

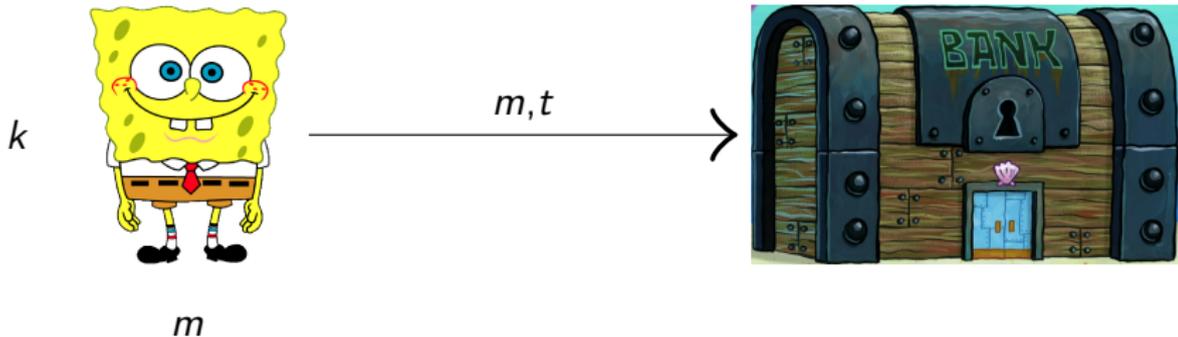
GBC083 – Segurança da Informação

Aula 5 - Integridade e Criptografia autenticada

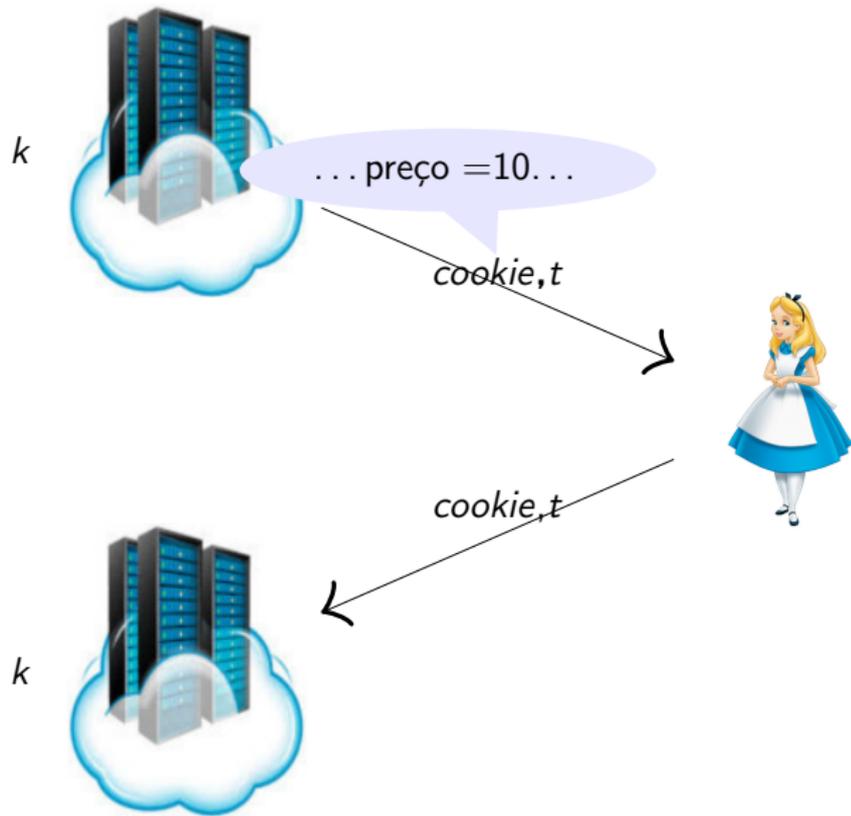
9 de Maio de 2016

Integridade de mensagens – Sigilo vs. Integridade

- ▶ Até agora, preocupados com garantia de sigilo da comunicação
- ▶ E sobre **integridade**?
 - ▶ Garantir que uma mensagem recebida é originada da pessoa certa e que não foi modificada
 - ▶ Mesmo se o atacante controla o canal
 - ▶ Técnicas padrões de detecção e correção de erros não são suficientes
 - ▶ Ferramenta certa é um **código de autenticação de mensagens**



