3. \text{Output}: (c, t)$$

- ✗ MAC exposto
- ✗  $c$  potencialmente malicioso desencriptado antes da autenticação
- ✗ MACs não projetados para garantir confidencialidade
- ✗  $t \leftarrow \text{MAC}(K_2, m \parallel n)$  é seguro

MAC-then-Encrypt: encriptar a mensagem e a sua autenticação  
→ AE com  $K = (K_1, K_2)$  feita sequencialmente

$$1. t \leftarrow \text{MAC}(K_1, m) \quad 2. c \leftarrow \$E(K_2, m \parallel t) \quad 3. \text{Output}: c$$

- ✗  $c$  potencialmente malicioso desencriptado antes da autenticação
- ✗ "padding oracle attack"
  - ↳ a desencriptação falhou por causa do padding ou do MAC?

Encrypt-then-MAC: autenticar a encriptação da mensagem  
→ AE com  $K = (K_1, K_2)$  feita sequencialmente

$$1. c \leftarrow \$E(K_1, m) \quad 2. t \leftarrow \text{MAC}(K_2, c) \quad 3. \text{Output}: c$$

- ✗ o texto cifrado não é desencriptado a não ser que esteja autenticado
- ✗ útil contra ataques DoS — a verificação MAC é muito rápida

## CONSTRUÇÕES OTIMIZADAS

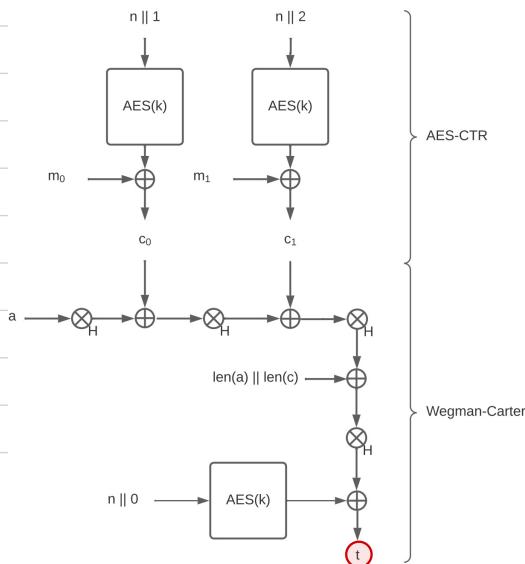
• AEADs modernos não são composições "black-box" de encriptação/MAC, mas sim camadas de encriptação/autenticação visíveis em todas as construções

Desempenho: encriptação/desencriptação de blocos em paralelo - "throughput", "streamability", requisitos de memória, ...

Modo Galois-Counter (GCM): AES-CTR + Wegman-Carter = Encrypt-then-MAC  
 → chaves de 128 bits → "nonce" de 96 bits →  $CTR \geq 1$

GHASH: função de hash universal  $\boxed{hk} \leftarrow AES(k, 0)$   
 $P(x) = (pad(A), pad(c), |A||C|)$ , em  $hk$

- a camada de encriptação linda o paralelismo do modo CTR
- a camada de autenticação bloqueia se "a" for conhecido no fim
- se "a" for conhecido desde o início, GCM é streamable - a autenticação do bloco anterior é acumulada enquanto o bloco atual está a ser encriptado!



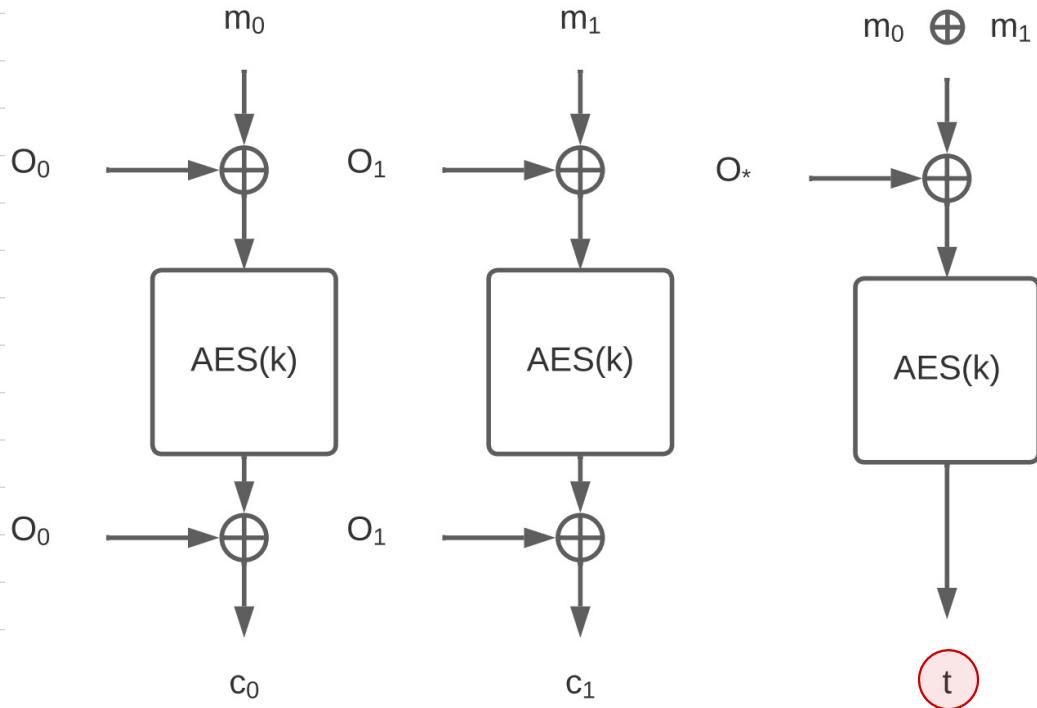
Offset Code Book (OCB): o offset depende da chave e da "nonce", é incrementado para cada bloco e último é computado a partir do último bloco de texto processado

$$\rightarrow T = E(k, S \oplus O_*) \oplus E(k, a_0 \oplus O_0) \oplus E(k, a_1 \oplus O_1) \oplus \dots, \quad S = m_0 \oplus m_1 \oplus \dots$$

Reutilização do "Nonce": o atacante pode identificar blocos duplicados e quebrar a autenticidade do OCB, combinando blocos de mensagens autenticadas com OCB para criar outra mensagem autenticada, mas não consegue extrair a chave (ao contrário de GCM)

Eficiência: OCB é 3 vezes mais rápido do que GCM, apesar de ambos fazermos o mesmo número de chamadas à cifra por blocos

→ OCB requer encriptação e desencriptação ao contrário de GCM



**Modo IV Sintético:** composição genérica de encriptação baseada em "nonce" e PRF, reforçando AES-GCM contra a reutilização do "nonce"

1.  $t \leftarrow \text{PRF}(k_1, a \parallel p \parallel n)$
2.  $c \leftarrow \text{Enc}(k_2, m=t, p)$
3. Output:  $(c, t)$

Se  $n$  se repetirem, mas  $a$  ou  $p$  mudarem, não há problema!

Se  $n$ ,  $a$  e  $p$  se repetirem, há problema!

A "tag" usada como "nonce" da encriptação é muito provavelmente fresca

O esquema não é "streamable"

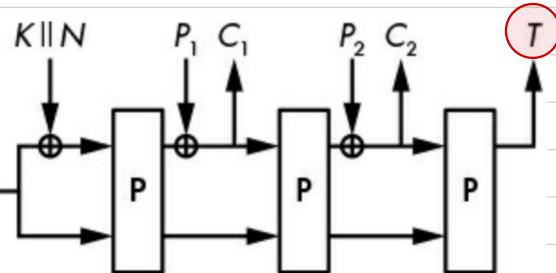
**AE baseada em Permutações:** Duplex  $\approx$  abração da esponja do SHA-3

P: permutação fixa (nem chave)

h: valor público

O último bloco tem de ser "padded"  $H_0$

Eficiência: rápido; "streamable"



Reutilização do "Nonce": resiliente — inforgeabilidade não afetada, mas mensagem comprometida desde o primeiro bloco até ao último bloco subsequente com um prefixo comum, mantendo-se confidencial após a divergência

Número mecanográfico: 

--	--	--	--	--	--	--	--

Versão A

Nome completo: \_\_\_\_\_

**Grupo 1 - Segurança Criptográfica e Aleatoriedade (35%)****1.1. Cifras Clássicas.**

Considere uma utilização da cifra Vigenére para cifrar textos escritos em Português em que o tamanho da chave é igual ao tamanho da mensagem a cifrar.

- Explique porque razão podemos dizer que esta cifra oferece uma noção de *segurança perfeita*, no caso de a chave (gerada utilizando uma distribuição uniforme) ser utilizada apenas uma vez. Indique na sua resposta o que entende por *segurança perfeita* neste contexto.

Segurança Perfeita é o criptograma ser indistinguível de uma string de bits aleatória, isto é, não ter padrões nem propriedades que permitam ao atacante descobrir algo mais do que o seu comprimento - decifrar sem a chave é tão difícil como adivinhar a mensagem.

A cifra de Vigenere oferece segurança perfeita neste contexto porque, como o tamanho da chave é igual ao tamanho da mensagem a cifrar e a chave é gerada usando uma distribuição uniforme, cada letra é *shifted* um número (gerado de forma uniformemente aleatória) de posições, não seguindo nenhum padrão e sendo, portanto, indistinguível de uma permutação aleatória.

- Descreva um ataque que permita recuperar a chave, no caso de a chave ser utilizada muitas vezes para cifrar diversos textos (do mesmo tamanho  $n$ ) em Português.

Se a mesma chave for utilizada muitas vezes para cifrar diversos textos em Português, a cifra de Vigenere é vulnerável a um ataque de análise de frequências.

Para concretizar o ataque, basta recolher muitos criptogramas e analisar a frequência de cada letra, comparando as frequências das letras nos criptogramas com as frequências das letras no universo dos textos em Português.

Sabendo que o tamanho da chave é  $n$  e tendo efetuado a análise de frequências, é possível recuperar a chave.

**1.2. Distribuições Simples.**

Seja  $p > 2$  um inteiro primo, e  $\ell > 0, \lambda \geq \ell$  inteiros. Seja  $x \leftarrow_{\$} [k]$  a operação de gerar um valor com a distribuição uniforme do conjunto de inteiros  $\{0, \dots, k-1\}$ . As seguintes distribuições são uniformes?

Distribuição  $\{x \bmod p \mid x \leftarrow_{\$} [2^\lambda]\}$

$\{x \bmod 2^\ell \mid x \leftarrow_{\$} [2^\lambda]\}$

$\{x \bmod p \mid x \leftarrow_{\$} [2^{\lambda+p}]\}$

$\{x \bmod p \mid x \leftarrow_{\$} [2^\lambda + p]\}$

$\{x \bmod 2^\ell \mid x \leftarrow_{\$} [2^\lambda + p]\}$

Sim

X

X

X

X

X

Distribuição  $\{x \bmod 2^\ell \mid x \leftarrow_{\$} [p]\}$

$\{x \bmod p \mid x \leftarrow_{\$} [p]\}$

$\{x \bmod 2^\ell \mid x \leftarrow_{\$} [2^\lambda \cdot p]\}$

$\{x \bmod 2^\ell \mid x \leftarrow_{\$} [2^{\lambda+p}]\}$

$\{x \bmod p \mid x \leftarrow_{\$} [2^\lambda \cdot p]\}$

Sim

X

X

X

X

X

SIM QUANDO É MÚLTIPLO

### 1.3. Pseudo-Aleatoriedade

1. Qual a diferença entre um gerador de números aleatórios e um gerador de números pseudo-aleatórios?

Um gerador de números aleatórios é uma fonte de entropia/incerteza, gera números de forma verdadeiramente aleatória e imprevisível.

Um gerador de números pseudo-aleatórios é um algoritmo determinista que recebe uma semente pequena e uniforme para gerar uma string de bits longa e aparentemente aleatória.

A diferença é que os primeiros são não-deterministas e os segundos são deterministas.

2. Os geradores de números pseudo-aleatórios utilizados nos computadores modernos oferecem segurança mesmo quando o estado interno do gerador é comprometido. Explique que tipo de segurança deve este tipo de construção oferecer para valores gerados no passado e para valores que venham a ser gerados no futuro.

Os geradores de números pseudo-aleatórios devem oferecer resistência a backtracking e resistência a previsões.

Resistência a backtracking é não comprometer a aleatoriedade passada (segurança para valores gerados no passado) se o estado interno do gerador for comprometido - bits anteriormente gerados são impossíveis de recuperar.

Resistência a previsões é adicionar uma camada extra (oculta) de entropia e o output futuro (segurança para valores que venham a ser gerados no futuro) voltar a parecer aleatório novamente se o estado interno do gerador for comprometido - bits futuros são impossíveis de prever.

### 1.4. Dificuldade Computacional

Suponha que pretende parametrizar um sistema criptográfico para 80-bits de segurança. Qual seria o tamanho das chaves utilizadas para (1) cifras simétricas, (2) MAC, e (3) o tamanho do output das funções de hash quando se pretende resistência a colisões? Justifique a sua resposta.

Para 80-bits de segurança:

(1) cifra simétrica com chave de 80 bits, porque corresponde a  $2^{80}$  possibilidades diferentes da chave

(2) MAC com chave de 80 bits, porque corresponde a  $2^{80}$  possibilidades diferentes da chave

(3) output das funções de hash de 160 bits, porque descobrir uma colisão necessita de  $2^{n/2}$  operações (cada par é uma possível colisão), logo  $2^{160/2} = 2^{80}$

"Se o melhor ataque que se conhece contra uma primitiva criptográfica é exponencialmente mais rápido do que enumerar todas as chaves possíveis, então essa primitiva é insegura." Esta afirmação é verdadeira? Justifique.

Não, a afirmação é falsa porque depende do modelo de segurança. Se o modelo de segurança for  $(t, e)$  e  $t$  for um número de operações exponencialmente inferior do que enumerar todas as chaves possíveis, então a primitiva criptográfica considera-se segura para esse modelo de segurança.

Contudo, a afirmação é ambígua. Se existe um ataque exponencialmente mais rápido do que enumerar todas as chaves possíveis, então a primitiva criptográfica será provavelmente insegura para o modelo de segurança pretendido, porque  $n$  bits de chave não estão a oferecer segurança de  $n$  bits.

### 1.5. Segurança Heurística vs Segurança Demonstrável.

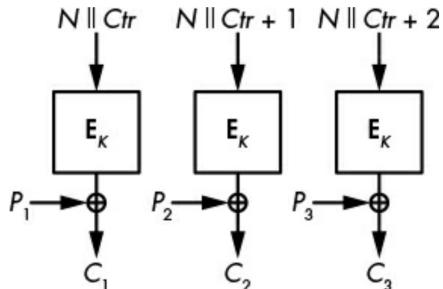
Explique os conceitos de segurança heurística e segurança demonstrável, dando um exemplo de uma construção/primitiva criptográfica cuja segurança seja estabelecida com base em cada uma destas aproximações.

