DeepL

Assine o DeepL Pro para traduzir documentos maiores.
Visite www.DeepL.com/pro para mais informações.

Digestões, controlo da integridade e derivação de chaves

deti universidade de aveiro
departamento de eletrónica, telecomunicações e informática

SIO

João Paulo Barraca

Funções de digestão

Visão geral

- Produzir um **resumo** digital de dados denominado **message digest**
  - Os dados são um texto ou qualquer informação binária

- O **comprimento do** resumo da mensagem **é fixo**
  - independentemente do comprimento do texto
    - Tanto um item de dados de 200 bytes como um item de dados de 200 TB resultarão num resumo com o mesmo comprimento

- O valor do resumo da mensagem **depende fortemente** dos dados

- Duas digestões são normalmente **muito diferentes**
  - Mesmo que os dados originais sejam extremamente semelhantes

Funções de digestão

Propriedades

- Resistência à pré-imagem
  - Dado um resumo, é impossível encontrar um texto original que o produza
  - Ou seja: não podemos passar de um resumo para os dados (não os podemos "desencriptar")

- Resistência à 2ª pré-imagem
  - Dado um texto, é impossível encontrar outro com o mesmo resumo
  - Ou seja: se **tivermos um texto**, não podemos encontrar outro com o mesmo resumo

- Resistência à colisão
  - É impossível encontrar dois textos com a mesma digestão
  - Ou seja: dados dois textos únicos, estes resultarão **num resumo diferente**
    - Relacionado com o paradoxo do aniversário: probabilidade de colisão $P = 2^{n/2}$ quando o *n* típico é >=256

Funções de digestão

Vamos verificar: Independência do tamanho

- Considerando os textos semelhantes, mas diferentes:
  - T1: "Olá Utilizador_A!"
  - T2: "Olá Utilizador_XPTO! Bem-vindo a esta aula"

- Algoritmos diferentes criarão resumos com comprimentos diferentes, mas **independentes** da dimensão do texto
  - MD5 (128 bits):
    - T1: 70df836fdaf02e0dfc990f9139762541
    - T2: 18f12f09c45d880ce738afe4780c2f3e
  - SHA-1 (160 bits):
    - T1: f591aa1eabcc97fb39c5f422b370ddf8cb880fde
    - T2: 622f7832e204f2d70161cf42480c4bf0f13e7324
  - SHA-256 (256 bits):
    - T1: 9649d8c0d25515a239ec8ec94b293c8868e931ad318df4ccd0dffd67aff89905
    - T2: 6453be3f643d0a7e9b5890eed76bb63df8b6b071b30d5f97269a530c289b9839

Funções de digestão

Vamos verificar: Dependência de conteúdo

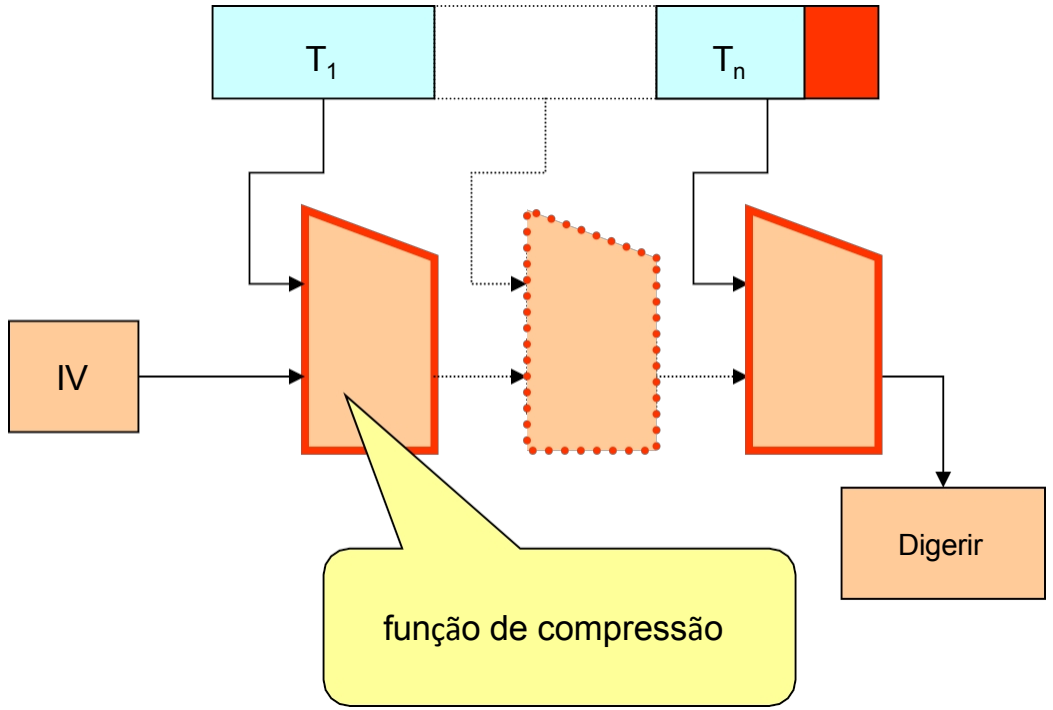
- Considerando os textos semelhantes, mas diferentes (1 bit de diferença 'B' -> 'C'):
  - T1: "Olá Utilizador_B!", [0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x20, 0x55, 0x73, 0x65, 0x72, 0x5f, **0x42**, 0x21]
  - T2: "Olá Utilizador_C!", [0x48, 0x65, 0x6c, 0x6c, 0x6f, 0x20, 0x55, 0x73, 0x65, 0x72, 0x5f, **0x43**, 0x21]

