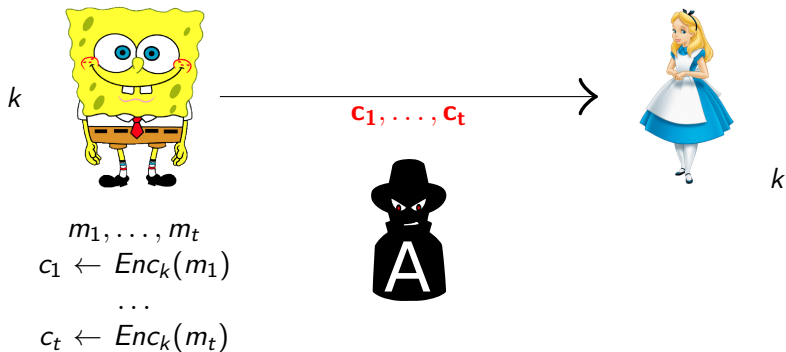


# Noções de segurança ainda mais fortes

- ▶ Até agora, usou-se modelo de atacantes passivos que apenas escutavam as comunicações
- ▶ Mudança do tipo de ataque considerado: atacante ativo
- ▶ Atacante pode modificar texto cifrado sendo enviado
- ▶ Segurança contra ataque de texto cifrado escolhido
- ▶ Em inglês, *Chosen-Ciphertext Attack*-security

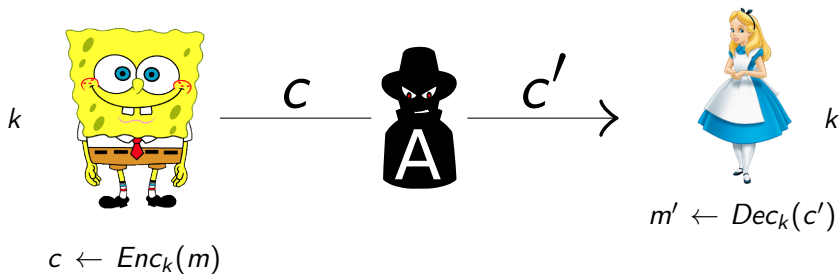
# Sigilo de múltiplas mensagens



# Até agora

- ▶ E se atacante pode ser ativo e interferir no canal de comunicação?

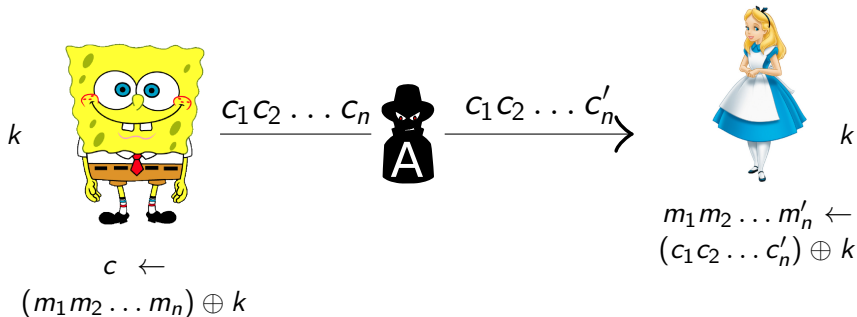
# Atacante interfere no canal



# Maleabilidade

- ▶ Informal: um esquema é maleável se é possível modificar um texto cifrado e causar uma mudança predizível ao texto em claro
- ▶ Maleabilidade pode ser perigosa
  - ▶ Modificação de conteúdo de emails
  - ▶ Alteração de números em transações financeiras
- ▶ Todos os esquemas visto até agora são maleáveis
- ▶ One-Time Pad é maleável

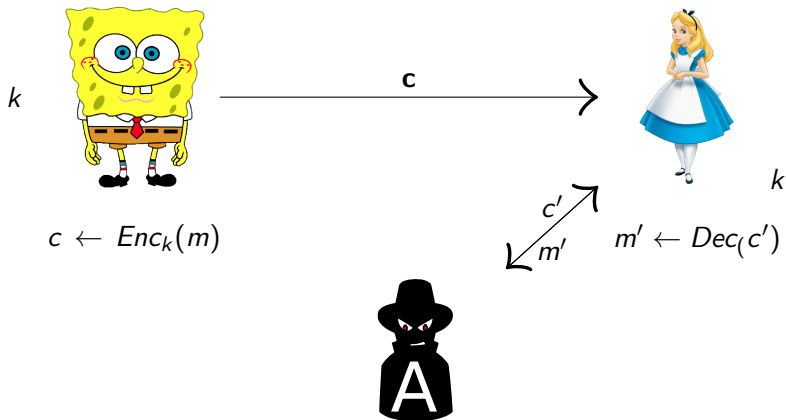
# Atacante interfere no canal (OTP)