Segurança heurística assenta em analistas criptográficos que tentam e falham constantemente em quebrar uma primitiva criptográfica e refutar a sua segurança de  $n$  bits - muitas pessoas talentosas tentaram quebrar a cifra e falharam. Por exemplo, AES.

Segurança demonstrável assenta em provas matemáticas, em que quebrar uma primitiva criptográfica implica resolver um problema matemático difícil ou quebrar outro esquema - provar que quebrar um esquema é pelo menos tão difícil como resolver outro problema reconhecidamente difícil, garantindo que a primitiva criptográfica continua difícil enquanto o outro problema continuar difícil. Por exemplo, One-Time Pad.

**Grupo 2 - Criptografia Simétrica (25%)****2.1. Aplicações de cifras de bloco.**

- a) Identifique o esquema na figura, explique as propriedades de segurança que permite garantir quando utilizado corretamente. Explique o que uma utilização correta significa do lado do emissor e do receptor indicando nomeadamente o que é transmitido entre os dois, e o pressupostos de segurança devem ser cumpridos.



O esquema é o Counter Block Mode (CTR), modo de operação de uma cifra por blocos como o AES.

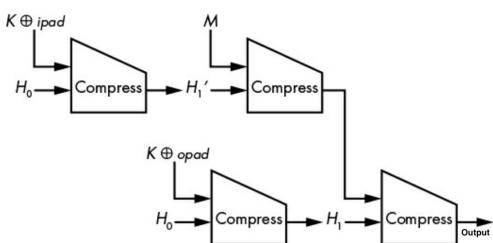
Quando utilizado corretamente, garante a confidencialidade dos dados porque não revela duplicados - é seguro.

O emissor nunca deve repetir o nonce N com a mesma chave K e deve ter acordada a chave K com o receptor, transmitindo o criptograma C e o nonce N.

O receptor deve receber o nonce N (podendo começar a pré-processar a desencriptação) e o criptograma C, sabendo a chave K.

A encriptação e a desencriptação são idênticas, fazendo o XOR de cada bloco da mensagem/criptograma com a encriptação do nonce N concatenado com o counter Ctr incrementado em cada bloco e inicializado a 0.

- b) Considere o seguinte diagrama. Identifique o esquema na figura, explique as propriedades de segurança que permite garantir quando utilizado corretamente. Explique o que uma utilização correta significa do lado do emissor e do receptor indicando nomeadamente o que é transmitido entre os dois, e o pressupostos de segurança devem ser cumpridos.



O esquema é o Hash-Based MAC (HMAC), código de autenticação de mensagens - hashing com chave.

Quando utilizado corretamente, garante a autenticidade dos dados porque retorna uma PRF - é seguro.

O emissor deve definir o valor de H0 - constante e público - e deve ter acordada a chave K com o receptor, transmitindo a tag T.

O receptor deve receber a tag T e verificá-la, comparando-a com o resultado do mesmo processo para a mensagem, o mesmo H0 e a mesma chave K.

Os valores de ipad e opad são fixos.

A função de hash subjacente deve ser resistente a colisões ou, pelo menos, uma PRF

- 2.2. Explique porque é que uma cifra de blocos é o componente central das funções de hash da família SHA-1 e SHA-2 que utilizam a construção de Merkle-Damgård.

As funções de hash da família SHA-1 e SHA-2 utilizam a construção de Merkle-Damgård que, por sua vez, utiliza a construção da Davies-Meyer como função de compressão.

A função de compressão na construção de Davies-Meyer é uma cifra por blocos, que recebe um input de tamanho arbitrário e retorna um output de tamanho fixo, dado por  $H_i = E(M_i, H_{i-1}) + H_{i-1}$ , sendo resultado da divisão da mensagem em blocos e da aplicação da cifra por blocos.

### Grupo 3 - Funções de Hash (20%)

**3.1.** Considere uma função de hash  $h : \{0,1\}^* \rightarrow \{0,1\}^n$ , que mapeia bit strings de tamanho arbitrário em bit strings de tamanho fixo  $n$ .

a) Explique o conceito de *resistência a colisões* desta função de hash.

Resistência a colisões é ser difícil encontrar  $m_0$  e  $m_1$  diferentes tais que  $h(m_0) = h(m_1)$ , devendo ser necessárias  $2^{n/2}$  operações - colisões devem ser tão difíceis de encontrar como a menagem original, pelo que os atacantes não devem ser capazes de encontrar duas menagens distintas que retornem o mesmo valor de hash

b) Descreva um ataque que demonstre que nenhuma função de hash deste tipo pode ser *resistente a colisões* contra atacantes com poder computacional ilimitado.

Nenhuma função de hash desse tipo pode ser resistente a colisões contra atacantes com poder computacional ilimitado porque colisões vão sempre existir devido ao pigeonhole principle.

Um ataque possível é o ataque do aniversário: computar  $2^{n/2}$  hashes de  $2^{n/2}$  mensagens escolhidas e armazenar todos os pares mensagem-hash numa lista, ordená-la em relação ao valor de hash e procurar duas entradas consecutivas com o mesmo valor de hash.

**3.2.** A construção Sponge, na qual se baseia a função de hash SHA-3 utiliza um componente interno que não é uma cifra de blocos. Explique que componente é este, e porque razão se abandonou a construção de Merkle-Damgård.

O componente interno da construção Sponge é uma permutação de  $l$  bits, que permite construir uma função de hash com output de comprimento arbitrário, absorvendo/permutando o input no estado interno e espremendo/buscando o output a partir do estado interno.

A construção de Merkle-Damgård foi abandonada para a construção Sponge porque esta é mais simples, mais versátil e permite outputs de tamanho variável.

### Grupo 4 - Cifras Autenticadas (20%)

**4.1.** Descreva a construção Encrypt-And-Mac de uma cifra autenticada, e explique as suas vantagens e desvantagens.

A construção Encrypt-And-Mac de uma cifra autenticada encripta a mensagem e gera a tag a partir da mensagem -  $C = E(K_1, P)$  e  $T = \text{MAC}(K_2, P)$ , sendo o output  $(C, T)$ .

A vantagem é que pode ser paralelizável, por serem operações independentes.

A desvantagem é que o MAC fica exposto, mesmo que seja seguro pode revelar informação sobre a mensagem e permite que criptograma malicioso/corrompido seja desencriptado antes da autenticação.

**4.2.** Que propriedades de segurança deve uma cifra AEAD garantir para a mensagem (plaintext), os meta-dados (associated data) e o criptograma (ciphertext)?

Uma cifra AEAD deve garantir que a mensagem é confidencial, autenticada e não repetida/omitida/removida, que os meta-dados são apenas autenticados e que o criptograma não revela informações sobre a menagem - o adversário não deve conseguir distinguir duas encriptações de qualquer mensagem nem forjar mensagens novas.

**4.3.** Como se utiliza o modo de funcionamento SIV (Synthetic IV)? Que propriedades de segurança adicionais oferece em relação a outras cifras AEAD? Recorde que esta construção calcula  $T = \text{MAC}(K_1, N, A, M)$  e  $C = \text{ENC}(K_2, T, M)$ .

O modo SIV é uma composição genérica de encriptação autenticada baseada em nonce e PRF, reforçando o AES-GCM contra a reutilização do nonce.

$T = \text{PRF}(K_1, A \parallel P \parallel N)$  e  $C = E(K_2, N = T, P)$ , sendo o output  $(C, T)$ .

O vetor de inicialização da cifra por blocos para a encriptação é a tag, que por sua vez resulta da concatenação dos meta-dados com a mensagem e com o nonce - daí IV sintético.

Se o nonce se repetir, mas os meta-dados ou a mensagem não, então não há problema e a tag usada como nonce de encriptação é muito provavelmente fresca.

# Complexidade Computacional

Segunda Lei da Termodinâmica: é necessário um mínimo de energia para mudar o estado de um único bit -  $E = kT$

Dado que o modelo de segurança usado pela criptografia moderna não é compatível com "segurança por esfumamento", a sua segurança deve basear-se em resultados de Complexidade Computacional

Máquina de Turing:

1. Uma máquina de Turing pode escrever na fita e ler da fita
2. A cabeça de escrita/leitura pode mover-se para a esquerda e para a direita
3. A fita é infinita
4. Os estados especiais para aceitação e rejeição tomam efeito imediatamente

Linguagem Turing-Reconhecível: uma máquina de Turing reconhece-a - se dada uma palavra  $w \in L$  como input, então a TM vai sempre parar e dar uma resposta positiva

Linguagem Turing-Decidível: uma máquina de Turing decide-a

Tese Church-Turing: todos os modelos de expressividade suficiente são equivalentes

Complexidade Computacional: medida do tempo necessário para completar uma tarefa como uma função do tamanho do input

Notação "Big O":  $f: \mathbb{N} \rightarrow \mathbb{R}$ ,  $f(n) = O(g(n))$ ,  $\exists k > 0 \exists n_0 \forall n > n_0: |f(n)| \leq k g(n)$

$g(n)$  é um limite superior (assintótico) para  $f(n)$

Notação "small o":  $f(n) = o(g(n))$ ,  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$

Teoria da Complexidade: considera só um tipo especial de programas que recebe um input e retorna uma resposta de aceitação ou rejeição desse input

Linguagem Definida pelo Programa: conjunto de palavras aceites pelo programa

Decisão da Linguagem: o programa decide, para cada palavra de input, se pertence à linguagem ou não

Classe de Complexidade Temporal:  $\text{TIME}(t(n))$  - coleção de todas as linguagens que não são decidíveis por uma máquina de Turing em tempo  $O(t(n))$

Classe de Complexidade Espacial:  $\text{SPACE}(t(n))$  - coleção de todas as linguagens que não são decidíveis por uma máquina de Turing usando espaço  $O(t(n))$  na sua fita

$$\rightarrow \text{TIME}(n^2) \subseteq \text{SPACE}(n^2)$$

Classe P:  $P = \bigcup_{k \in \mathbb{N}} \text{TIME}(n^k)$  - classe de complexidade de todas as funções que podem ser decididas em tempo polinomial por uma máquina de Turing

Ex:  $\gcd(a, b) = 1$ ; números primos; CFL

Classe PSPACE:  $\text{PSPACE} = \bigcup_{k \in \mathbb{N}} \text{SPACE}(n^k)$  - classe de complexidade das funções que podem ser decididas por uma máquina de Turing usando espaço polinomial na fita

NTIME( $t(n)$ ): classe de complexidade temporal - coleção de todas as linguagens que não são decidíveis em tempo  $O(t(n))$  por uma máquina de Turing não-determinística

Classe NP: não-determinismo polinomial -  $NP = \bigcup_{k \in \mathbb{N}} \text{NTIME}(n^k)$

linguagens decidíveis por uma máquina de Turing não-determinista em tempo polinomial  
linguagens verificáveis por uma máquina de Turing (determinística) em tempo polinomial

uma máquina de Turing com input  $(w, c)$  consegue decidir se  $w$  faz parte da linguagem usando  $c$  como informação adicional - o tempo só depende de  $w$

$$\text{Ex: } a^k \equiv n \pmod{p}; \text{ números compostos; clique; } \{(S, t) \mid S = \{x_1, \dots, x_k\} \wedge S' \subseteq S \wedge \sum x_i = t\}$$

# Problemas Difíceis

P: classe de linguagens com decisão de pertença rápida P? NP

NP: classe de linguagens com verificação de pertença rápida

$$\rightarrow NP \subseteq EXPTIME = \bigcup_{k \in \mathbb{N}} TIME(2^{n^k})$$

Função Computável em Tempo Polinomial: se existe uma máquina de Turing de tempo polinomial que para só com  $f(w)$  na sua fita, quando começa com input  $w$

Redução em Tempo Polinomial:  $A \leq_p B \iff \forall w, w \in A \Rightarrow f(w) \in B$

redução polinomial  $A \rightarrow B$

Se  $A \leq_p B$  e  $B \in P$ , então  $A \in P$

Linguagem NP-Completa:  $B \in NP \wedge \forall A \in NP, A \leq_p B$   
NP-DIFÍCIL

Se  $B$  é NP-Completa e  $B \in P$ , então  $P = NP$

Se  $B$  é NP-Completa e  $B \leq_p C$  para  $C \in NP$ , então  $C$  é NP-Completa

O conjunto de linguagens (problemas) NP-Completas é o conjunto das linguagens (problemas) mais difíceis em NP

Ex: TSP; clique; knapsack ( $x \in \sum_{i \in \mathbb{N}} q_i y_i$ )

$$\rightarrow P \subseteq NP \subseteq PSPACE = NSPACE \subseteq EXPTIME$$

Teorema Fundamental da Aritmética: para qualquer inteiro positivo  $n$ , existe uma única fatorização de  $n$  como produto de primos crescentes

Problema da Fatorização: encontrar os números primos  $p$  e  $q$  tais que  $N = p \times q$

Problema do Logaritmo Discreto: encontrar  $y$  tal que  $g^y \equiv x \pmod{p}$

# RSA

O que é necessário para abrir um "canal seguro"?

Com criptografia simétrica e  $N$  atores, potencialmente  $\frac{N^2 + N}{2}$  chaves diferentes

• As matrizes não são universais!