- Uma pequena diferença no texto (1 bit) resulta num **digest completamente diferente**
  - MD5:
    - T1: c32e0f62a7c9c815063d373acac80c37
    - T2: 324a1bfc3041259480c6ad164cf0529f
  - SHA-1:
    - T1: bab31eb62f961266758524071a7ad8221bc8700b
    - T2: bd758d82899d132cd2af66dc3402b948d98de62d
  - SHA-256:
    - T1: e663a01d3bec4f35a470aba4baccece79bf484b5d0bffa88b59a9bb08707758a
    - T2: 69f78345da90c6b8d4785b769cd6ae09e0531716fe5f5a392fde1bdc70a2bb7d

Funções de digestão

Abordagens

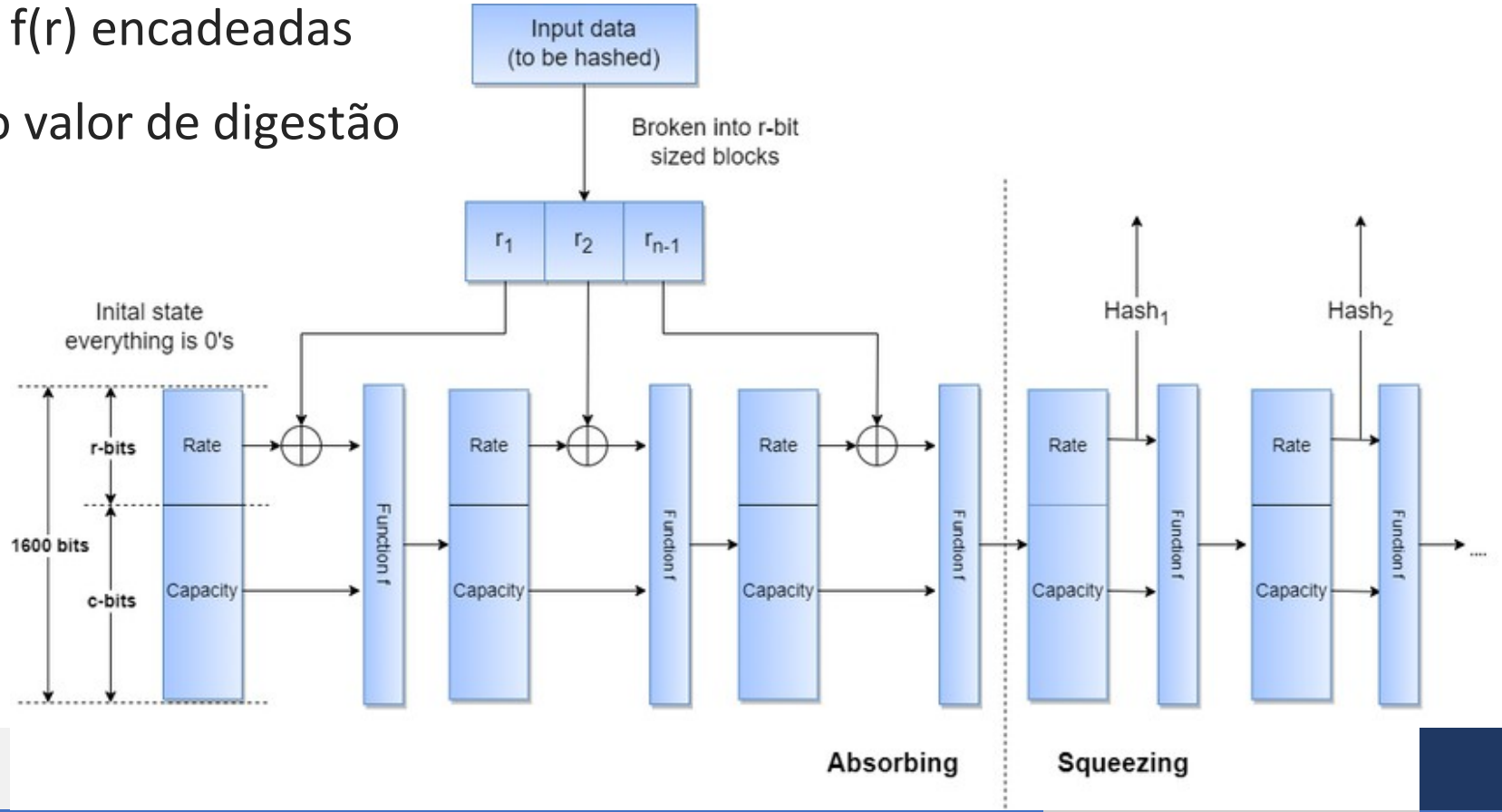
- Construção de Merkle-Damgård
  - Funções de compressão unidirecional resistente à colisão
    - Pode ser uma cifra de bloco!
  - Compressão iterativa
  - Acolchoamento do comprimento
  - O tamanho do resumo é o último bloco
  - Pode ser retomado!
    - Digesto é o estado em T_n
  - Algoritmos: MD5, SHA1, SHA2