# Até agora

- ▶ E se atacante pode ser ativo e interferir no canal de comunicação?
  - ▶ Impersonificação do emissor original
  - ▶ Injeção de dados no canal de comunicação

# Injeção





# Ataques de textos cifrados escolhidos

- ▶ Chosen Cyphertext Attack (CCA)
- ▶ Modelar condições em que atacante pode influenciar o que é decriptografado e observar os efeitos
  - ▶ Como modelar?
- ▶ Permite atacante enviar textos cifrados de sua escolha ao recipiente e aprender o texto em claro correspondente
  - ▶ Em adição ao ataque de texto em claro escolhido

# Segurança contra CCA

- ▶ Definir experimento aleatório  $PrivCCA_{A,\Pi}(n)$ :
  - ▶  $k \leftarrow Gen(1^n)$
  - ▶  $A(1^n)$  interagem com uma **oráculo de encriptação**  $Enc_k(\cdot)$  e um **oráculo de deciptação**  $Dec_k(\cdot)$  e produz  $m_0$  e  $m_1$  do mesmo comprimento
  - ▶  $b \leftarrow \{0, 1\}$ ,  $c \leftarrow Enc_k(m_b)$
  - ▶  $A$  pode continuar a interagir com  $Enc_k(\cdot)$ ,  $Dec_k(\cdot)$ , mas não pode requisitar deciptação de  $c$
  - ▶  $A$  produz  $b'$ ;  $A$  é bem sucedido se  $b = b'$  e experimento neste caso produz 1

# Segurança contra CCA

- ▶  $\Pi$  é seguro contra CCA se para todos atacantes de tempo polinomais  $A$ , existe uma função negligível  $\epsilon$  tal que

$$P(\text{PrivCCA}_{A,\Pi}(n) = 1) \leq 0.5 + \epsilon(n)$$

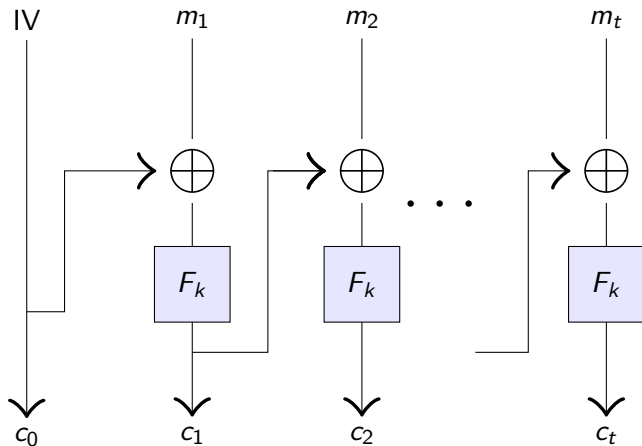
# Ataque de texto cifrado escolhido e maleabilidade

- ▶ Se um esquema for maleável, então não pode ser seguro contra CCA
  - ▶ Modificar  $c$ , enviar texto cifrado modificado  $c'$  para o oráculo de decifração
- ▶ Segurança contra CCA implica em não-maleabilidade

# Ataques de oráculos de preenchimento (*padding-oracle*)

- ▶ Na definição de segurança contra CCA, o atacante pode obter a decifração de qualquer texto cifrado de sua escolha (além do texto cifrado de desafio)
  - ▶ Isto é realista?
- ▶ Mostraremos um cenário real onde:
  - ▶ Um bit sobre textos cifrados decifrados é vazado
  - ▶ Isto pode ser explorado para descobrir todo o texto em claro

# Modo CBC



# Mensagens de comprimento arbitrário

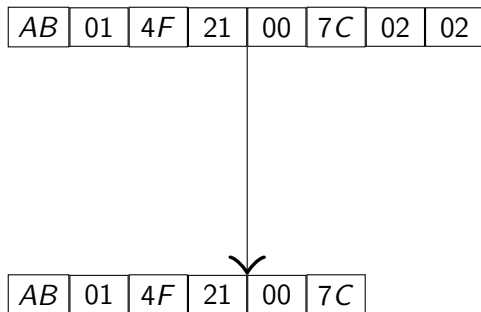
- ▶ Mensagem  $\rightarrow$  dados codificados  $\rightarrow$  texto cifrado
  - ▶ Considere que mensagem é composta de  $N$  bytes
- ▶ Codificação PKCS #5:
  - ▶ Seja  $L$  o comprimento do bloco em bytes da cifra
  - ▶ Seja  $b$  o número de bytes que são necessários para adicionar à mensagem para alcançar o tamanho múltiplo de  $L$ 
    - ▶  $1 \leq b \leq L$  note que  $b \neq 0$
  - ▶ Concatenar  $b$  na mensagem (codificado em 1 byte),  $b$  vezes
    - ▶ Ou seja, se 3 bytes de padding são necessários, concatenar 0x030303
  - ▶ Se  $N$  é múltiplo de  $L$ , concatenar  $L$  bytes com valor  $L$