**Modelo de Criptografia de Chave Pública:** em vez de uma chave por canal, cada agente tem duas chaves

1.  $K_p$  — chave pública

2.  $K_s$  — chave secreta (privada)

$$\rightarrow D_{K_s}(E_{K_p}(m)) = m$$

$$\rightarrow D_{K_p}(E_{K_p}(m)) \neq m$$

• É ainda melhor se  $D_{K_p}(E_{K_s}(m)) = m$ , mas não é necessário

$$a \mid b \Leftrightarrow \exists k \in \mathbb{Z}, ak = b$$

$$1. a \mid 1 \Rightarrow a = \pm 1$$

$$2. ((a \mid b) \wedge (b \mid a)) \Rightarrow a = \pm b$$

$$3. (\forall b)(b \mid 0)$$

$$4. (b \mid g) \wedge (b \mid h) \Rightarrow ((\forall m, n \in \mathbb{Z})(b \mid (mg + nh)))$$

$$5. ((a \mid (b+c)) \wedge (a \mid b)) \Rightarrow a \mid c$$

$$a \mid b \Rightarrow b = ak$$

$$a \mid (b+c) \Rightarrow (b+c) = ak' = (ak + c) \Rightarrow a(k' - k) = c \Rightarrow a \mid c \quad \square$$

Número Primo:  $\forall p \in \mathbb{Z}, p > 1 \wedge (\forall d \in \mathbb{N}, d \mid p \rightarrow (d=1 \vee d=p))$

Máximo Divisor Comum:  $g \mid a \wedge g \mid b \wedge ((\forall d)(d \mid a \wedge d \mid b) \rightarrow d \mid g) \rightarrow g = (a, b)$

Inteiros Co-Primos:  $(a, b) = 1$

**Teorema:**  $d = (a, b)$  é a menor combinação linear positiva de  $a$  e  $b$

$$\downarrow \cdot S = \{ ax + by : x, y \in \mathbb{Z} \wedge ax + by > 0 \} \rightarrow S \neq \emptyset \quad (a^2 + b^2 \in S)$$

$$\cdot d = \min(S)$$

$$\cdot d' : d' \mid a \wedge d' \mid b \rightarrow d = ax + by = d'q_1x + d'q_2y = d'(q_1x + q_2y) \rightarrow d' \mid d$$

$$\cdot a = dq + r, \quad 0 \leq r < d \rightarrow r = a - dq = a - (ax + by)q = a(1 - qx) + b(-qy)$$

$\rightarrow r$  é uma combinação linear de  $a$  e  $b$

$r > 0 \Rightarrow r \in S$ , mas, como  $r < d$ , isso seria absurdo porque  $d = \min(S)$

$$\rightarrow 0 \leq r \Rightarrow r = 0 \quad \text{e } d \mid a$$

Com o mesmo argumento, mostra-se que  $d \mid b$

$$\rightarrow d = (a, b) \quad \square$$

**Teorema:** um intérro  $p$  é primo se e só se  $(\forall a, b \in \mathbb{Z} \setminus \{0\}) (p \mid ab \Rightarrow p \mid a \vee p \mid b)$

$\downarrow$  Seja  $p$  um primo e  $p \mid ab$ . Se  $p \nmid a$ , então a prova está feita. Se  $p \mid a$ , então, como  $p$  não tem divisores,  $(p, a) = 1 \rightarrow (\exists x, y) 1 = ax + py \Rightarrow b = bax + bpy \Rightarrow p \mid b$

$\leftarrow$  Seja  $p$  que satisfaça (1) e  $S = \{ n \mid n > 1 \wedge n \mid p \} \rightarrow S \neq \emptyset \quad (p \in S)$

Seja  $m = \min(S)$ ,  $m \mid p \wedge (\exists k) m \mid k = p$

Como  $p$  satisfaz (1),  $p \mid m$  ou  $p \mid k$ , mas  $p \mid k \Rightarrow k > p \Rightarrow p = k \Rightarrow m = 1$  CONTRADIÇÃO

**Teorema Fundamental da Aritmética:** todo o intérro positivo pode ser escrito de uma única forma como um produto de primos crescentes

$a, b, q \in \mathbb{Z}, r \in \mathbb{N}, 0 \leq r \leq b \wedge a = bq + r \Leftrightarrow a \bmod b = r \Leftrightarrow a \equiv r \pmod{b} \Leftrightarrow b \mid (a - r)$

$$\mathbb{Z}_n = \{0, 1, \dots, n-1\}$$

$$1. (w+x) \bmod n = (x+w) \bmod n$$

$$2. (w \times x) \bmod n = (x \times w) \bmod n$$

$$3. ((w+x)+y) \bmod n = (w+(x+y)) \bmod n$$

$$4. ((w \times x) \times y) \bmod n = (w \times (x \times y)) \bmod n$$

$$5. (w \times (x+y)) \bmod n = ((w \times x) + (w \times y)) \bmod n$$

$$6. (0+w) \bmod n = w \bmod n$$

$$7. (1 \times w) \bmod n = w \bmod n$$

$$8. (\forall w \in \mathbb{Z}_n) (\exists z \in \mathbb{Z}_n) (w + z = 0 \bmod n)$$

$\cdot (a+b) \equiv (a+c) \pmod{n} \Rightarrow b \equiv c \pmod{n}$

POIS

$\forall a \in \mathbb{Z}_n \exists b \in \mathbb{Z}_n a+b \equiv 0 \pmod{n}$

$\downarrow$  MAS

$\cdot ((a \times b) \equiv (a \times c) \pmod{n} \Rightarrow b \equiv c \pmod{n})$  se  $(a, n) = 1$

$\cdot \text{se } (a, n) \neq 1, f: \mathbb{Z}_n \rightarrow \mathbb{Z}_n$ , então  $f$  não é sobrejetiva

$\cdot m = (a, n) \rightarrow m | a \wedge m | n \rightarrow ((\exists x)(\exists y) a = xm \wedge n = ym)$

$\hookrightarrow f(0) = a \times 0 = 0 \wedge f(y) = ay = xy_m = x_n = 0 \pmod{n}$

**Teorema de Fermat:** Seja  $p$  um primo e  $a$  tal que  $p \nmid a$ , então  $a^{p-1} \equiv 1 \pmod{p}$

$f: \mathbb{Z}_p^* \rightarrow \mathbb{Z}_p^*$

$\begin{cases} z \mapsto az \pmod{p} \\ f \text{ é sobrejetiva} \end{cases} \rightarrow \begin{cases} a \pmod{p}, 2a \pmod{p}, \dots, (p-1)a \pmod{p} \\ \prod_{i=1}^{p-1} ai \pmod{p} = \prod_{i=1}^{p-1} i \end{cases}$

$\begin{aligned} a^{p-1} &\equiv (p-1)! \pmod{p} \\ \text{Como } ((p-1)!, p) &= 1 \rightarrow a^{p-1} \equiv 1 \pmod{p} \quad \square \end{aligned}$

$$\phi(n) = \left| \{i \mid i < n \wedge (i, n) = 1 \} \right|$$

$\cdot$  Se  $p$  é primo, então  $\phi(p) = p-1$

$\cdot$  Se  $p \neq q$  são primos, então o conjunto de elementos  $n$  de  $\mathbb{Z}_{pq}$  tal que  $(n, pq) \neq 1$  é  $\{p, 2p, \dots, (q-1)p, q, 2q, \dots, (p-1)q\}$

$\rightarrow \phi(pq) = (pq-1) - ((q-1) + (p-1)) = pq - (p+q) + 1 = (p-1)(q-1) = \phi(p)\phi(q)$

**Teorema de Euler:**  $(a, n) = 1 \Rightarrow a^{\phi(n)} \equiv 1 \pmod{n}$

$\downarrow$

$\cdot R = \{x \in \mathbb{N}: 0 < x < n \wedge (n, x) = 1\} = \{x_1, x_2, \dots, x_{\phi(n)}\}$

$\cdot S = \{ax_1 \pmod{n}, ax_2 \pmod{n}, \dots, ax_{\phi(n)} \pmod{n}\}$

$\cdot ((a, n) = 1 \wedge ((x_i, n) = 1)) \Rightarrow ((ax_i, n) = 1) \rightarrow S \subseteq R$

$\cdot (ax_i \pmod{n}) = (ax_j \pmod{n}) \Rightarrow x_i = x_j \rightarrow S = R$

$$\rightarrow \prod_{i=1}^{\phi(n)} a x_i \equiv \prod_{i=1}^{\phi(n)} x_i \pmod{n} \Leftrightarrow a^{\prod_{i=1}^{\phi(n)}} \prod_{i=1}^{\phi(n)} x_i \equiv \prod_{i=1}^{\phi(n)} x_i \pmod{n} \Leftrightarrow a^{\phi(n)} \equiv 1 \pmod{n}$$

**Corolário:** Sejam  $p$  e  $q$  números primos,  $n = pq$  e  $0 < m < n$ , então  $m^{\phi(n)+1} \equiv m \pmod{n}$

$$\begin{aligned} \cdot (m, n) \neq 1 &\Leftrightarrow p \mid m \wedge q \mid m \\ \cdot p \mid m \quad (m = cp) &\Rightarrow (q, m) = 1 \quad \text{SENAO, } p \mid m \wedge q \mid m \wedge m < pq \\ \rightarrow m^{\phi(q)} \equiv 1 \pmod{q} &\Leftrightarrow (m^{\phi(q)})^{\phi(p)} \equiv 1 \pmod{q} \Leftrightarrow m^{\phi(n)} \equiv 1 \pmod{q} \\ \Leftrightarrow (\exists k \in \mathbb{N}) \quad (m^{\phi(n)} = 1 + kq) &\Leftrightarrow m^{\phi(n)+1} = m + kcpq = m + kcn \Leftrightarrow m^{\phi(n)+1} \equiv m \pmod{n} \end{aligned}$$

## RSA:

1. Gere dois primos grandes de magnitude comparável:  $p$  e  $q$
2. Definir  $n = pq$
3. Gere  $e < \phi(n) = (p-1)(q-1)$  tal que  $(e, \phi(n)) = 1$
4. Computar  $d = e^{-1} \pmod{\phi(n)}$

Chave Pública:  $(n, e)$   
 Chave Privada:  $(n, d)$

$$\begin{aligned} \text{Cifrar: } m^e \pmod{n} \\ \text{Decifrar: } (m^e \pmod{n})^d \pmod{n} \equiv m^{ed} \pmod{n} \equiv m^{k\phi(n)+1} \pmod{n} = m \end{aligned}$$

Teste de Primilidade:  $\mathcal{O}(n^{12} \log^{12} n) \rightsquigarrow \mathcal{O}(n^6 \log^6 n) \rightsquigarrow \dots$

**Teorema de Miller-Rabin**: Se  $p$  é um primo ímpar, então a equação  $x^2 \equiv 1 \pmod{p}$  não tem duas soluções:  $x \equiv 1 \vee x \equiv -1$

- Se  $x$  é uma solução, então  $x^2 - 1 \equiv 0 \pmod{p} \Leftrightarrow (x+1)(x-1) \equiv 0 \pmod{p}$
- $\rightarrow p \mid (x+1) \vee p \mid (x-1)$
- Se  $p \mid (x-1)$ , então  $\exists k \quad (x-1 = kp) \rightarrow x \equiv 1 \pmod{p}$
- Se  $p \mid (x+1)$ , então  $\exists k \quad (x+1 = kp) \rightarrow x \equiv -1 \pmod{p}$  □

**Negativo:** Se existe uma solução para  $x^2 \equiv 1 \pmod{p}$  que não é  $x = \pm 1$ , então  $p$  não é primo.

**Teorema:** Seja  $p$  um primo ímpar e  $a$  um inteiro tal que  $p \nmid a$ , então é possível escrever  $p-1 = 2^k d$ , sendo  $d$  ímpar. Então, um dos seguintes é verdade:

1.  $a^d \equiv -1 \pmod{p}$
2.  $\exists x \in \{a^{2^0 d}, a^{2^1 d}, \dots, a^{2^{k-1} d}\}, x \equiv 1 \pmod{p}$  pelo Teorema de Fermat

Como cada termo é o quadrado do anterior, então ou

1.  $a^d \equiv 1 \pmod{p}$
  2.  $\exists i, 0 \leq i < k: a^{2^i d} \equiv 1 \pmod{p} \wedge a^{2^{i+1} d} \not\equiv 1 \pmod{p}$
- Pelo Teorema de Miller-Rabin,  $a^{2^{i+1} d} \equiv -1 \pmod{p}$

Seja  $n$  um número ímpar e  $n-1 = 2^k d$ , com  $d$  ímpar

**Testemunha de Miller-Rabin:** inteiro  $a$  tal que  $(a, n) = 1$  e

1.  $a^d \not\equiv 1 \pmod{p}$
2.  $\forall i, 0 \leq i < k \Rightarrow a^{2^i d} \not\equiv -1 \pmod{p}$

Dado um número ímpar composto  $n$  e um inteiro  $a$  ( $0 < a < n$ ), a probabilidade de  $a$  não ser uma testemunha de Miller-Rabin é  $1/2$ .

Se se repete o teste com  $m$  valores aleatoriamente gerados de  $a$  e nenhum delas for uma testemunha de Miller-Rabin, a probabilidade de  $n$  ser composto é  $1 - 2^{-m}$ .

PKC é cerca de 1000 vezes mais lento do que cifras simétricas normais.  
PKC é vulnerável a ataques de dicionário se a mensagem provém de um conjunto pequeno de mensagens administráveis.

RSA é um homomorfismo multiplicativo,  $E_K(x_1) E_K(x_2) = (x_1^e \pmod{n}) (x_2^e \pmod{n}) = (x_1 x_2)^e \pmod{n} = E_K(x_1 x_2)$ , pelo que é vulnerável.

Optimal Asymmetric Encryption Padding (OAEP): torna RSA não-maleável

• OAEP usa um PRNG para garantir a indistingibilidade e não-maleabilidade dos criptogramas, ao tornar a encriptação probabilística

Assinar com RSA:  $m^d \pmod{n}$  — a verificação é a "decifração" com  $e$

• É possível forjar assinaturas de  $0^d \pmod{n} = 0$ ,  $1^d \pmod{n} = 1$  e de  $(n-1)^d \pmod{n} = (-1)^{\frac{d}{\text{mod } 2}}$

"Blinding Attack": dado  $r$  tal que  $r^e m \pmod{n}$  é uma mensagem planarível de ser assinada, então  $s = (r^e m)^d = r m^d$ , obtendo-se  $m^d$

PSS:

1. Escolher uma string aleatória de  $r$ -bytes  $r$  usando o PRNG

$$2. m' = 0^{16} \parallel \text{Hash1}(m) \parallel r$$

3. Computar a string de  $h$  bytes  $h = \text{Hash1}(m')$

$$4. \text{Definir } l = 00 \dots 00 \parallel 01 \parallel r$$

$$5. \text{Definir } l = l \oplus \text{Hash2}(h)$$

6. Converter  $p = l \parallel h \parallel l$  para um número  $x < n$

7. Computar  $x^d \pmod{n}$  para obter a assinatura

Teorema Chines dos Restos: seja  $m = \prod_{i=1}^n m_i$  com  $(\forall i, j) (i \neq j \Rightarrow (m_i, m_j) = 1)$ ,

então

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ \dots \\ x \equiv a_2 \pmod{m_2} \end{cases}$$

tem uma solução para  $x$  e todas as soluções  $y$  do sistema são  $y \equiv x \pmod{m}$

$$1. x_p = y^{\wedge} \pmod{p}, \quad \wedge = d \pmod{(p-1)}$$

$$2. x_q = y^t \pmod{q}, \quad t = d \pmod{(q-1)}$$

$$3. x = x_p q \left( \frac{1}{q} \pmod{p} \right) + x_q p \left( \frac{1}{p} \pmod{q} \right) \pmod{n}$$

# RSA

O que é necessário para abrir um "canal seguro"?