- ▶ Garantir a identidade em operações



► Comunicações em diferentes momentos: a ordem é autêntica

Sigilo e integridade

- ▶ Sigilo e integridade são preocupações **ortogonais**
 - ▶ Possível ter um sem o outro
- ▶ Encriptação não provê, em geral, qualquer integridade
 - ▶ Relacionado ao problema da maleabilidade

Código de autenticação de mensagens

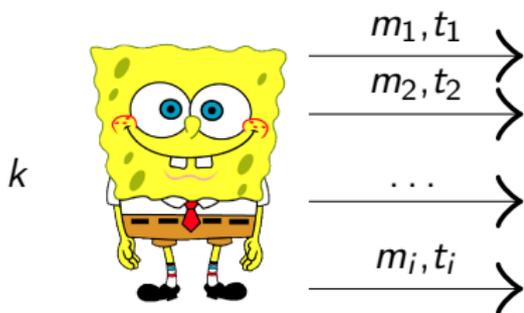
- ▶ Message authentication code (MAC)
- ▶ Um MAC é definido por 3 algoritmos (Gen , Mac , $Vrfy$):
 - ▶ Gen : recebe entrada 1^n , produz chave uniforme k , $|k| = n$.
 - ▶ Mac : recebe como chave de entrada k e mensagem $m \in \{0, 1\}^*$; produz etiqueta $t \in \{0, 1\}^*$

$$t \leftarrow Mac_k(m)$$

- ▶ $Vrfy$: recebe chave k , mensagem m , e etiqueta t como entrada; produz 1: **aceita** ou 0: **rejeita**
- ▶ Para toda m e todo k produzido por Gen ,
 $Vrfy_k(m, Mac_k(m)) = 1$

Segurança?

- ▶ Apenas uma definição padrão
- ▶ Modelo de ameaça
 - ▶ Ataque de escolha de mensagem adaptativa
 - ▶ Supor que o atacante pode induzir o transmissor a autenticar mensagens da escolha do atacante
- ▶ Meta de segurança
 - ▶ Inforjabilidade existencial
 - ▶ Atacante não deve ser capaz de forjar uma etiqueta válida em qualquer mensagem não autenticada pelo transmissor

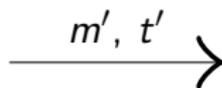


$$t_1 \leftarrow \text{Mac}_k(m_1)$$

$$t_2 \leftarrow \text{Mac}_k(m_2)$$

\dots

$$t_t \leftarrow \text{Mac}_k(m_t)$$



$\text{Vrfy}_k(m', t')???$

Definição formal

- ▶ Fixar A, Π
- ▶ Definir experimento aleatório $Forge_{A, \Pi}(n)$:
 - ▶ $k \leftarrow Gen(1^n)$
 - ▶ A interage com um oráculo $Mac_k(\cdot)$; seja M o conjunto de mensagens enviado a esse oráculo
 - ▶ A produz (m, t)
 - ▶ A é bem sucedido e o experimento produz 1 se $Vrfy_k(m, t) = 1$ e $m \notin M$

Segurança para MACs

- ▶ Π é seguro se para todos atacantes em tempo polinomial A , existe uma função negligível ϵ tal que

$$P(\text{Forge}_{A,\Pi}(n) = 1) \leq \epsilon(n)$$

Ataques replay

- ▶ Note que ataques replay não prevenidos
 - ▶ Nenhum mecanismo sem estado pode previr ataques replay
- ▶ Ataques replay podem ser uma preocupação real
- ▶ Necessário proteção contra ataques replay em um nível mais alto