# Decriptação

- ▶ Para decriptar:
  - ▶ Usar decriptação em modo CBC para obter dados
  - ▶ Digamos que o byte final dos dados tem valor  $b$ 
    - ▶ Se  $b = 0$  ou  $b > L$ , retornar “erro”
    - ▶ Se  $b$  bytes no final não são iguais a  $b$ , retornar “erro”
    - ▶ Caso contrário, remova os  $b$  bytes finais dos dados codificados e mostre o que restou da mensagem



## Exemplo com $L = 8$



# Injeção



Oráculo de padding  
verifica se os bytes  
finais estão corretos

# Oráculos de padding

- ▶ Oráculos de paddings podem ser encontrado em aplicações Web
- ▶ Mesmo se um erro não é explicitamente retornado, um atacante pode ser capaz de detectar diferenças no tempo de processamento, comportamento de CPU e entre outros

## Ideia principal do ataque

- ▶ Sendo um  $IV$  e um bloco de texto cifrado  $c$ 
  - ▶ Texto em claro com PKCS =  $F_k^{-1}(c) \oplus IV$
- ▶ Observação principal: se um atacante modificar o  $i$ -ésimo byte de  $IV$ , isto causa uma mudança predizível somente para o  $i$ -ésimo byte dos dados codificados

## Cenário inicial do ataque

$F_k^{-1}(c)$ : 

XX	XX	XX	XX	XX	XX	XX	XX
----	----	----	----	----	----	----	----



$IV$ : 

AB	01	4F	21	00	7C	02	9E
----	----	----	----	----	----	----	----



Texto em  
claro  
com  
PKCS#5:

XX	XX	XX	XX	XX	XX	XX	XX
----	----	----	----	----	----	----	----

- ▶  $k$  é desconhecido pelo atacante
- ▶  $IV$  e texto cifrado são conhecidos pelo atacante
- ▶ Atacante quer descobrir texto em claro.

$$F_k^{-1}(c): \quad \boxed{XX} \boxed{XX} \boxed{XX} \boxed{XX} \boxed{XX} \boxed{XX} \boxed{XX} \boxed{XX}$$



$$IV: \quad \boxed{\phantom{00}} \boxed{\phantom{00}} \boxed{\phantom{00}} \boxed{21} \boxed{00} \boxed{7C} \boxed{02} \boxed{9E}$$



Texto em  
claro  
com  
PKCS#5:

$$\boxed{\phantom{00}} \boxed{\phantom{00}} \boxed{\phantom{00}} \boxed{06} \boxed{06} \boxed{06} \boxed{06} \boxed{06}$$

- ▶ Atacante modifica IV até obter erro de padding

$F_k^{-1}(c)$ : 

XX	XX	XX	XX	XX	XX	XX	XX
----	----	----	----	----	----	----	----



IV: 

AB	41	4E	20	01	7D	03	9F
----	----	----	----	----	----	----	----



Texto em  
PKCS#5:

XX	07	07	07	07	07	07
----	----	----	----	----	----	----

$$IV' = IV \oplus 41$$

$$07 = F_k^{-1}(c) \oplus IV \oplus 41$$

$$07 \oplus 41 \oplus 01 = F_k^{-1}(c)$$

- ▶ Atacante modifica bytes à direita em IV para obter padding igual a 07
  - ▶  $0x9E \oplus 0x06 \oplus 0x07 = 0x9F$
- ▶ Atacante faz tentativas para aprender cada byte que não está no padding, um por vez.

## Complexidade do ataque?

- ▶  $\leq L$  tentativas para aprender o número # de bytes de padding
- ▶  $\leq 2^8 = 256$  tentativas para aprender cada byte de texto em claro



# Resumo

- ▶ Ataques de texto-cifrado escolhido representam uma ameaça significativa e real
- ▶ Encriptação moderna deve ser projetada para ser segura contra CCA
- ▶ Veremos exemplo de um esquema seguro contra CCA