Com criptografia simétrica e  $N$  atores, potencialmente  $\frac{N^2 + N}{2}$  chaves diferentes

**Modelo de Criptografia de Chave Pública:** Em vez de uma chave por canal, cada agente tem duas chaves

1.  $K_p$  - chave pública

2.  $K_s$  - chave secreta (privada)

$$\rightarrow D_{K_s}(E_{K_p}(m)) = m$$

$$\rightarrow D_{K_p}(E_{K_p}(m)) \neq m$$

E ainda melhor se  $D_{K_p}(E_{K_s}(m)) = m$ , mas não é necessário

$$a | b \Leftrightarrow \exists k \in \mathbb{Z}, ak = b$$

**Número Primo:**  $\forall p \in \mathbb{Z}, p > 1 \wedge (\forall d \in \mathbb{N}, d | p \rightarrow (d=1 \vee d=p))$

**Máximo Divisor Comum:**  $g | a \wedge g | b \wedge ((\forall d)(d | a \wedge d | b) \Rightarrow d | g) \rightarrow g = (a, b)$

**Inteiros Co-Primos:**  $(a, b) = 1$

**Teorema:**  $d = (a, b)$  é a menor combinação linear positiva de  $a$  e  $b$

**Teorema:** Um inteiro  $p$  é primo se e só se  $(\forall a, b \in \mathbb{Z} \setminus \{0\})(p | ab \Rightarrow p | a \vee p | b)$

**Teorema Fundamental da Aritmética:** Todo o inteiro positivo pode ser escrito de uma única forma como um produto de primos crescentes

**Teorema de Fermat:** Seja  $p$  um primo e  $a$  tal que  $p \nmid a$ , então

$$a^{p-1} \equiv 1 \pmod{p}$$

$$\phi(n) = \left| \{ i \mid i < n \wedge (i, n) = 1 \} \right|$$

**Teorema de Euler:**  $(a, n) = 1 \Rightarrow a^{\phi(n)} \equiv 1 \pmod{n}$

**Corolário:** Sejam  $p$  e  $q$  números primos,  $n = pq$  e  $0 < m < n$ , então

$$m^{\phi(n)+1} \equiv m \pmod{n}$$

**RSA:**

1. Gere dois primos grandes de magnitude comparável:  $p$  e  $q$
2. Defini  $n = pq$
3. Gere  $e < \phi(n) = (p-1)(q-1)$  tal que  $(e, \phi(n)) = 1$
4. Computar  $d = e^{-1} \pmod{\phi(n)}$

**Chave Pública:**  $(n, e)$

**Chave Privada:**  $(n, d)$

**Cifrar:**  $m^e \pmod{n}$

**Decifrar:**  $(m^e \pmod{n})^d \pmod{n} \equiv m^{ed} \pmod{n} \equiv m^{k\phi(n)+1} \pmod{n} = m$

**Teorema de Miller-Rabin:** Se  $p$  é um primo ímpar, então a equação  $x^2 \equiv 1 \pmod{p}$  só tem duas soluções:  $x \equiv 1 \vee x \equiv -1$

**Negativo:** Se existe uma solução para  $x^2 \equiv 1 \pmod{p}$  que não é  $x = \pm 1$ , então  $p$  não é primo

**Teorema:** Seja  $p$  um primo ímpar e  $a$  um inteiro tal que  $p \nmid a$ , então é possível escrever  $p-1 = 2^k d$ , sendo  $d$  ímpar. Então, um dos seguintes é verdade:

1.  $a \equiv -1 \pmod{p}$

2.  $\exists x \in \{a^{2^0 d}, a^{2^1 d}, \dots, a^{2^{k-1} d}\}, x \equiv 1 \pmod{p}$  pelo Teorema de Fermat

Seja  $n$  um número ímpar e  $n-1 = 2^k d$ , com  $d$  ímpar

**Testemunha de Miller-Rabin:** inteiro  $a$  tal que  $(a, n) = 1$  e

1.  $a^d \not\equiv 1 \pmod{p}$

2.  $\forall i, 0 < i < k \Rightarrow a^{2^i d} \not\equiv -1 \pmod{p}$

Dado um número ímpar composto  $n$  e um inteiro  $a$  ( $0 < a < n$ ), a probabilidade de  $a$  não ser uma testemunha de Miller-Rabin é  $1/2$ .

• RSA é um homomorfismo multiplicativo,  $E_K(x_1)E_K(x_2) = (x_1^e \pmod{n})$   
 $(x_2^e \pmod{n}) = (x_1x_2)^e \pmod{n} = E_K(x_1x_2)$ , pelo que é maleável e vulnerável

Optimal Asymmetric Encryption Padding (OAEP): torna RSA não-maleável

Assim com RSA:  $m^d \pmod{n}$

• É possível forjar assinaturas de  $0^d \pmod{n} = 0$ ,  $1^d \pmod{n} = 1$  e de  
 $(n-1)^d \pmod{n} = (-1)^{\frac{d}{\text{mod } 2}}$

"Blinding Attack": dado  $r$  tal que  $r^e m \pmod{n}$  é uma mensagem planarível de ser assinada, então  $s = (r^e m)^d = r m^d$ , obtendo-se  $m^d$

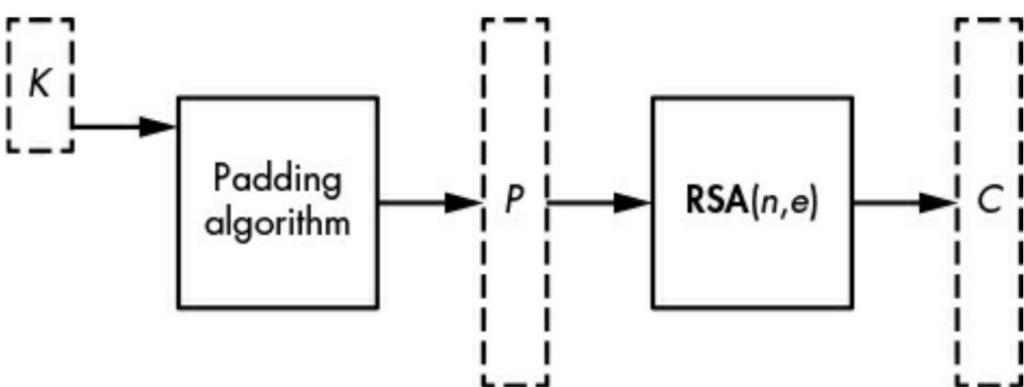
PSS:

1. Escolher uma string aleatória de  $r$ -bytes  $r$  usando o PRNG
2.  $m' = 0^{16} \parallel \text{Hash1}(m) \parallel r$
3. Computar a string de  $h$  bytes  $h = \text{Hash1}(m')$
4. Definir  $\ell = 00 \dots 00 \parallel 01 \parallel r$
5. Definir  $\ell = \ell \oplus \text{Hash2}(h)$
6. Converter  $\ell = \ell \parallel h \parallel b_c$  para um número  $x < n$
7. Computar  $x^d \pmod{n}$  para obter a assinatura

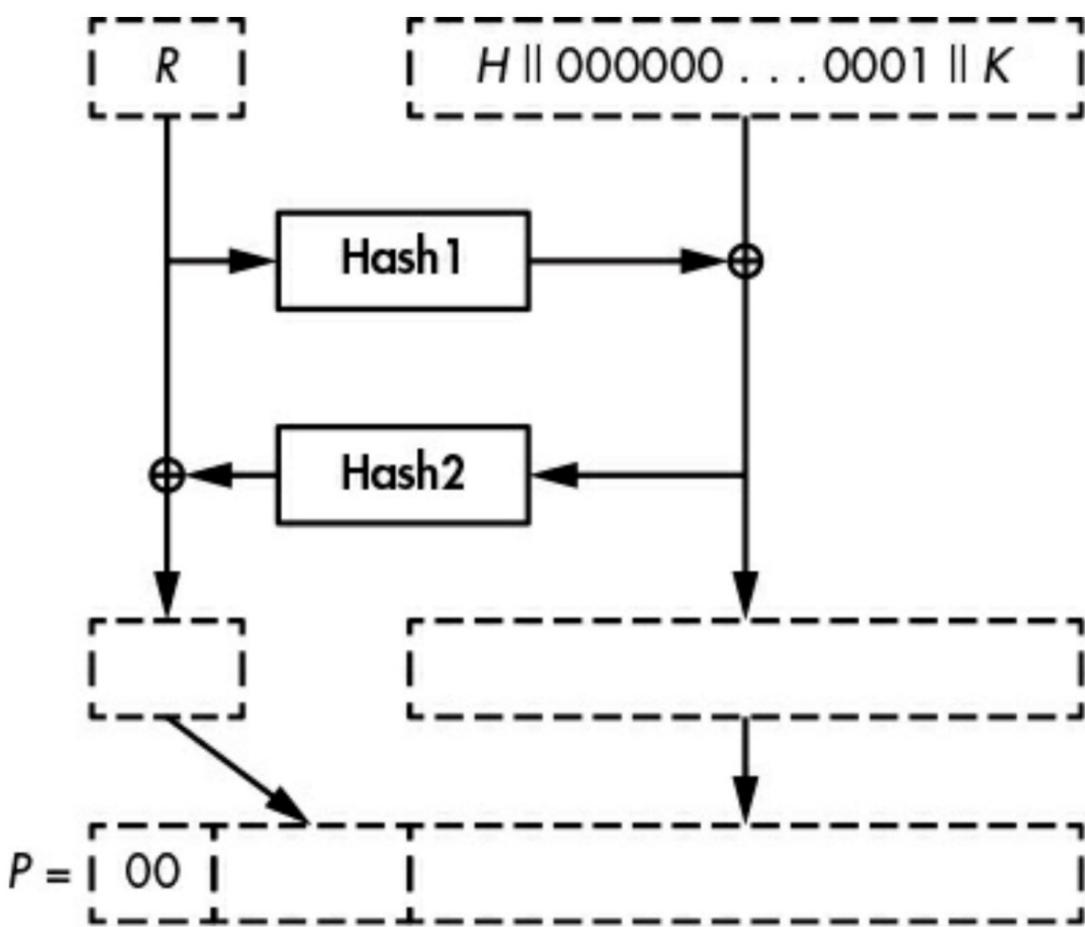
Teorema Chines dos Restos: seja  $m = \prod_{i=1}^n m_i$  com  $(\forall i, j) (i \neq j \Rightarrow (m_i, m_j) = 1)$ ,  
então  $\begin{cases} x \equiv a_1 \pmod{m_1} \\ \dots \\ x \equiv a_2 \pmod{m_2} \end{cases}$

tem uma solução para  $x$  e todas as soluções  $y$  do sistema são  $y \equiv x \pmod{m}$

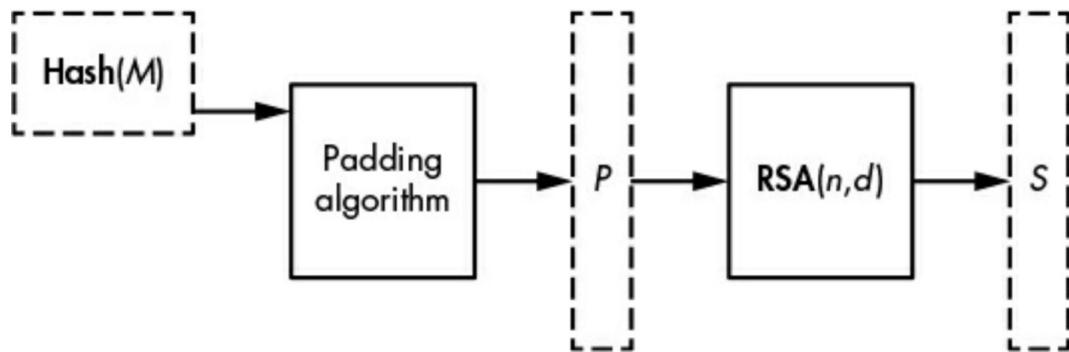
1.  $x_p = y^{\lambda} \pmod{p}$ ,  $\lambda = d \pmod{(p-1)}$
2.  $x_q = y^t \pmod{q}$ ,  $t = d \pmod{(q-1)}$
3.  $x = x_p q (1/q \pmod{p}) + x_q p (1/p \pmod{q}) \pmod{n}$



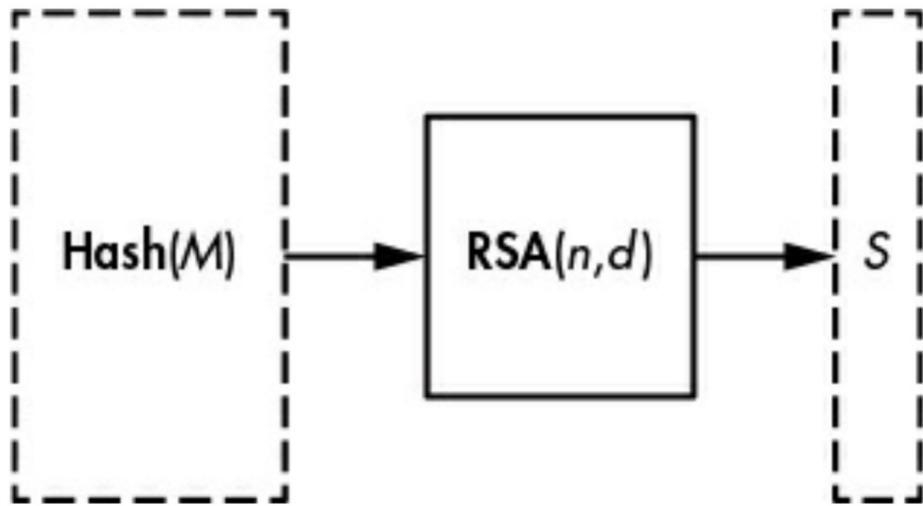
## Optimal Asymmetric Encryption Padding (OAEP)



# The PSS Signature Standard



# Full Domain Hash Signatures



# Problema do Logaritmo Discreto

**Problema do Logaritmo Discreto:** Sejam  $a$  e  $n$  inteiros tais que  $(a, n) = 1$ , então, pelo Teorema de Euler, a equação  $a^m \equiv 1 \pmod{n}$  tem pelo menos uma solução (fazendo  $m = \phi(n)$ ) — o menor inteiro positivo  $m$  tal que  $a^m \equiv 1 \pmod{n}$  é a **ordem** de  $a \pmod{n}$

**Raiz Primitiva:** Sejam  $a$  e  $n$  inteiros tais que  $(a, n) = 1$ ; se a ordem de  $a \pmod{n}$  é  $\phi(n)$ , então  $a$  é uma **raiz primitiva** de  $n$

• Só  $2, 4, p^n$  e  $2p^n$  (com  $p$  primo ímpar) têm raízes primitivas

• Sejam  $p$  um inteiro primo e  $a$  uma raiz primitiva de  $p$ . Então, as potências de  $a$  (desde 1 até  $(p-1)$ ) não são todas distintas, pelo que "cobrem" todos os inteiros desde 1 até  $(p-1)$ .

$$\downarrow \quad \forall b, \exists r : b \equiv r \pmod{p} \wedge 0 \leq r \leq p-1$$

$$\downarrow \quad \forall b, \exists i : b \equiv a^i \pmod{p} \wedge 0 \leq i \leq p-1$$

**Índice:** Sejam  $p$  um primo e  $a$  uma raiz primitiva de  $p$ ; se  $b \equiv a^i \pmod{p}$ , então  $i$  é o **índice** de  $b$  em relação a  $a \pmod{p}$

$$\text{ind}_{a,p}(1) = 0, \text{ porque } a^0 \equiv 1 \pmod{p}$$

$$\text{ind}_{a,p}(a) = 1, \text{ porque } a^1 \equiv a \pmod{p}$$

$$xy \equiv a^{\text{ind}_{a,p}(xy)} \equiv a^{\text{ind}_{a,p}(x)} a^{\text{ind}_{a,p}(y)} \equiv a^{\text{ind}_{a,p}(x) + \text{ind}_{a,p}(y)} \pmod{p}$$