Funções de digestão

Abordagens

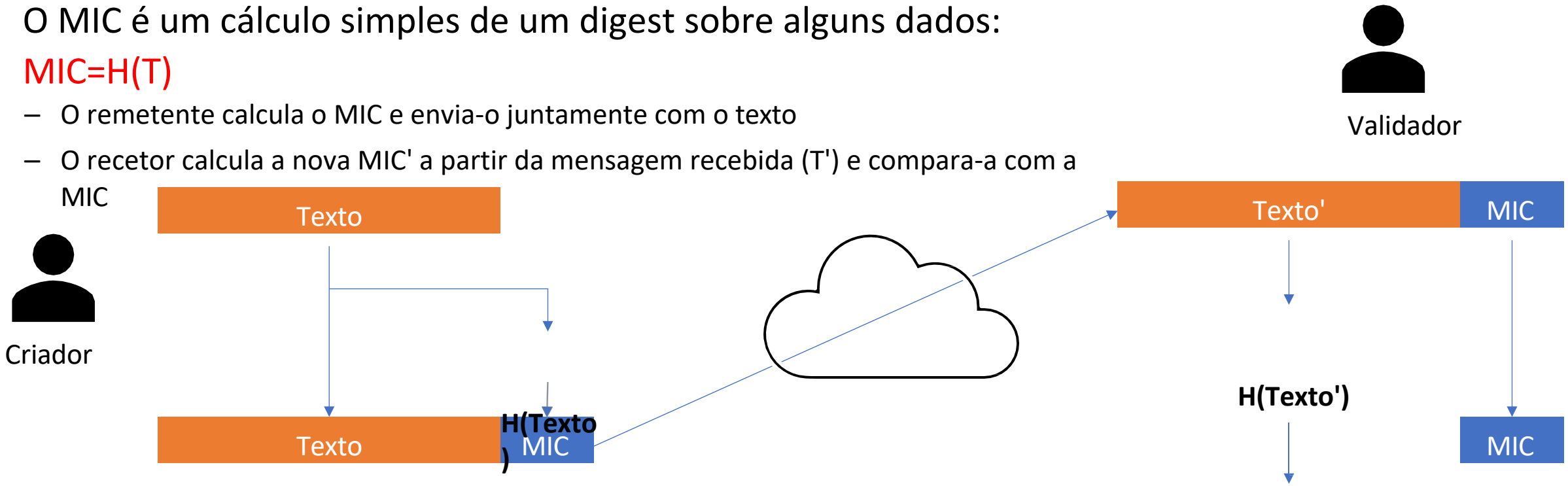
- Funções da esponja
  - Dados divididos em blocos de tamanho *r*
  - Fase de absorção: chamadas f(r) encadeadas
  - Squeezing: extrair bits para o valor de digestão
  - Algoritmos: SHA3



Código de Integridade da Mensagem (MIC)

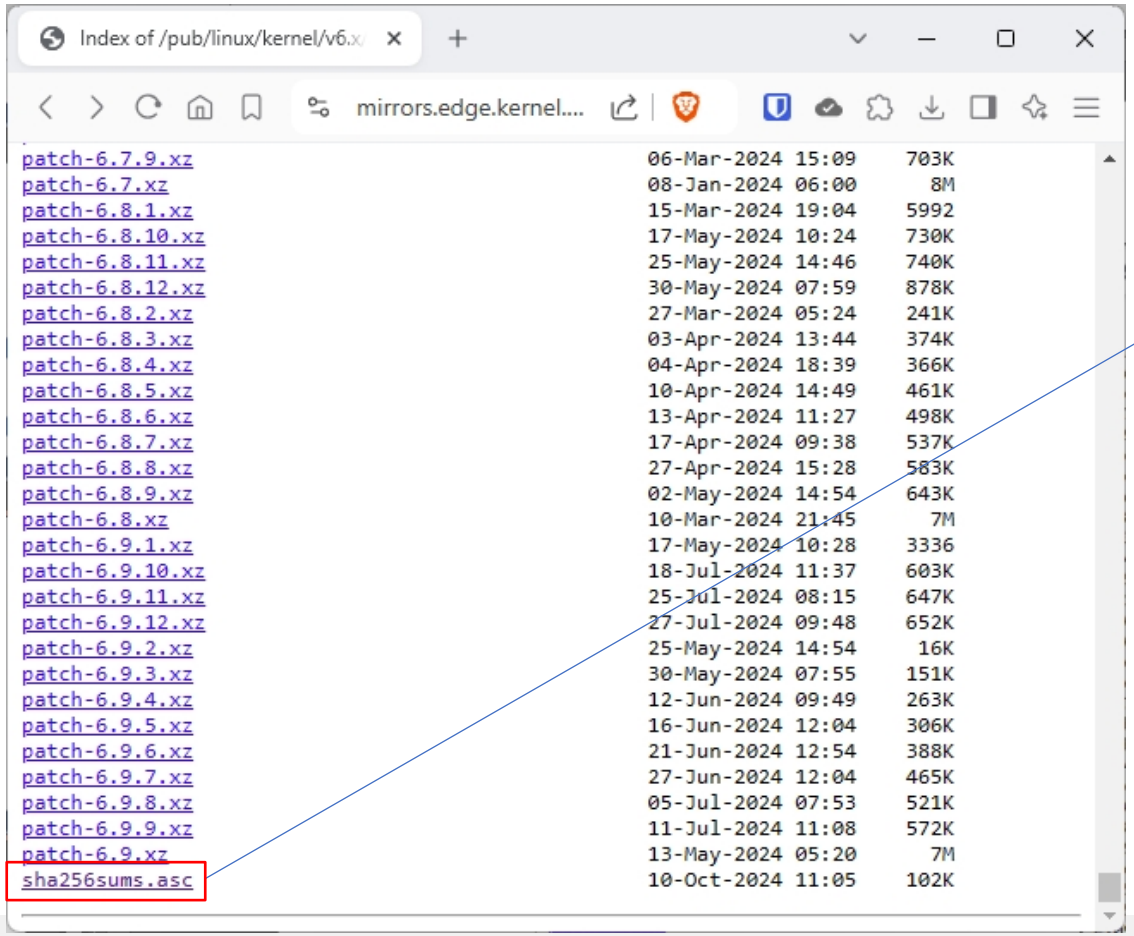
- Fornecer a capacidade de detetar alterações **arbitrárias** nos dados
  - Erros de comunicação/armazenamento de um processo aleatório ou sem controlo de integridade
  - Os humanos/atacantes podem alterar o texto e calcular um novo MIC!