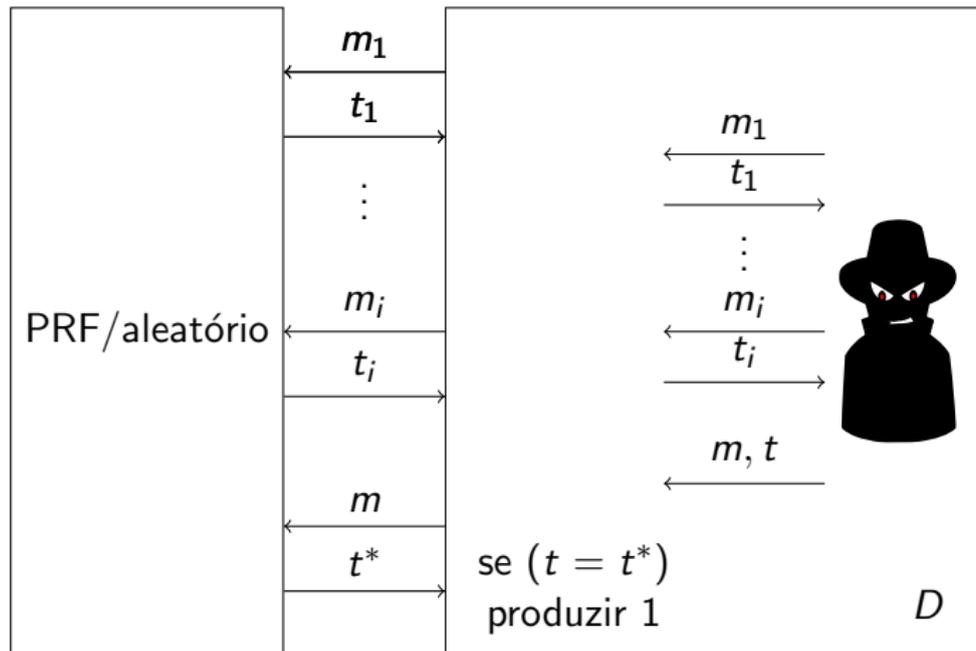
Intuição para desenvolvimento de método seguro para integridade

- ▶ Necessária uma função chaveada Mac tal que:
 - ▶ dados $Mac_k(m_1)$ e $Mac_k(m_2), \dots$,
 - ▶ deve ser inviável prever o valor $Mac_k(m)$ para qualquer $m \in \{m_1, \dots, \}$
- ▶ Podemos usar uma função pseudo-aleatória como sendo o Mac

Construção de segurança para integridade: primeira versão

- ▶ Seja F uma função pseudo-aleatória que preserve comprimento da entrada
- ▶ Construir o seguinte esquema Π MAC:
 - ▶ Gen – escolher uma chave uniforme k para F
 - ▶ $Mac_k(m)$ – computar $F_k(m)$
 - ▶ $Vrfy_k(m, t)$ – produzir 1 se e só se $F_k(m) = t$
- ▶ Teorema: esquema Π é um MAC seguro.

Prova por redução



- ▶ Supor existência de um “diferenciador” entre PRF ou aleatório que usa o atacante do tipo *Forge*

Análise

- ▶ Duas opções:
 - ▶ Quando D interage com F_k para uma chave uniforme k , a visão do adversário é idêntica à visão do atacante no experimento MAC real
 - ▶ $P(D^{F_k} \text{ produz } 1) = P(\text{Forge}_{Adv, \Pi}(n) = 1)$
 - ▶ Quando D interage com f uniforme, então sendo $f(m_1), \dots, f(m_i)$ não ajuda prever $f(m)$ para qualquer $m \notin \{m_1, \dots, m_i\}$
 - ▶ $P(D^f \text{ produz } 1) = 2^{-n}$

Análise

- ▶ Com F é uma função pseudo-aleatória

$$|P(D^{F_k} \text{ produz } 1) - P(D^f \text{ produz } 1)| < \epsilon(n)$$

Portanto,

$$P(\text{Forge}_{Adv, \Pi}(n) = 1) = P(D^{F_k} \text{ produz } 1) \leq 2^{-n} + \epsilon(n)$$