**ORDEM:**  $\text{ord}_{a,n} = m : a^m \equiv 1 \pmod{n}$

**RAIZ PRIMITIVA:**  $a : \text{ord}_{a,n} = \phi(n) \Leftrightarrow a^{\phi(n)} \equiv 1 \pmod{n}$

**ÍNDICE:**  $\text{ind}_{a,p}(b) = i : a^i \equiv b \pmod{p} \Leftrightarrow \text{ind}_{a,p}(b) = \log_a(b) \pmod{p}$

# Diffie - Hellman

1. Alice e Bob têm de concordar num grupo multiplicativo público  $\mathbb{Z}_p^*$
  1. Alice e Bob têm de concordar num primo  $p$  e numa base  $g \in \mathbb{Z}_p^*$
  2. Alice e Bob geram chaves privadas aleatórias
  2. Alice e Bob geram os inteiros  $a \in \mathbb{Z}_p^*$  e  $b \in \mathbb{Z}_p^*$
  3. Alice e Bob geram chaves públicas
  3. Alice e Bob geram  $A = g^a \pmod p$  e  $B = g^b \pmod p$
  4. Alice e Bob trocam as respectivas chaves públicas (através de um canal público desprotegido) para concordarem num valor secreto sem terem de o transmitir
  4.  $K = A^b = (g^a)^b = g^{ab} = (g^b)^a = B^a$
- $K = g^{ab}$  é o segredo partilhado que é passado a uma função de derivação de chave para gerar uma ou mais chaves simétricas
- Segurança:
1.  $g$  deve ser uma raiz primitiva de  $p$  — gerar todos os valores possíveis em  $\mathbb{Z}_p^*$
  2.  $(p-1)/2$  deve ser primo — evitar sub-grupos dentro do grupo

$$K = A^b = (g^a)^b = g^{ab} = g^{ba} = (g^b)^a = B^a$$

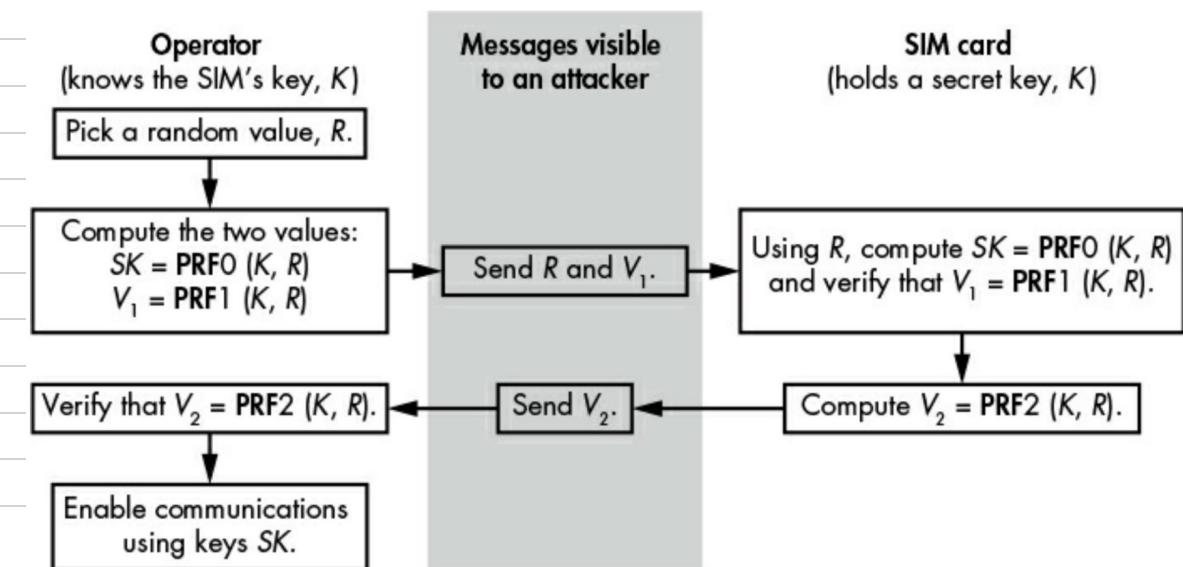
**Problema Computacional de Diffie-Hellman:** computar o segredo partilhado  $g^{ab}$  dados apenas os valores públicos  $g^a$  e  $g^b$  e nenhum dos valores secretos  $a$  ou  $b$  — é possível obter o segredo  $g^{ab}$  só com a informação pública?

- O DLP é pelo menos tão difícil como CDH
- DH entrega o mesmo nível de segurança que RSA para um dado módulo

**Problema Decisional de Diffie-Hellman:** dados  $g^a$  e  $g^b$  e um valor que é  $g^{ab}$  ou  $g^c$  para um  $c$  aleatório, é possível determinar que  $K$  foi escolhido?

Assunção Decisional de Diffie-Hellman: nenhum atacante consegue resolver DDH

- Se DDH é difícil, então CDH também é difícil
- Se é possível resolver CDH, então também é possível resolver DDH
- DDH é fundamentalmente mais difícil do que CDH
- A dificuldade de DDH é uma assunção criptográfica
- Para parâmetros bem escolhidos, confia-se que CDH e DDH são difíceis



## MODELOS DE ATAQUE

"Eavesdropper": observa as mensagens trocadas entre as duas partes legítimas e executaem o protocolo de acordo de chave e pode registar, modificar, apagar e injetar mensagens - o protocolo não deve libertar informação sobre o regredo

"Data Leak": adquire a chave de sessão e todos os regredos temporários de uma ou mais execuções do protocolo, mas não os regredos de longo-prazo

"Breach/Corruption": aprende o regredo de longo-prazo de uma ou mais partes  
↳ a segurança não é atingível porque o atacante pode impersonar uma ou mais partes, mas não deve ser capaz de recuperar regredos anteriores à recuperação da chave

## OBJETIVOS DE SEGURANÇA

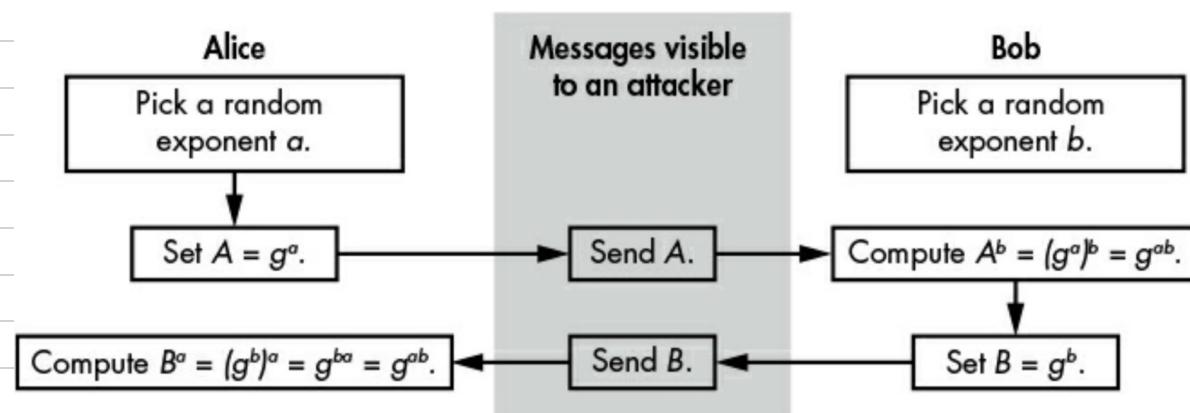
Autenticação: cada parte deve ser capaz de autenticar a outra - autenticação mútua

Controlo da Chave: nenhuma parte deve ser capaz de escolher o regredo partilhado ou influenciá-lo para um subconjunto específico

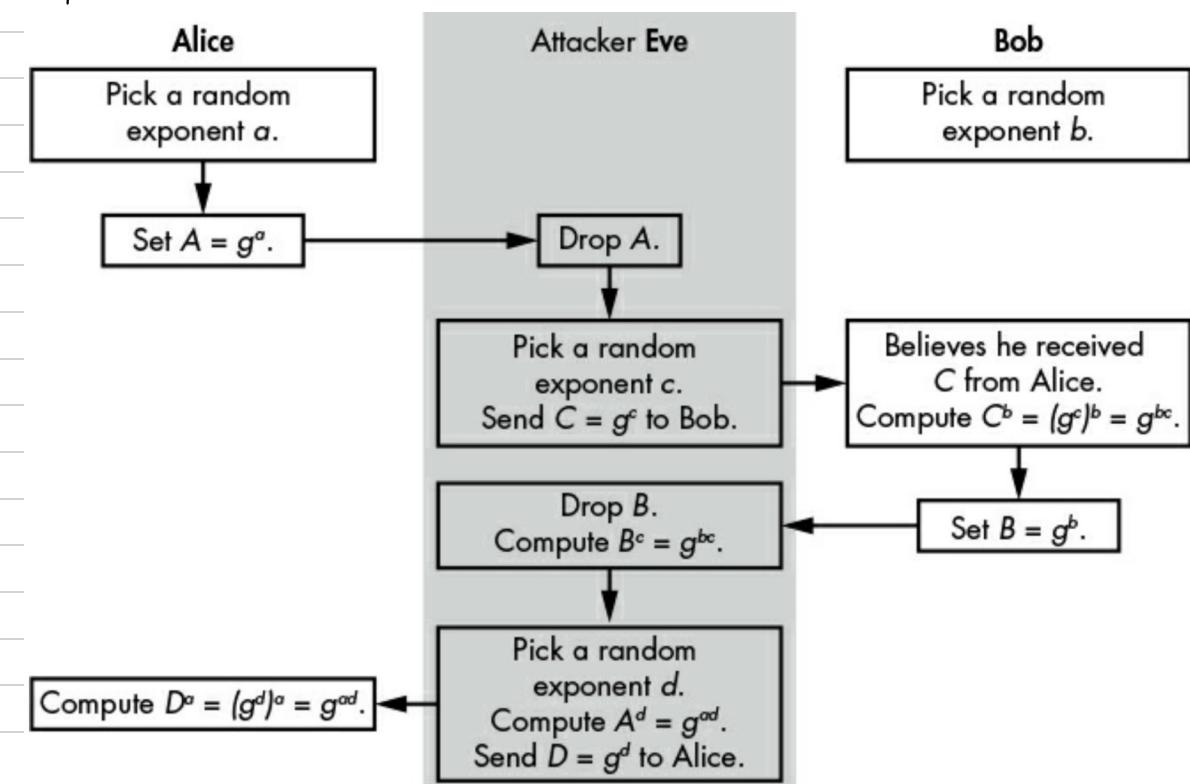
Secretismo para a Frente: mesmo que todos os regredos de longo-prazo sejam expostos, regredos partilhados de execuções anteriores do protocolo não podem ser computados, mesmo que um atacante registre todas as execuções anteriores ou seja capaz de injetar ou modificar mensagens de execuções anteriores

Resistência a "Key-Compromise Impersonation (KCI)": prevenir que um atacante comprometa uma chave de longo-prazo de uma parte e seja capaz de a usar para impersonar outra parte

Diffie-Hellman Anônimo: os participantes não têm nenhuma identidade que possa ser verificada pela outra parte e nenhuma parte tem uma chave de longo-prazo



Ataque: Man-in-the-Middle



Diffie-Hellman Autenticado: uma chave pública e uma chave privada para cada parte

Segurança:

1. "eavesdroppers" — não obtêm nenhuma informação sobre o segredo partilhado
2. secretismo para a frente — os expoentes efémeros não são recuperáveis
3. controlo do segredo —  $g^b$  é imprevisível desde que  $a \neq 0$  e  $b \neq 0$

Insegurança:

1. "data leak(s)" — registar  $X$  e  $\gamma g^X$  e repeti-los, imitando Xavier

SOLUÇÃO

→ integrar as chaves de longo-prazo na computação do segredo partilhado

Menezes - Oett - Vanstone (MOV): mais seguro e com melhor desempenho do que DH

• só duas mensagens pequenas, independentes e de ordem arbitrária

• nem necessidade de exponenciais/mensagens de assinatura/verificação

$$Z = g^y : (B Y^B)^{a+kA} = (g^b g^B)^{a+kA} = \boxed{g^{(b+yB)(a+kA)}} = (g^a g^{kA})^{b+yB} = (A X^A)^{b+yB}$$

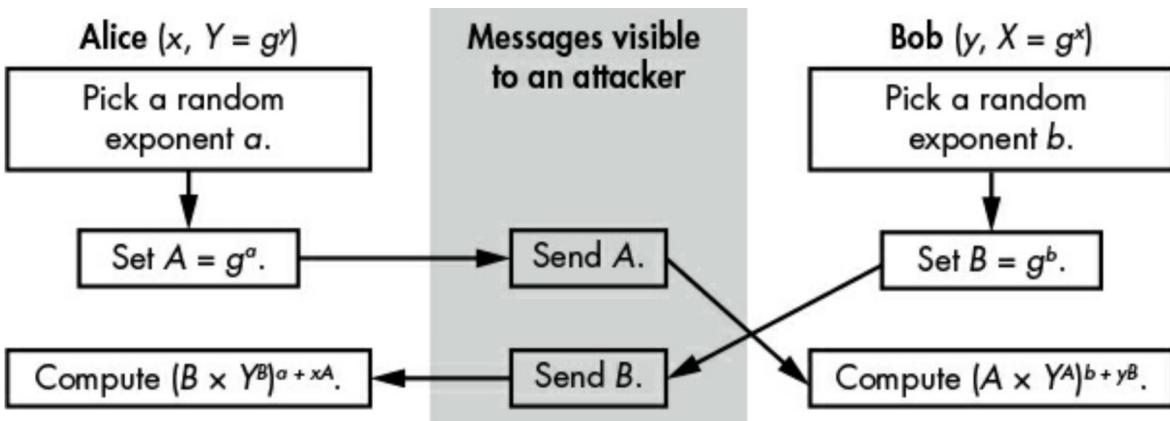
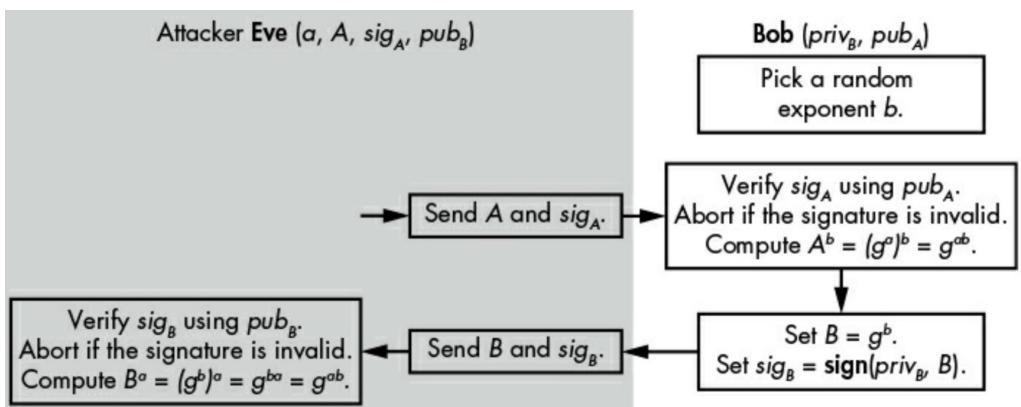
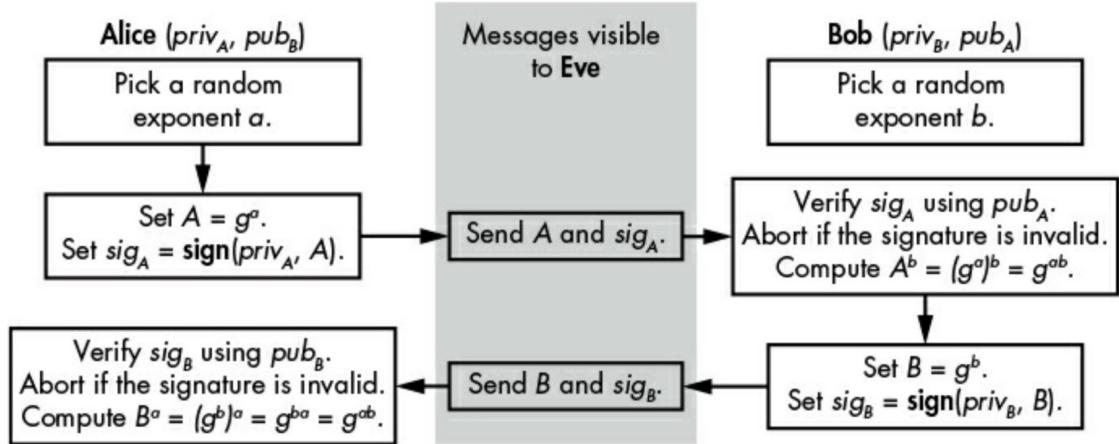
Segurança: "breach" — se a chave privada for comprometida, os segredos partilhados estão seguros porque a sua computação também envolve a chave efêmera privada