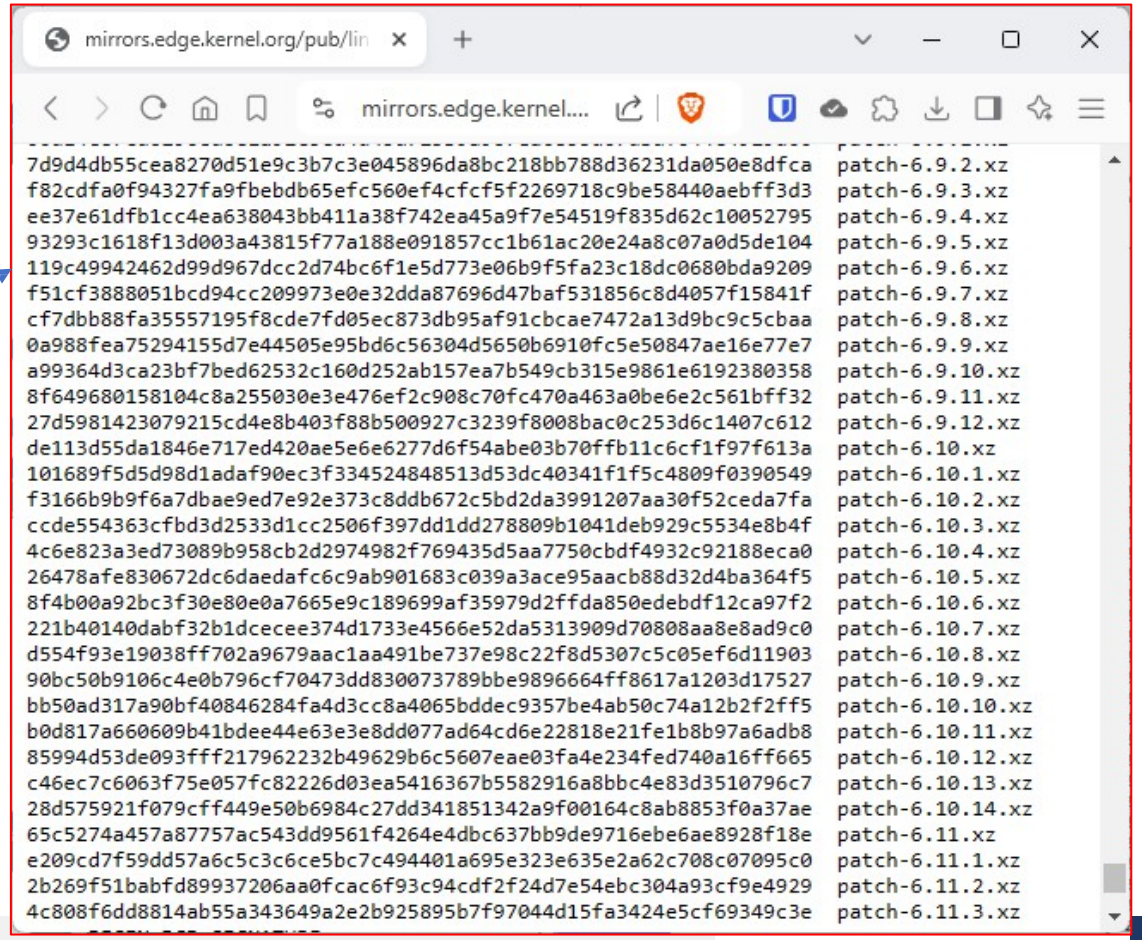
- O MIC é um cálculo simples de um digest sobre alguns dados:
  MIC=H(T)
  - O remetente calcula o MIC e envia-o juntamente com o texto
  - O recetor calcula a nova MIC' a partir da mensagem recebida (T') e compara-a com a MIC



