Sistema de Autenticação

- Conjunto A de Informação de Autenticação
- Conjunto V de Informação de Validação
- Função $f: A \to V$
- Função $g: V \times A \to \{ true, false \}$

Exemplo:

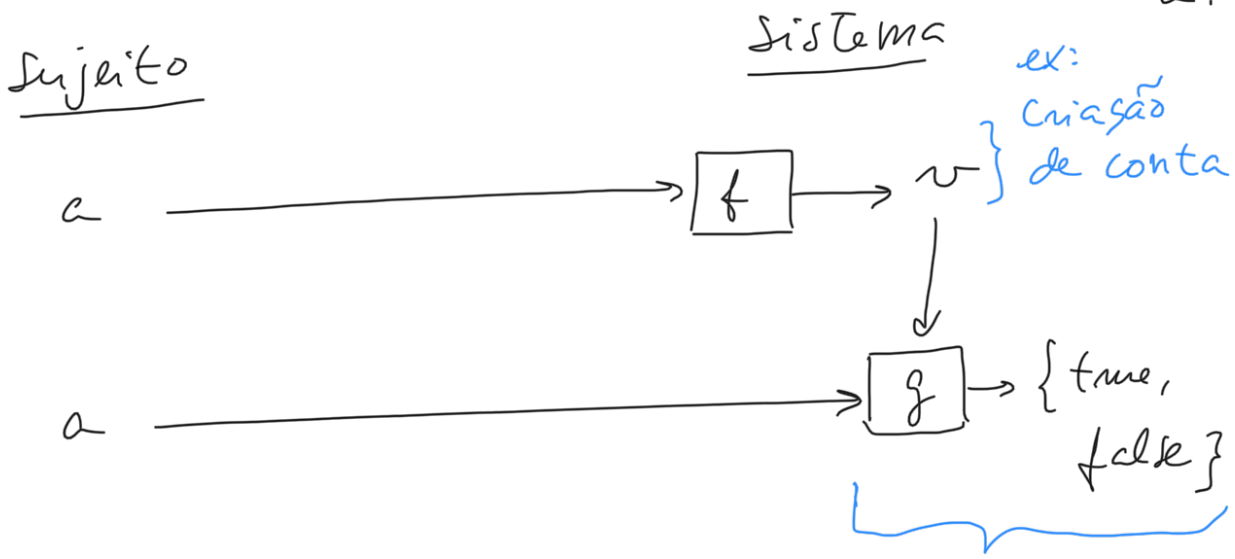
- $f(a) = H(a)$ — função que utiliza uma função de hash criptográfica

$a = \{ username, password \}$

username → id

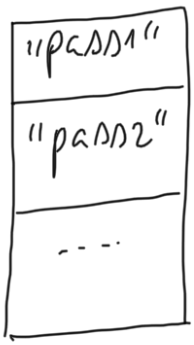
password → autenticação

- $$g(v)(a) = \begin{cases} true, & \text{se } v = H(a) \\ false, & \text{se } v \neq H(a) \end{cases}$$



Passwords: ataques de dicionário

Tipo 1: dicionário de passwords



- entrada: informação de validação V
- saída: informação de autenticação

1. Para cada a' em Dicionário
   1.1. Se $f(a') = v$ retornar a'
2. Retornar "Falha"

Tipo 2

Entrada: função de autenticação $g(v)$

Saída: informação de autenticação

1. Para cada a' em Dicionário
   1.1. Se $g(v)(a') = true$
   retornar a'
2. Retornar "Falha"

Pesquisa de Passwords

- Exaustiva (Brute Force)
  Testa todas as palavras possíveis de forma arbitrária
- Inteligente
  ↓
  Testa primeiro passwords prováveis
  - Dicionários
  - Algoritmos que combinam palavras

Proteção contra ataques de dicionário

- Aumentar incerteza da password
  - ↳ passwords aleatórias → Problema: memorização
  - ↳ seleção proativa – é escolhida pelo utilizador mas entretanto é verificada pelo sistema
    - ↳ verificação offline
    - ↳ uso de "salt"

Ex: não pode ter nomes presentes em dicionários, incluir dados do utilizador (BI, NIF, matrícula, ...), ter letras minúsculas, maiúsculas, números, caracteres especiais, etc.

- Controlar o acesso à informação de verificação
  ex: no Linux, o hash da password é guardado no shadow file
  ↓
  acedido apenas por root

- Aumentar tempo de processamento de funções $f(a)$ e $g(v)(a)$

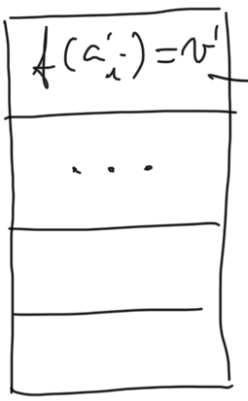
- e/ou limitar acesso a $g(v)(a)$
  ex: atrasar a apresentação do ecrã de login num telefone de forma proporcional ao número de tentativas de login falhadas.

Proteção contra ataques de Tipo 1

- Tornar função $f(a)$ mais demorada
  - ex: função de hash iterada n vezes
- Controlar acesso à informação de verificação
  - ex: Linux shadow file

Ataques com pré-computação (Rainbow attack)

Array associativo M:



- Para todos os utilizadores
- f é semelhante para todos os utilizadores

Pré-calcular entradas

Ataque:
Dado v, retornar $M[v]$

Proteção "salt"

- Proteção contra ataques de dicionário e de pré-computação

  ↳ limita pesquisa de passwords separadamente para cada salt

- tornar f diferente para cada utilizador

  ex: $f_u(a) = hash(a \,||\, salt_u)$

  ↓

  bytes aleatórios mas conhecidos

  ↳ semelhante ao IV na cifra simétrica

- Pré-computação depende de salt
  ↳ não pode ser usada para atacar todos os utilizadores do sistema
  ↳ aumenta drasticamente o número de entradas de M

Proteção contra ataques de Tipo 2

↳ limitar acesso à função de autenticação $g(v)(a)$ após deteção de tentativas de autenticação falhadas

ex: demoran a executar $g(v)(a)$; Terminar ligação; bloquear utilizador, ou jailing (acesso ao serviço com funcionalidade limitada)

=> Problema: Limitar acesso a intrusos e ao mesmo tempo garantir disponibilidade do serviço

Medidas possíveis:

- Usar CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Humans Apart)

- Two-way factor (ex: autenticação com password + token; enviar SMS com token)

- Protocolos challenge-Response

- Passwords de utilização única (OTP - one time password)
  ↳ ex: S/key algorithm

Password aging

↳ limitar tempo de utilização duma password

↳ memorização de passwords anteriores pelo servidor

Ameaça: Spoofing attacks

↳ obtenção da password através da simulação da interface de autenticação ou intercepção desta

- Prevenção
  - ↳ Trusted Path (ex: CTRL+ALT+DEL no windows) ↓ Secure Attention Sequence
  - ↳ Autenticação mútua
- Deteção: Registo e apresentação do número e data das autenticações falhadas