Insegurança: secretismo para a frente — se Eve interceptar  $A = g^a$  e substituir por  $A' = g^{a'}$  e se depois comprometer a chave privada  $x$ , consegue determinar a chave computada pelo Bob na etapa SOLUÇÃO de confirmação da chave

⚠  $g^b$  não é aleatório porque pertence a um grupo → deve ser "hashed"

⚠ implementações TLS suportam DH anônimo

⚠ OpenSSL usava primos inseguros



# Curvas Elípticas

ECC é mais seguro e eficiente do que RSA e DH

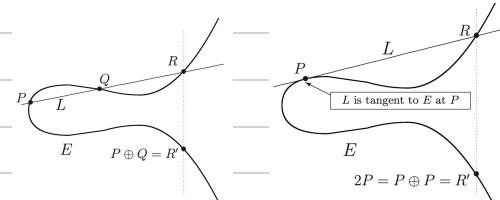
Curva Elíptica:  $E: Y^2 = X^3 + AX + B$

Adição de Pontos:  $P \oplus Q = R'$

$(a, b) \oplus (a, -b) = P + P' = O$  - infinito

$$1. P + O = O + P = P$$

$$2. P + (-P) = O$$



$$3. (P + Q) + R = P + (Q + R)$$

$$4. P + O = O + P$$

Uma Curva Elíptica é um Grupo Abeliano!

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & \text{se } P_1 \neq P_2 \\ \frac{3x_1^2 + A}{2y_1}, & \text{se } P_1 = P_2 \end{cases}$$

$$P_3: \begin{cases} x_3 = \lambda^2 - x_1 - x_2 \\ y_3 = \lambda(x_1 - x_3) - y_1 \end{cases}$$

Multiplicação por Inteiro:  $nP = P + \underbrace{\dots + P}_n$  - algoritmo de exponenciação rápida

Considerem-se as curvas elípticas cujos pontos têm coordenadas num campo finito  $\mathbb{F}_p$  e que:

$$(1) \quad A, B \in \mathbb{Z}_p: 4A^3 + 27B^2 \neq 0$$

$$(2) \quad E: Y^2 = X^3 + AX + B$$

$$(3) \quad E(\mathbb{F}_p) = \{(x, y) \in \mathbb{F}_p \mid y^2 = x^3 + Ax + B\} \cup \{O\}$$

Quantos pontos tem uma curva elíptica?

Teorema de Hasse:  $|E(\mathbb{F}_p)| = p+1-t_p, \quad |t_p| \leq 2\sqrt{p}$

$$\text{DLP: } \mathbb{Z}_p: g^n = l$$

$$\text{DLP: } x = g^{y \pmod p}$$

$$\text{ECDLP: } E(\mathbb{Z}_p): nG = L$$

$$\text{ECDLP: } Q = kP$$

• ECDLP permite números menores e mais segurança do que DLP

Como resolver ECDLP?

Encontrar uma colisão  $c_1P + d_1Q = c_2P + d_2Q \wedge Q = kP$

$$\Leftrightarrow c_1P + d_1kP = c_2P + d_2kP$$

$$\Leftrightarrow (c_1 + d_1k)P = (c_2 + d_2k)P$$

$$\Leftrightarrow c_1 + d_1k = c_2 + d_2k$$

$$\Leftrightarrow d_2k - d_1k = c_1 - c_2$$

$$\Leftrightarrow k = \frac{c_1 - c_2}{d_2 - d_1}$$

• O número de pontos na curva não é um segredo!

Diffie-Hellman em Curvas Elípticas: ponto  $G$  fixo

A:  $P_A = d_A G$

B:  $P_B = d_B G$

K:  $d_A P_B = d_B P_A = d_A d_B G$

Elliptic Curve Digital Signature Algorithm (ECDSA):  $G, E$  e  $n$  fixos

Chave Privada:  $d$

Chave Pública:  $P = dG$

Assinar:

1.  $h \leftarrow H(m), 0 \leq h \leq n-1$
2.  $k \leftarrow \{1, \dots, n-1\}$
3.  $\langle x, y \rangle \leftarrow kG$
4.  $\langle r, s \rangle \leftarrow \langle x, (h+rd)/k \rangle \pmod n$

Verificar:

1.  $w \leftarrow s^{-1} = 1/s = k/(h+rd) \pmod n$
2.  $m \leftarrow wh = hk/(h+rd) \pmod n$
3.  $v \leftarrow wr = rk/(h+rd)$
4.  $Q = mG + vP = uG + vdG = \frac{u+v}{k}G$
5.  $x_Q \stackrel{?}{=} r$  ?

$m+vd = hk/(h+rd) + drk/(h+rd) = (hk+d rk)/(h+rd) = k(h+rd)/(h+rd) = k$

• A assinatura com ECC é mais curta e mais rápida do que com RSA

EIDSA: mais rápido, mais simples e determinístico — leva inverso modular

"Ingredientes":

1. campo finito  $\mathbb{F}_q$ , sendo  $q = p^k$  ( $p$  primo ímpar)
2. curva elíptica  $E$  em  $\mathbb{F}_q$ , sendo  $|E(\mathbb{F}_q)| = 2^c p$  ( $p$  primo grande)
3. ponto  $G$  em  $E(\mathbb{F}_q)$  com ordem  $p$
4. função de hash criptográfica com output de  $2b$  bits, sendo  $2^{b+1} > q$

Chave Privada:  $k_B : k_B \parallel s = H(s) \wedge |k_B| = |s| = b$

Chave Pública:  $K_B = k_B G$

Assinar:

$$1. r = H(s \parallel m)$$

$$2. R = r G$$

$$3. S \equiv r + H(R \parallel K_B \parallel m) k_B \pmod{p}$$

$$\hookrightarrow 2^c S G = 2^c (r + H(R \parallel K_B \parallel m) k_B) G \\ = 2^c \underbrace{r G}_{R} + 2^c H(R \parallel K_B \parallel m) \underbrace{k_B G}_{K_B}$$

Transmitir:  $S \parallel R \parallel m$

Verificar:  $2^c S G = 2^c R + 2^c H(R \parallel K_B \parallel m) K_B$

EIDSA: instância específica de EIDSA — Curve 25519 e SHA-512

A encriptação com curvas elípticas é incomum porque o tamanho da mensagem é muito restrito (2,5% de RSA)

Integrated Encryption Scheme (IES): encripta ao gerar um par de chaves DH, combinando a chave privada com a chave pública do receptor, derivando uma chave simétrica a partir do segredo partilhado e usando uma cifra autenticada

Elliptic-Curve Integrated Encryption Scheme (ECIES): KDF; MAC; E; G; n  
 Chave Privada:  $k_B \in [1, n-1]$  Chave Pública:  $K_B = k_B G$

Cifrar:

1.  $r \leftarrow [1, n-1]$
2.  $R \leftarrow rG$
3.  $P = (P_x, P_y) \leftarrow rK_B$
4.  $S \leftarrow P_x$
5.  $K_E || k_M \leftarrow \text{KDF}(S || S_1)$
6.  $c \leftarrow E_{K_E}(m)$
7.  $d \leftarrow \text{MAC}_{k_M}(c || S_2)$

Transmitir:  $R || c || d$

Decifrar:

1.  $P = (P_x, P_y) \leftarrow k_B R = K_B rG = rK_B G = rK_B$
2.  $S \leftarrow P_x$
3.  $K_E || k_M \leftarrow \text{KDF}(S || S_1)$
4.  $d ? = \text{MAC}_{k_M}(c || S_2)$
5.  $m = E_{K_E}^{-1}(c)$

## CURVAS FAMOSAS

NIST SP 800-90: para gerar uma sequência pseudo-aleatória -  $s_i \in [0, n-1]$

1.  $s_i \leftarrow \pi(s_{i-1}, P)$
2.  $n_i \leftarrow \text{lsb}_{240}(\pi(s_i, Q))$

Como escolher uma curva?

- A ordem do grupo não deve ser um produto de números pequenos
- Tratar a adição  $P+Q$  de forma diferente quando  $P=Q$  pode libertar informação crítica, pelo que é melhor só usar uma única fórmula para adição

Curvas NIST:  $p = 2^{256} - 2^{224} + 2^{192} + 2^{96}$

$$E: y^2 = x^3 - 3x + b$$

Curva 25519:  $p = 2^{255} - 19$

$$E: y^2 = x^3 + 486662x^2 + x$$

- ataques por canais colaterais
- reutilizar  $k$  leva a recuperar a chave privada  $d = (k_{A_1} - h_1)/r = ((h_1 + rd) - h_1)/r$
- deve ser verificado se os pontos  $P$  e  $Q$  pertencem à curva certa

# PKI

- Todas as primitivas criptográficas de chave pública assumem que as chaves públicas são autênticas, senão, os protocolos são vulneráveis a ataques MitM
- PKI fornece a cobertura legal/regulatória necessária

Trusts - Third - Party (TTP)  $\approx$  Certificate Authority (CA)

Como é que o Bob pode verificar que a Alice tem a chave privada associada a uma chave pública?

- O Bob tem um canal autenticado para a TTP
- A Alice prova à CA que tem pp ao avisar um pedido de certificado e a CA devolve  $\Sigma_X$
- A CA fornece/verifica os dados que a Alice quer no certificado ( $ID_A, PK_A, CA, V$ )
- A CA armazena os dados como uma estrutura de dados **Abstract Syntax Notation 1**

certificado = dados da Alice + assinatura da CA  
confiança no certificado < confiança na CA

- A assinatura digital garante que o certificado é autêntico para o Bob
- A CA pode estar "offline" porque o Bob pode obter o certificado através da Alice

Como verificar um Certificado de Chave Pública? ( $ID_A, PK_A, VAL, M-I, \sigma_{CA}$ )

1. identidade correta?
2. dentro do período de validade?
3. metainformação faz sentido?
4. CA confiável para certificar a chave pública? } PKI
5. assinatura da CA no certificado é válida

**Certificado X.509 V.3:** sujeito; emissor; validade; chave pública; informações; série  
Extensões: identificadores da chave; restrições básicas; uso da chave

Infraestrutura de Chave Pública (PKI): conjunto de papéis, políticas, hardware, software e procedimentos necessários para criar, gerir, distribuir, usar, armazenar e revogar certificados digitais

Objetivo: um utilizador certificado pode ser assegurado por uma entidade certificadora confiável que uma chave pública pertence a outra entidade final e pode ser usada para um dado propósito sob direitos e responsabilidades bem definidas para todas as partes

Inicialização: como é que os utilizadores conhecem uma CA e verificam a sua assinatura?  
→ Todas as chaves públicas estão codificadas em certificados X.509 e alguns certificados contêm as chaves públicas de CAs que são usadas para verificar assinaturas

A chave pública da Alice é autenticada se (1) o Bob tem o certificado para a CA que emitiu o certificado da Alice e (2) o Bob confia que a CA verificou os dados no certificado da Alice

Como é que o Bob sabe que pode confiar na CA?

O Bob obtém o certificado diretamente da CA e confia implicitamente nela

Autoridades Registradoras (RA): "front-end" - com contacto direto com as entidades finais, não responsáveis por verificar os dados que vão nos certificados e garantir que a entidade (é a única que) possui a chave privada

Autoridades Certificadoras (CA): "back-end" - infraestrutura onde os certificados são armazenados

Revogação: os certificados podem ter de ser invalidados (enquanto ainda parecem válidos, mas estejam dentro do período de validade) devido a (1) perda da chave privada, (2) fuga de dados ou (3) metadados tornarem-se incorretos

Lista de Revogação de Certificados (CRL): "black-list" de certificados revogados publicada periodicamente (e excepcionalmente) pela CA que os clientes devem verificar

1. Trusted Service Provider Lists (TSL): "whitelist" atualizada de certificados confiáveis
2. Online Certificate Status Protocol (OCSP): serviço confiável, normalmente gerido pela CA, verifica os certificados
3. Certificate Pinning: serviços/navegadores/aplicações transportam suas próprias listas que identificam bons certificados para entidades importantes

Cadeias de Certificados e Hierarquia de CAs: o Bob é inicializado com certificados auto-assinados (sujeito = emissor = CA) por CAs raiz em quem confia implicitamente

- Certificados Auto-Assinados podem ser gerados por qualquer um, mas validá-los implica acreditar que quem tem a chave privada é uma CA que só gera bons certificados
- Se  $CA_A$  assina o certificado de  $CA_B$ , a confiança em  $CA_B$  é menor ou igual à que a confiança em  $CA_A$

Confiança: a CA da Alice é descendente de uma CA raiz confiada pelo Bob

1.  $CA_{atual} = CA_{da\ Alice}$

↑ Se: Bob confia implicitamente no certificado da CA atual ✓

Se: CA atual é subordinada de alguma CA

a. Bob obtém o certificado de CA

b. Bob verifica se o certificado da CA atual é válido em relação a CA

c.  $CA_{atual} = \widehat{CA}$

Senão: X

OID

Política de Certificado: conjunto de regras de operação PKI com direitos e responsabilidades de entidades finais e CAs, que podem ser escritas na lei — acreditação

# Encriptação Homomórfica

Encriptação Totalmente Homomórfica: suporta homomorfismos de anel (2 operações)