MIC' **iguais?**

Exemplo de utilização em kernel.org para validar a integridade do ficheiro

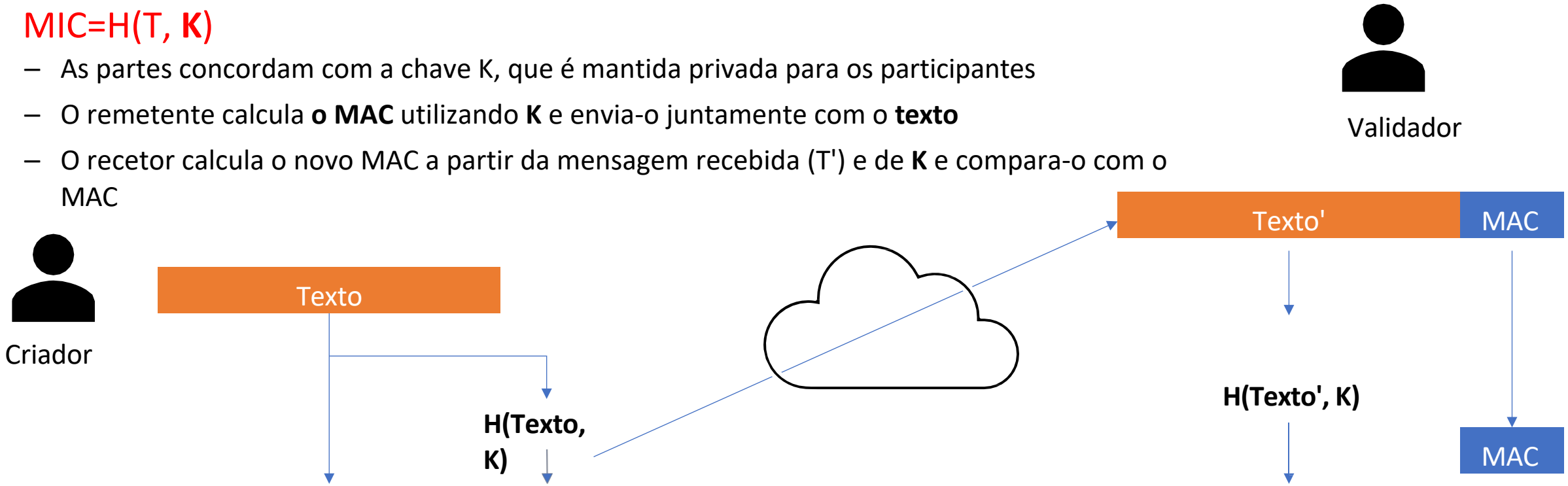




Código de autenticação de mensagem (MAC)

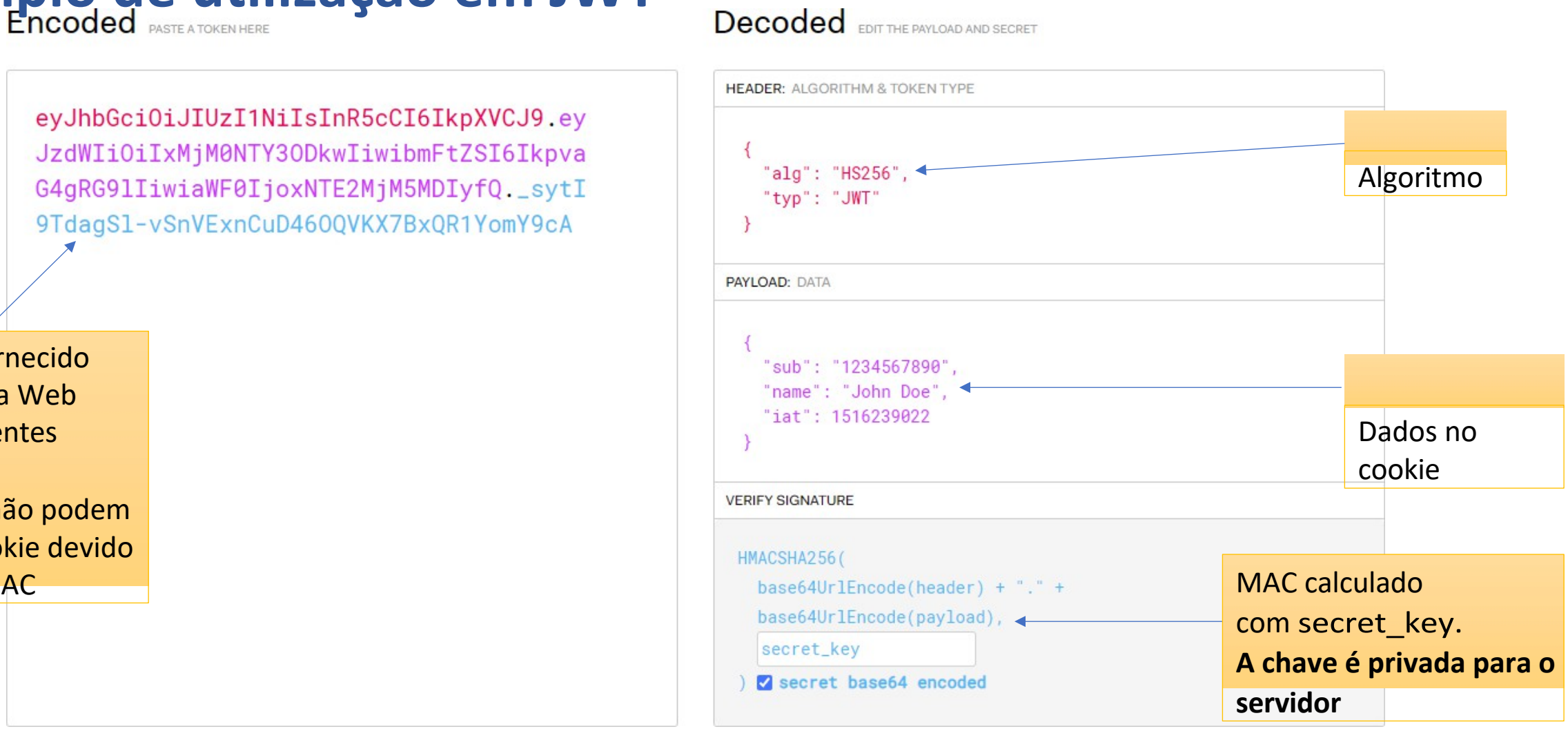
- Fornecer a capacidade de detetar alterações **deliberadas** nos dados
  - Qualquer alteração nos dados, mesmo que seja efectuada por atacantes!