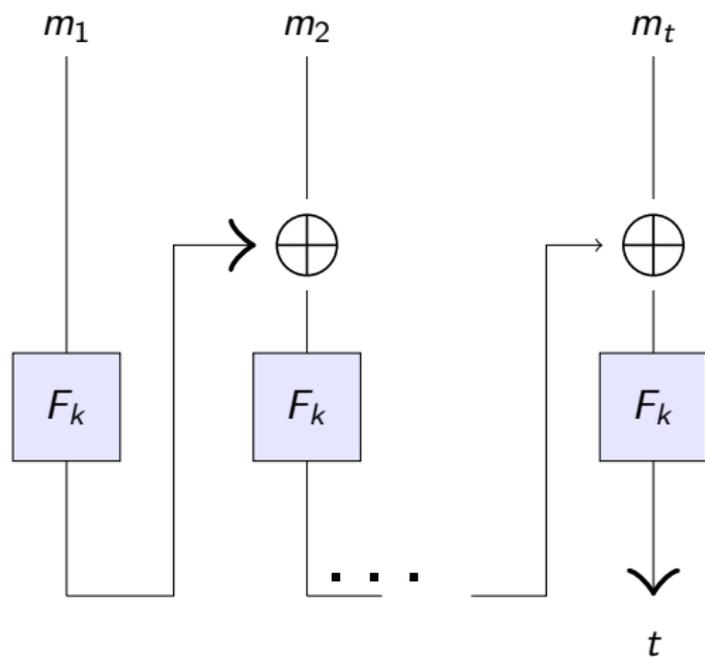
Desvantagens deste primeiro método baseado em cifra de 1 bloco

- ▶ Na prática, cifra de blocos (aka funções pseudo-aleatórias) tem tamanho pequeno e fixo
 - ▶ Exemplo, AES tem bloco de 128 bits
 - ▶ Então a construção prévia é limitada à autenticação de mensagens curtas (comprimento fixo)

Até agora

- ▶ Mostramos como construir um MAC seguro para mensagens curtas de tamanho fixo baseado em qualquer cifra de blocos (função pseudo-aleatória)
- ▶ Queremos obter um MAC seguro para mensagens de tamanho arbitrário
 - ▶ Aqui será o CBC-MAC

Modo CBC-MAC básico



Modo CBC-MAC vs. Modo CBC

- ▶ CBC-MAC é determinístico (sem IV)
- ▶ Em CBC-MAC, somente o valor final é mostrado
 - ▶ Verificação é realizada por recomputação do resultado
- ▶ Ambos são essenciais para segurança

Segurança do CBC-MAC básico

- ▶ Se F é uma função pseudo-aleatória que preserva comprimento, então para qualquer l fixo, o CBC-MAC é um MAC seguro para mensagens de comprimento $l \cdot n$
- ▶ O transmissor e o receptor devem concordar com o parâmetro de comprimento l antecipadamente
 - ▶ CBC-MAC básico **não** é seguro se isto não é feito
 - ▶ Sejam (m, t) e (m', t') conhecidos
 - ▶ Atacante consegue gerar: $m'' = m | [(m'_1 \oplus t) | m'_2 | \dots]$
 - ▶ $\text{CBC-MAC}(m'') = m'$ pois o $m'_1 \oplus t$ elimina contribuição da primeira parte

Extensões CBC-MAC

- ▶ Várias maneiras de manipular mensagens de tamanho variável
- ▶ Uma simples: transmitir tamanho inicial antes de aplicar CBC-MAC básico
 - ▶ Também pode ser adaptado para mensagens que não tem tamanho múltiplo do comprimento do bloco
- ▶ Outra forma: usar outra chave e encriptar último bloco (Encrypt Last Block):
$$\text{CBC-MAC-ELB}(m, (k_1, k_2)) = E(k_2, \text{CBC-MAC}(k_1, m))$$

Modo CBC-MAC: "prepend length"