Encriptação Parcialmente Homomórfica: suporta uma operação

## ESQUEMA DE PAILLIER

Seja  $n = pq$  o produto de dois primos de igual número de bits e tais que:

$$(p, q-1) = 1 \quad \wedge \quad (p-1, q) = 1$$

Então,  $\mathbb{Z}_n \times \mathbb{Z}_{n^2}^*$  é isomórfico para  $\mathbb{Z}_{n^2}^*$  dado por  $f(a, b) = (1+n)^a b^n \mod n^2$

Um elemento uniforme  $y \in \mathbb{Z}_{n^2}^*$  corresponde a um elemento uniforme  $(a, b) \in \mathbb{Z}_n \times \mathbb{Z}_n^*$

Seja  $y \in \mathbb{Z}_{n^2}^*$  um resíduo índice- $n$  módulo  $n^2$  se  $\exists x \in \mathbb{Z}_{n^2}^* : y = x^n \mod n^2$

$\text{Res}(n^2)$ : conjunto de resíduos índice- $n$  módulo  $n^2$

$\forall x \in \mathbb{Z}_{n^2}^* : x \leftrightarrow (a, b) \Rightarrow x^n \mod n^2 \leftrightarrow (a, b)^n = (na, b^n) \mod n = (0, b^n) \mod n$

$\forall y \in \mathbb{Z}_{n^2}^* : y \leftrightarrow (0, b)$  é um resíduo índice- $n$

PORQUÊ?  $(n, \phi(n)) = 1 \rightarrow \exists d = n^{-1} \mod \phi(n)$

$\rightarrow \forall a \in \mathbb{Z}_n : (a, (b^{d \mod n}))^n = (na, b^{dn}) \mod n = (0, b) \leftrightarrow y$

$\text{Res}(n^2) = \{y \mid b \in \mathbb{Z}_n^* \wedge y \leftrightarrow (0, b)\}$

Há  $n$  raias índice- $n$  para  $\forall y \in \text{Res}(n^2)$  e computar a potência índice- $n$  é uma função  $n$ -para-1

Se  $x \in \mathbb{Z}_{n^2}^*$  é uniforme, então  $(x^n \mod n^2)$  é um elemento uniforme de  $\text{Res}(n^2)$

Problema de Decisão da Residuabilidade Composta: distinguir  $m \in \mathbb{Z}_{n^2}^*$  de  $m \in \text{Res}(n^2)$

**GENMODULUS:** algoritmo de tempo polinomial que, no input  $1^n$ , faz output de  $(n, p, q)$  com  $n = pq$ , donde  $p$  e  $q$  primos de  $\lambda$  bits

O Problema Decisional da Residuosidade Compõsita é difícil relativamente a GENMODULUS se para todos os algoritmos de tempo polinomial probabilístico  $D$  existe uma função negligível  $\mathcal{E}$  tal que

$$\Pr[D(n, (r^m \bmod n^2)) = 1] - \Pr[D(n, r) = 1] \leq \mathcal{E}(n)$$

onde em cada caso as probabilidades não retíadas da experiência em que GENMODULUS( $1^n$ ) faz output de  $(n, p, q)$  e é escolhido um  $r \in \mathbb{Z}_{n^2}^*$  uniforme

**Assunção DCR:** existe um GENMODULUS para o qual o problema é difícil

**Encriptar:**  $m \in \mathbb{Z}_n$  em relação à chave pública  $n$

1. Escolher um resíduo índice- $n$  uniforme  $(0, r)$  — indistingível de  $(r^1, r)$  uniforme
2.  $c \leftrightarrow (m, 1)(0, r) = (m + 0, 1 \cdot r) = (m, r)$  — indistingível de  $c'$   
 $c' \leftrightarrow (m, 1) \cdot (r^1, r) = ((m + r^1) \bmod n, r)$

Como  $m + r \pmod{n}$  é uniformemente distribuído em  $\mathbb{Z}_n$ ,  $c'$  não depende de  $m$

**Encriptação:**  $r \in \mathbb{Z}_n^*$  uniforme  $\rightarrow c = (1+n)^m r^n \pmod{n^2}$ ,  $c \in \mathbb{Z}_{n^2}^*$   
 $c = ((1+n)^m 1^n) (1+n)^0 r^n \pmod{n^2} \leftrightarrow (m, 1) \cdot (0, r) \Leftrightarrow c \leftrightarrow (m, r)$

**Desencriptação:**

$$(1) \quad \hat{c} = c^{\phi(n)} \pmod{n^2}$$

$$\hat{c} \leftrightarrow (m, r)^{\phi(n)} = (m^{\phi(n)}, r^{\phi(n)}) \pmod{n^2} = (m^{\phi(n)}, 1) \pmod{n}$$

$$(2) \quad \hat{m} = (\hat{c} - 1)/n$$

$$(3) \quad \hat{m} = m^{\phi(n)-1} \pmod{n}$$

$$(1+n)^a \equiv 1 + an \pmod{n^2}$$

Encriptação Homomórfica: permite certas computações nos dados encriptados, que resultam num criptograma com o resultado encriptado

Encriptação de Paillier: a operação é a adição modular

$$E(m_1) \times E(m_2) \pmod{n^2} = E(m_1 + m_2) \pmod{n}$$

$$\underbrace{(1+n)^{m_1}}_{\pi_1^n} \underbrace{(1+n)^{m_2}}_{\pi_2^n} \equiv (1+n)^{(m_1+m_2 \pmod{n})} (\pi_1 \pi_2)^n \pmod{n^2}$$

Votação:

$$1. n$$

$$2. c_i = (1+n)^{v_i} \pi_i^n \pmod{n^2}$$

$$3. c_T = \prod c_i \pmod{n^2}$$

$$4. v_T = \sum v_i \pmod{n}$$

## PROVAS

Seja  $n = pq$  o produto de dois primos de igual comprimento

$$1. (n, \phi(n)) = 1, \quad \phi(n) = \left| \{k \mid 1 \leq k < n \wedge (n, k) = 1\} \right| = (p-1)(q-1)$$

$$\phi(p^a) = p^a - p^{a-1}$$

$$2. \forall a, a > 0 \Rightarrow (1+n)^a \equiv 1 + a \pmod{n^2} \Leftrightarrow \text{ord}(1+n) \text{ em } \mathbb{Z}_{n^2}^* \text{ é } n$$

3. O grupo  $\mathbb{Z}_n \times \mathbb{Z}_n^*$  é isomórfico para o grupo  $\mathbb{Z}_{n^2}^*$  com o isomorfismo  
 $f: \mathbb{Z}_n \times \mathbb{Z}_n^* \rightarrow \mathbb{Z}_{n^2}^*$

$$(a, b) \rightarrow (1+n)^a b^n \pmod{n^2}$$

$$|\mathbb{Z}_{n^2}^*| = \phi(n^2) = \phi(p^2 q^2) = p(p-1)q(q-1) = pq(p-1)(q-1) = |\mathbb{Z}_n \times \mathbb{Z}_n^*|$$

$$(1+n)^{a_1-a_2} (b_1/b_2)^n \equiv 1 \pmod{n^2} \Rightarrow (1+n)^{(a_1-a_2)\phi(n)} (b_1/b_2)^n \equiv 1 \pmod{q^2}$$

$$\Rightarrow (1+n)^{(a_1-a_2)\phi(n)} \equiv 1 \pmod{n^2} \Rightarrow (a_1-a_2)\phi(n) \equiv 0 \pmod{n} \Rightarrow n \mid (a_1-a_2) \Rightarrow \frac{a_1}{b_1} = \frac{a_2}{b_2}$$

# Computação Clássica

↓  
Modelo de Turing/Von Neumann: modelo de computação em que só é executada uma instrução de cada vez

• A velocidade de computação está diretamente relacionada com a frequência, que é limitada pela energia necessária para mudar o estado de um bit

Computação Reversível: as limitações físicas resultantes da dissipação de calor podem ser evitadas para quase todas as operações, tornando-as invertíveis  
↓ EXCEÇÃO

DELETE

- (1) Reversibilidade Lógica: o input e o output podem ser obtidos um do outro  
(2) Reversibilidade Física: o dispositivo pode funcionar no sentido oposto, pelo que, pela Segunda Lei da Termodinâmica, não limpa energia

• Num computador quântico, os programas não avaliados pela evolução unitária de um input dado pelo estado do sistema. Como todos os operadores unitários são invertíveis, é sempre possível "descomputar" a computação realizada

Conjunto Universal de Portas Lógicas: pode expressar qualquer computação {↑/↓}  
↓ MÁS INSUFICIENTE

Para um computador:  $A \xrightarrow{\text{FANOUT}} A$   
REVERSÍVEL

$A \xrightarrow[\text{PRIMITIVO}]{\text{ERASE}} X$   
IRREVERSÍVEL

Como evitar ERASE primitivo?  $f: a \rightarrow (a, f(a))$

$f: a \rightarrow (a, j(a), f(a))$

FANOUT:  $(a, j(a), f(a)) \rightarrow (a, j(a), f(a), f(a))$

$f^*: (a, j(a), f(a), f(a)) \rightarrow (a, f(a))$

$\begin{array}{c} A \\ B \\ C \end{array} \xrightarrow{\text{TOFFOLI}} \begin{array}{c} A \\ B \oplus (A \wedge C) \\ C \end{array}$

UNIVERSAL INVERTÍVEL  
 $(B \oplus (A \wedge C)) \oplus (A \wedge C) = B$

# Computação Quântica

Sistema Quântico: partícula com dois spin  $1/2$   $| \downarrow \rangle | 0 \rangle (| 1 \rangle)$   $| \downarrow \rangle | 0 \rangle (| 1 \rangle)$

Estado da Partícula:  $\phi = \alpha | 0 \rangle + \beta | 1 \rangle$ ,  $\alpha, \beta \in \mathbb{C}$  SUPERPOSIÇÃO LINEAR

$|\alpha|^2$  e  $|\beta|^2$  representam as probabilidades de encontrar a partícula nesse estado

Para  $k$  partículas de spin  $1/2$ :  $2^k$  estados — geradores do espaço de Hilbert

$$U_0 = \begin{pmatrix} e^{i(\delta + \frac{\pi}{2} + \frac{\gamma}{2})} \cos(\theta/2) & e^{i(\delta + \frac{\pi}{2} - \frac{\gamma}{2})} \sin(\theta/2) \\ -e^{i(\delta - \frac{\pi}{2} + \frac{\gamma}{2})} \sin(\theta/2) & e^{i(\delta - \frac{\pi}{2} - \frac{\gamma}{2})} \cos(\theta/2) \end{pmatrix} \xrightarrow{\delta=\alpha=\gamma=0} \begin{pmatrix} \cos(\theta/2) & \sin(\theta/2) \\ -\sin(\theta/2) & \cos(\theta/2) \end{pmatrix}$$

$$U_{\pi} = \begin{pmatrix} \cos(\pi/2) & \sin(\pi/2) \\ -\sin(\pi/2) & \cos(\pi/2) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

SWAP  $\left\{ \begin{array}{l} U_{\pi}|0\rangle = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \end{pmatrix} = -|1\rangle \\ U_{\pi}|1\rangle = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle \end{array} \right.$

Operador de Hadamard: transforma uma partícula com  $| \downarrow \rangle$  juntamente com sobreposição eiquiprovável de  $| \downarrow \rangle$  e  $| \uparrow \rangle$

$$\rightarrow U_{\frac{\pi}{2}}|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

Para  $k$  partículas com  $|0\rangle$  inicial:  $|0\rangle \rightarrow \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle \xrightarrow{q=2^K} \frac{1}{\sqrt{2^K}} \sum_{a=0}^{2^K-1} |a\rangle$

- A superposição de todas as strings binárias de comprimento  $k$
- A sobreposição de todos os inteiros desde  $0$  até  $2^K - 1$

Paralelismo Quântico:  $|a; 0\rangle \rightarrow |a; f(a)\rangle$   $\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle \rightarrow \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a; f(a)\rangle$

Calcula o valor de  $f(a)$  em paralelo para todos os valores de  $a$

# XOR

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

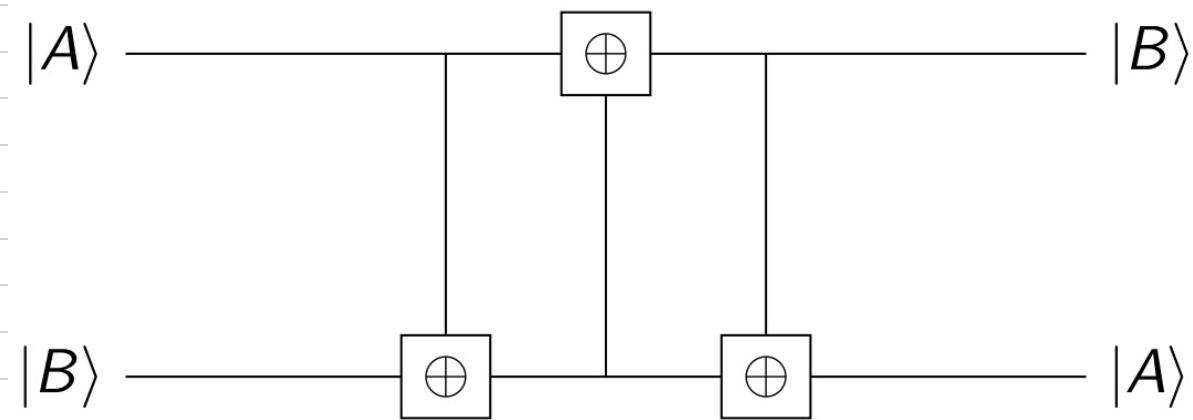
$$|01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

$$|10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$|11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$U_{\text{xor}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

A porta XOR permite mover informação



A porta XOR e portas de um bit não suficientes para expressar qualquer transformação num conjunto finito de bits

# Criptanálise Quântica

Considerese a sequência  $f(0), f(1), \dots, f(q-1)$  com  $q = 2^k$

Como determinar o período de  $f$ ? PARALELISMO QUÂNTICO

1. Conjunto de partículas com  $|1\rangle$  divididas em dois grupos:  $|0;0\rangle = |\downarrow \dots \downarrow; \downarrow \dots \downarrow\rangle$
2. Aplicar Hadamard ( $U_{H_2}$ ) a cada qubit do primeiro registo, obtendo a superposição de todos os inteiros possíveis desde 0 até  $2^k - 1$   $\rightarrow \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a;0\rangle$
3. Decompor o cálculo de  $f(a)$  num conjunto de operadores de 1 ou 2 bits, fazendo corresponder  $|a;0\rangle$  em  $|a;f(a)\rangle$
4. Aplicar esta seqüência de operadores unitários à superposição de estados, obtendo  $\rightarrow \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a;f(a)\rangle$  de uma só vez para todos os valores de  $a$
5. Considerar a versão quântica da transformação discreta de Fourier, que é trivialmente unitária:  $|a\rangle \rightarrow \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} e^{\frac{2\pi i a c}{q}} |c\rangle$
6. Aplicar esta transformação à superposição de estados armazenados nos registo de qubits, obtendo  $\rightarrow \frac{1}{q} \sum_{a=0}^{q-1} \sum_{c=0}^{q-1} e^{\frac{2\pi i a c}{q}} |c;f(a)\rangle$
7. Medir os spins das partículas que constituem o primeiro registo — o somatório sobre a vai conferir a interferência construtiva dos coeficientes  $e^{\frac{2\pi i a c}{q}}$  se  $\frac{c}{q}$  é um múltiplo de  $1/q$
8. Para obter o período, basta repetir o cálculo quântico  $\log(\log(\frac{1}{\epsilon}))$  vezes com uma alta probabilidade de um dos múltiplos ser coprimo com  $q$

Algoritmo de Fatoração de Shor: como fatorar  $n$ ?