- O MAC é um cálculo chaveado de um resumo sobre alguns dados:
  MIC=H(T, **K**)
  - As partes concordam com a chave K, que é mantida privada para os participantes
  - O remetente calcula **o MAC** utilizando **K** e envia-o juntamente com o **texto**
  - O recetor calcula o novo MAC a partir da mensagem recebida (T') e de **K** e compara-o com o MAC



iguais?

Exemplo de utilização em JWT



https://jwt.io/#debugger-io?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ._sytI9TdagSl-vSnVExnCuD46OQVKX7BxQR1YomY9cA

Código de autenticação de mensagem (MAC)

Abordagens

- Encriptação de um resumo normal (por exemplo, de SHA3)
  - Utilizando, por exemplo, uma cifra de bloco simétrica
- Utilizar a encriptação com feedback e propagação de erros
  - CBC-MAC ou GCM
- Adicionar uma chave aos dados com hash
  - MD5 com chave (128 bits)
    - MD5(K, keyfill, texto, K, MD5fill)
  - HMAC (o comprimento de saída depende da função H utilizada)
    - H(K, opad, H(K, ipad, texto))
    - ipad = 0x36 B timesopad = 0x5C B times B = tamanho do bloco de entrada H
      - HMAC-MD5, HMAC-SHA-1, etc.

Código de autenticação de mensagem (MAC)

Quando utilizado com encriptação

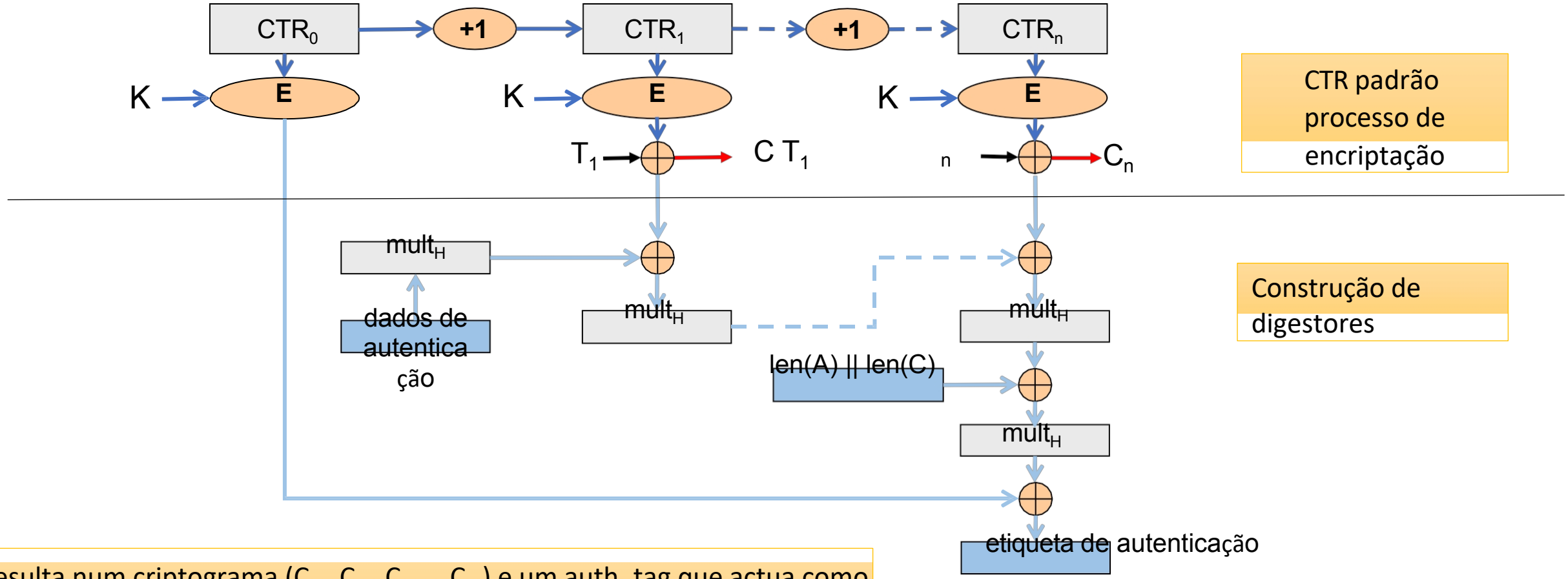
- **Encriptar e depois MAC: O MAC é calculado a partir do criptograma: M = C | MAC(C, K_2), C=E(T, K)$_1$**
  - **Permite verificar a integridade antes da desencriptação**
  - **O cálculo do MAC é frequentemente mais rápido do que a desencriptação**