$O((\log n)^3)$

1. Escolher  $x$ :  $\gcd(x, n) = 1$
2. Considerar a sequência  $x^i \pmod{n}$
3. Calcular o período de  $f$ :  $f(a) = f(a+r)$
4. Se  $r$  for ímpar, retorne
5. Se  $r$  for par:

$$(x^{\frac{r}{2}})^2 \equiv 1 \pmod{n}$$

$$(x^{\frac{r}{2}} + 1)(x^{\frac{r}{2}} - 1) \equiv 0 \pmod{n}$$

$$(x^{\frac{r}{2}} + 1)(x^{\frac{r}{2}} - 1) \equiv 0 \pmod{n}$$

$\therefore (x^{\frac{r}{2}} + 1)$  ou  $(x^{\frac{r}{2}} - 1)$  têm um fator comum com  $n \rightarrow$  Algoritmo de Euclides

Protocolo de Bennett & Brassard: transmitir informações através da orientação da onda de um fotão

1. Alice gera duas sequências pseudo-aleatórias de bits  $r_1$  e  $r_2$
  2. Alice interpreta  $r_1$  como a sequência de bases a usar
  3. Alice codifica  $r_2$  usando as bases obtidas por  $r_1$
  4. Alice envia a mensagem codificada a Bob
  5. Bob gera uma sequência pseudo-aleatória de bits  $r'$
  6. Bob tenta ler os fotões com a sequência de bases dada por  $r'$
  7. Bob vai acertar em cerca de 50% dos bits
  8. Alice envia  $r_1$  por um canal aberto
  9. Bob vai saber que bits de  $r_1$  acertou
  10. Bob envia  $r'$  a Alice pelo mesmo canal
  11. Alice vai saber que bits foram lidos com sucesso
- A sequência de bits lidos com sucesso é conhecida pelos dois intervenientes e pode ser usada para determinar as bases de polarização em comunicações futuras.
- Eve não consegue adivinhar  $r_1$  nem  $r'$ , por isso não consegue adivinhar que bits de  $r_2$  correspondem às coincidências das duas primeiras aleatoriedades, pelo que não poderá ler a informação futuramente codificada.

# Criptografia Pós-Quântica

Criptografia Pós-Quântica: primitivas criptográficas que não são ameaçadas pelo speedup possível de alcançar devido ao paralelismo quântico

## CRIPTOGRAFIA BASEADA EM MALHAS

Espaço Vetorial:  $V \subseteq \mathbb{R}^m \wedge \forall v_1, v_2 \in V: \forall \alpha_1, \alpha_2 \in \mathbb{R}: \alpha_1 v_1 + \alpha_2 v_2 \in V$

Combinação Linear:  $w = \alpha_1 v_1 + \dots + \alpha_n v_n, v_1, \dots, v_n \in V, \alpha_1, \dots, \alpha_n \in \mathbb{R}$

Espaço Gerado:  $\left\{ \alpha_1 v_1 + \dots + \alpha_n v_n \mid \alpha_1, \dots, \alpha_n \in \mathbb{R} \right\}$

Independência Linear:  $\alpha_1 v_1 + \dots + \alpha_n v_n = 0 \Rightarrow \alpha_1, \dots, \alpha_n = 0$

Base: conjunto de vetores linearmente independentes que geram um espaço vetorial

Dimensão: número de elementos da base

1.  $V$  tem uma base
2. Quaisquer bases de  $V$  têm a mesma dimensão
3. Um conjunto de vetores escritos como uma combinação linear de uma base é uma base se e só se o determinante da matriz correspondente não é nulo

Comprimento/Norma:  $\|v\| = \sqrt{x_1^2 + \dots + x_m^2}$

Produto Escalar:  $v \cdot w = (x_1, \dots, x_m) \cdot (y_1, \dots, y_m) = x_1 y_1 + \dots + x_m y_m = \|v\| \|w\| \cos \theta$

Ortogonalidade:  $v \cdot w = 0$

Desigualdade de Cauchy-Schwarz:  $\|v \cdot w\| \leq \|v\| \|w\|$

Base Ortogonal:  $\forall i, j: i \neq j \Rightarrow v_i \cdot v_j = 0$

Base Ortonormal:  $\forall i, j: i \neq j \Rightarrow v_i \cdot v_j = 0 \wedge \|v_i\| = 1$

Algoritmo de Gram-Schmidt: cria uma base ortogonal  $v_1^*, \dots, v_n^*$  para  $V$

Malha:  $L = \{a_1 v_1 + \dots + a_n v_n \mid a_1, \dots, a_n \in \mathbb{Z}\}$

Malha Inteira: os vetores têm coordenadas inteiros — subgrupo de  $\mathbb{R}^m$

• Quaisquer bases de  $L$  estão relacionadas por uma matriz de coeficientes inteiros e determinante  $\pm 1$

Subgrupo Aditivo: fechado para adição e subtração

Subgrupo Aditivo Discreto:  $\exists \varepsilon > 0 : L \cap \{w \in \mathbb{R}^m \mid \|v - w\| < \varepsilon\} = \{v\}$

• Um subconjunto aditivo de  $\mathbb{R}^m$  é uma malha se e só se é um subgrupo aditivo discreto

Domínio Fundamental:  $F(v_1, \dots, v_n) = \{t_1 v_1 + \dots + t_n v_n \mid 0 \leq t_i \leq 1\}$

•  $\forall w \in \mathbb{R} : w = t + v, t \in F \wedge v \in L$  únicos //

Determinante de um Domínio Fundamental: volume  $n$ -dimensional de  $F$

Desigualdade de Hadamard:  $\det L = \text{Vol}(F) \leq \|v_1\| \dots \|v_n\|$

$$F = F(v_1, \dots, v_n) = \begin{pmatrix} v_{11} & \dots & v_{1n} \\ \dots & \dots & \dots \\ v_{n1} & \dots & v_{nn} \end{pmatrix}$$

$$\text{Vol}(F(v_1, \dots, v_n)) = |\det F|$$

• Quaisquer domínios fundamentais de  $L$  têm o mesmo volume

•  $\det L$  é invariante de  $L$ , independente do domínio fundamental particular usado

Problema do Vetor Mais Curto (SVP): encontrar o vetor mais curto não nulo numa malha  $L$  — encontrar  $v \in L \wedge v \neq \vec{0}$  que minimiza a norma euclidiana  $\|v\|$

Problema do Vetor Mais Perto (CVP): dado um vetor  $w \in \mathbb{R}^n$  que não está em  $L$ , encontrar o vetor  $v \in L$  que está mais perto de  $w$  — encontrar  $v \in L$  que minimiza a norma euclidiana  $\|w - v\|$

Teorema de Hermite:  $\forall L, \exists v \in L: v \neq \vec{0} \wedge \|v\| \leq \sqrt{n} (\det L)^{\frac{1}{n}}$

Constante de Hermite: menor valor  $\gamma_n$  tal que  $\|v\|^2 \leq \gamma_n (\det L)^{\frac{2}{n}}$

Rácio de Hadamard:  $H = \left( \frac{\det L}{\|v_1\| \dots \|v_m\|} \right)^{\frac{1}{n}}$

Bola Fechada:  $B_R(a) = \{x \in \mathbb{R}^n \mid \|x-a\| \leq R\}$

Limitado:  $\exists R: S \subseteq B_R(0) = \{x \in \mathbb{R}^n \mid \|x\| \leq R\}$

Simétrico:  $\forall a \in S: -a \in S$

Convexo:  $\forall a, b \in S: \overrightarrow{ab} \in S$

Fechado:  $a \in \mathbb{R}^n: \forall B_R(a) \cap S \neq \emptyset \rightarrow a \in S$

$S \subseteq \mathbb{R}^n$

Teorema de Minkowski:

(1)  $S$  simétrico convexo:  $\text{Vol}(S) > 2^n \det L$

(2)  $S$  simétrico convexo fechado:  $\text{Vol}(S) \geq 2^n \det L$

$S$  contém um vetor malha não nulo

Algoritmo do Vértice Mais Próximo de Babai: se os vetores na base de  $L$  não suficientemente ortogonais uns aos outros, então resolve CVP

1.  $w = t_1 v_1 + \dots + t_n v_n, t_1, \dots, t_n \in \mathbb{R}$

2.  $a_i \leftarrow [t_i]$

3.  $a_1 v_1 + \dots + a_n v_n$

# Sistema Criptográfico GGH

→ Problema do Vetor Mais Próximo

Geração de Chave:

1. Alice escolhe uma "base boa"  $v_1, \dots, v_n$  e uma matriz inteira  $U$ :  $\det U = \pm 1$
2. Alice computa uma "base má"  $w_1, \dots, w_n$  resultante das linhas de  $W = Uv$

Encriptação:

1. Bob tem uma mensagem pequena  $m$  como um vetor com coordenadas  $(x_i)_i$ .
1. Bob escolhe um vetor pequeno aleatório  $r$
2. Bob usa a chave pública para computar  $C = x_1 w_1 + \dots + x_n w_n + r$

Desencriptação:

1. Alice computa  $v^{-1}$
2. Alice obtém o vetor encriptado na sua base, computando  $m' = C v^{-1}$
3. Alice aplica o Algoritmo de Babai para recuperar  $m$ 
  - a.  $z = m'$
  - b.  $m = z W^{-1}$

A Eve não consegue obter a mensagem original porque, usando o Algoritmo de Babai, se os vetores da base forem altamente não-ortogonais, então o vetor retornado pelo algoritmo está longe do vetor mais próximo.

# Criptografia Simétrica

Segurança Perfeita: decifrar sem chave é tão difícil como adquirir a mensagem

Segurança Semântica: incapacidade de distinguir o texto cifrado

Princípio de Kerckhoff: o único segredo num sistema criptográfico é a chave

Gerador Aleatório: amostra  $l > 1$  bits de uma fonte de entropia

Gerador Pseudo-Aleatório: algoritmo determinista que recebe uma semente  $s$  e gera bits aleatórios

Resistência a Backtracking: a aleatoriedade passada não deve ser comprometida

Resistência a Previsões: o output futuro deve parecer sempre aleatório (entropia)

Segurança  $(t, \epsilon)$ :  $t$  passos,  $\epsilon$  risco —  $n$  bits:  $(t, t/2^n)$

Cifra por Blocos: permutação pseudo-aleatória — invertível —  $N$  rondas

Redes de Substituição-Permutação: substituição/confusão + permutação/ligação

Redes de Feistel:  $(l, r) \rightarrow (r + R_i(k_i, l); l)$

Cifra por Fluxo:  $(k, m) \rightarrow KS$ ;  $c = KS + m$

LFSR:  $f(x) = 1 + X + \dots + X^n - 2^n - 1$  FCSR: filharm NCSR: não-lineares

FSR:  $R_1 = R_0 \ll 1 \parallel f(R_0)$

Função de Hash: não injetiva; bem distribuída; previsível; determinística; pública

Resistência a Pré-Imagens: encontrar o input que produz um output —  $2^n$

Resistência a Colisões: encontrar  $(m_0, m_1)$  tal que  $h(m_0) = h(m_1)$  —  $2^{n/2}$

Para segurança de  $n$  bits, hashes de  $2n$  bits!

Construção Merkle-Damgard: função de compressão de  $m+n$  para  $n$  bits

Construção Espião: abrover (permute a mensagem) e espremer (busca o output)

MAC:  $t = MAC(k, m)$  — autenticação e integridade

Função de Hash Universal:  $\Pr [UH(k, m_0) = UH(k, m_1)] \leq \epsilon$

1. Encrypt-and-MAC

2. MAC-then-Encrypt

3. Encrypt-then-MAC

# Criptografia Assimétrica

Notação "Big O":  $f(n) = \mathcal{O}(g(n)) \Rightarrow \exists K > 0, \exists n_0, \forall n > n_0: |f(n)| \leq K g(n)$

Notação "Small O":  $f(n) = o(g(n)) \Rightarrow \lim_{n \rightarrow +\infty} f(n)/g(n) = 0$   
 $\text{TIME}(n) \subseteq \text{SPACE}(n)$  P: decisão rápida NP: verificação rápida  
 $P \subseteq NP \subseteq PSPACE \subseteq NSPACE \subseteq EXPTIME$

Pequeno Teorema de Fermat:  $p \nmid a \Rightarrow a^{p-1} \equiv 1 \pmod p$

Teorema de Euler:  $(a, n) = 1 \Rightarrow a^{\phi(n)} \equiv 1 \pmod n \rightarrow m^{\phi(pq)+1} \equiv m \pmod {pq}$

RSA: 1.  $n = pq$  2.  $(e, \phi(n)) = (e, (p-1)(q-1)) = 1 \pmod n$  3.  $d \equiv e^{-1} \pmod {\phi(n)}$

OAEP:  $C = P + M$  PSS: segurança demonstrável FDHS: hash

"Blinding Attack":  $R^e M$  armável  $\rightarrow S = (R^e M)^d = R M^d \rightarrow M^d$

Diffie-Hellman:  $k = A^b = (g^a)^b = g^{ab} = g^{ba} = (g^b)^a = B^a$

CDH: computar  $g^{ab}$  dados  $g^a$  e  $g^b$  - DLP pelo menos tão difícil como CDH

DDH: dados  $g^a$  e  $g^b$  um valor que é  $g^c$  ou  $g^{ab}$  determinar qual foi escolhido

MQV:  $(A X^{aB})^{b+cB} = (g^a g^{cA})^{b+cB} = g^{(a+cA)(b+cB)} = (g^b g^{cB})^{a+cA} = (B Y^{cB})^{a+cA}$

Curvas Elípticas:  $y^2 = x^3 + Ax + B$

ECDSA:  $\langle d, P = dG \rangle$

ECDH:  $d_A P_B = d_B P_A = d_A d_B G$

ECIES:  $\langle k_B, K_B = k_B G \rangle$

PKI: identidade? validade? metainformação? CA confiável? assinatura?

Esquema de Paillier: 1.  $(n, \phi(n)) = 1$  2.  $(1+n)^a \equiv 1+an \pmod{n^2}$  3.  $f(a, b) = (1+n)^a b^n \pmod{n^2}$

Shor:  $f(a) = x^a \pmod n \rightarrow (x^{1/2} + 1)(x^{1/2} - 1) = 0 \pmod n$

B&B: transmite informação através da orientação da onda de um fotão