RUIM

- **Encriptar-e-MAC: O MAC é calculado a partir do texto simples: M = E(T, K_1) | MAC(T, K)$_2$**
  - Pode dar informações sobre o texto original (se for semelhante a outro texto)
  - O recetor verificará que o texto **só** foi manipulado **após a desencriptação e o cálculo do MAC (mais lento)**
  - O texto cifrado manipulado pode atacar o algoritmo de desencriptação sem ser detectado

- **MAC-then-Encrypt: O MAC é calculado a partir do texto simples: M = E(T | MAC(T, K_2), K)$_1$**
  - O MAC é encriptado (o que não é mau)
  - O recetor verificará que o texto **só** foi manipulado **após a desencriptação e o cálculo do MAC (mais lento)**

Exemplo: GCM (Modo Contador Galois)



CTR padrão processo de encriptação

Construção de digestores

Resulta num criptograma (C_1 , C_2 , C_3 ... C_n) e um auth_tag que actua como MAC

Requer um auth_data adicional

Derivação de chaves

Motivação

- Os algoritmos de cifra requerem chaves de dimensão fixa

  -56 , 128, 256... bits

- Podemos precisar de derivar chaves de várias fontes
  - Segredos partilhados
  - Palavras-passe geradas por humanos
  - Códigos PIN e segredos de pequena dimensão

- A fonte original pode ter baixa entropia
  - Reduz a dificuldade de um ataque de força bruta
  - Embora tenhamos de ter uma relação forte com uma chave útil

- Por vezes, são necessárias várias chaves do mesmo material
  - Embora não permita encontrar o material (uma palavra-passe, outra chave) a partir da nova chave

Derivação de chaves

Objectivos

- **Reforço de chaves**: aumentar a segurança de uma palavra-passe
  - Normalmente definido por humanos
  - Para tornar os ataques de dicionário impraticáveis

- **Expansão da chave**: aumentar/diminuir o comprimento de uma chave
  - Expansão para um tamanho adequado a um algoritmo
  - Eventualmente derivar outras chaves relacionadas para outros algoritmos (por exemplo, MAC)

Derivação de chaves

- A derivação de chaves requer a existência de:
  - Um **sal** que torna a derivação única
  - Um problema difícil
  - Um nível de complexidade escolhido

- Dificuldade computacional

  - A transformação requer recursos computacionais relevantes

- Dificuldade de memória
  - A transformação requer recursos de armazenamento relevantes
  - Limita os ataques que utilizam aceleradores de hardware dedicados

Derivação de chaves

Abordagem simples: Uma função Digest

- Argumentos:
  - Salt = Um valor aleatório
  - Palavra-passe = um segredo (fornecido por humanos)
  - H = Uma função de digestão adequada

key = H(password, salt)

- Vantagens:
  - A chave tem um comprimento grande e pode ser truncada para o comprimento adequado
  - Duas palavras-passe resultarão em chaves diferentes
  - Encontrar a chave não conduzirá à palavra-passe

- Problemas: simples, permitindo ataques de força bruta/diccionários

Derivação de chaves

Função de derivação de chaves baseada em palavra-passe (PBKDF2)

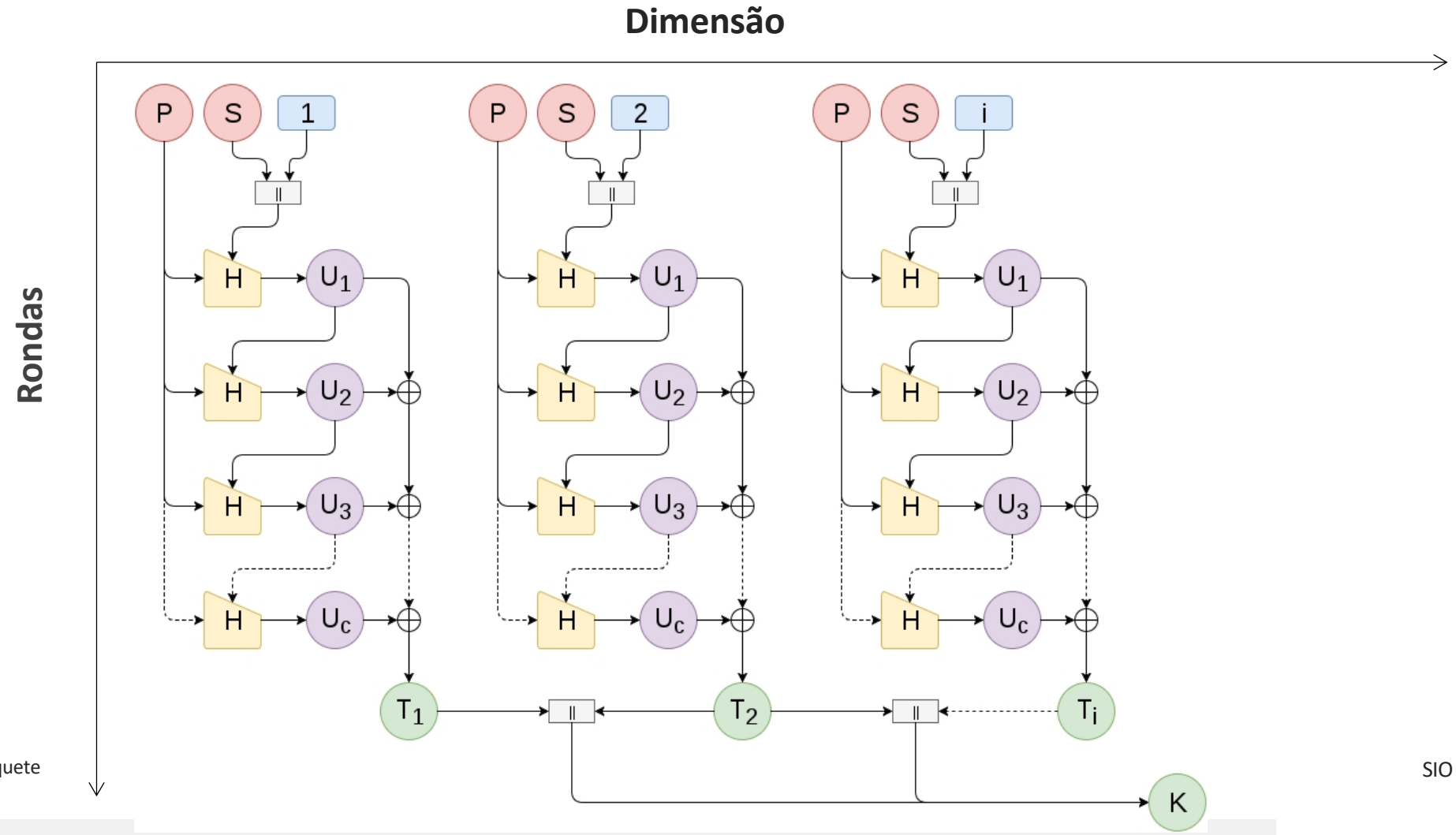
- Produz uma chave a partir de uma palavra-passe, com uma dificuldade escolhida

- **K = PBKDF2(PRF, Salt, rounds, dim, password)**
  - PRF: Função Pseudo-Aleatória: uma função de digestão
  - Sal: um valor aleatório
  - Rondas: o custo computacional (centenas de milhares)
  - Dim: o tamanho do resultado pretendido

- Operação: calcular as operações ROUNDS x DIM da PRF utilizando o SALT e a Password
  - Um maior número de rondas aumentará o custo dos ataques de força bruta/diccionários

Derivação de chaves

Função de derivação de chave baseada em palavra-passe (PBKDF2)



Derivação de chaves

criptografar

- Produz uma chave com um custo de computação e armazenamento escolhido

- **K = scrypt(password, salt, n, p, dim, r, hLen, Mflen)**
  - Palavra-passe: um segredo
  - Sal: um valor aleatório
  - N: o parâmetro de custo
  - P: o parâmetro de paralelização. p ≤ (232- 1) * hLen / MFLen
  - Dim: o tamanho do resultado
  - R: o tamanho dos blocos a utilizar (a predefinição é 8)
  - hLen: o tamanho da função de digestão (32 para SHA256)
  - Mflen: bytes na mistura interna (a predefinição é 8 x R)

Derivação de chaves: scrypt

- Produz uma chave com um custo de armazenamento selecionado

- K = scrypt(password, salt, n, p, dim, r, hLen, Mflen)
  - Palavra-passe: um segredo
  - Sal: um valor aleatório
  - N: o parâmetro de custo
  - P: o parâmetro de paralelização. $p \leq (2^{32} - 1) * hLen / MFLen$
  - Dim: o tamanho do resultado
  - R: o tamanho dos blocos a utilizar (a predefinição é 8)
  - hLen: o tamanho da função de digestão (32 para SHA256)
  - Mflen: bytes na mistura interna (a predefinição é 8 x R)

Derivação de chaves

criptografar