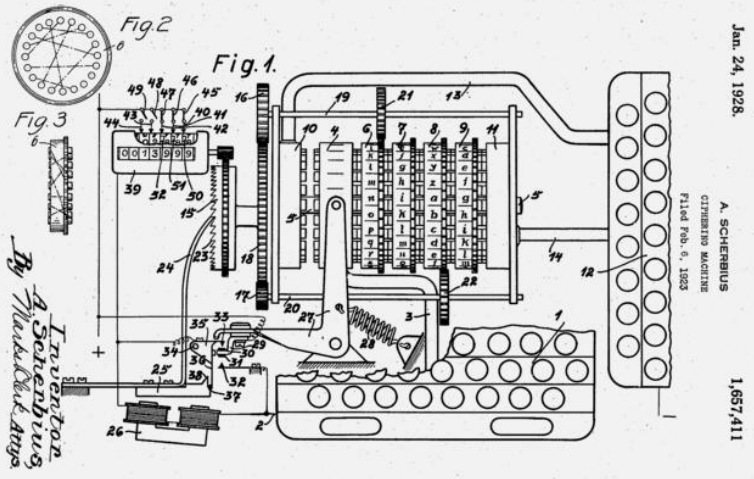
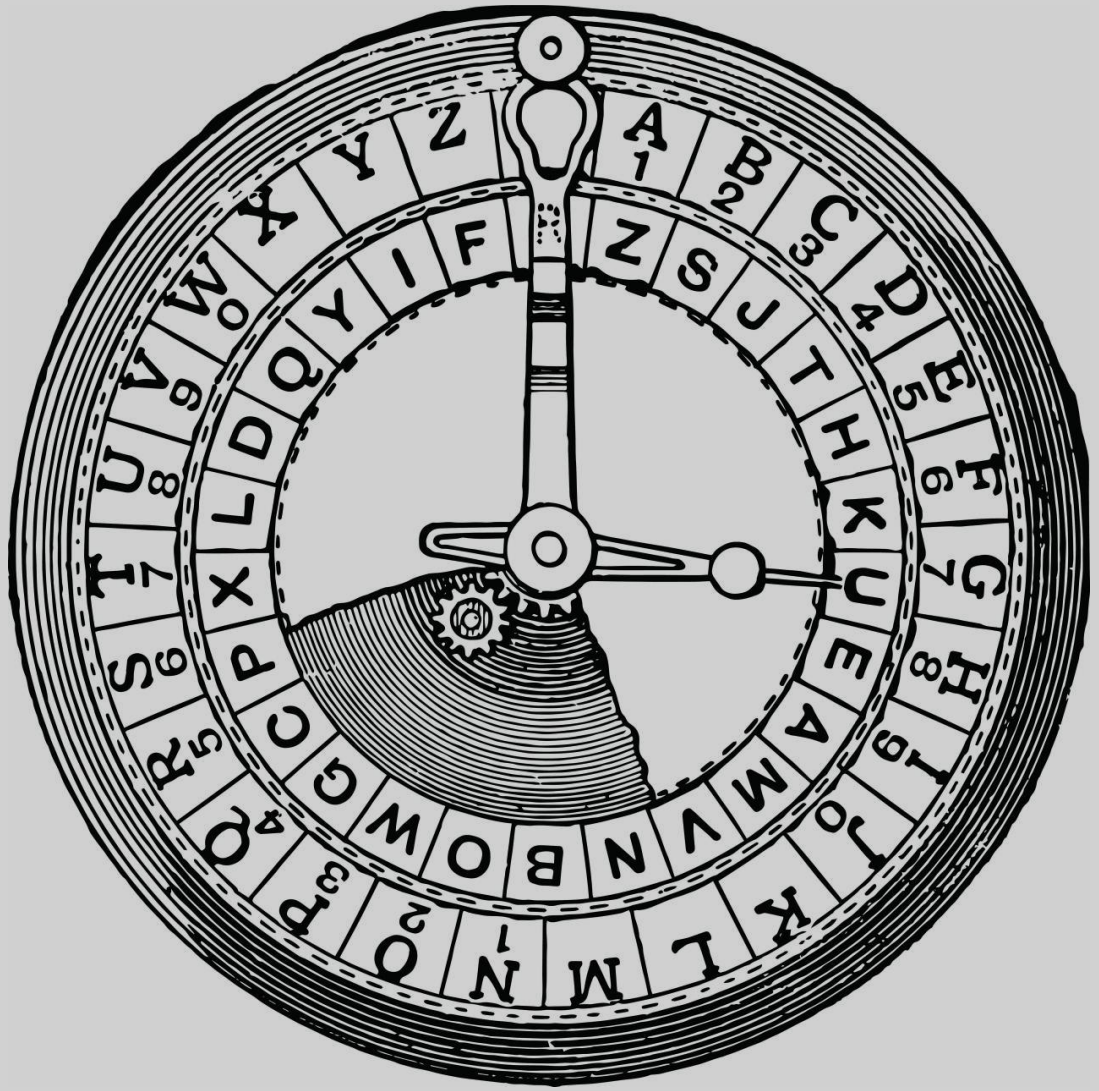
Segurança CCA

Fundamentos da Criptografia

Nicollas Mocelin Sdroievski



Nas últimas aulas



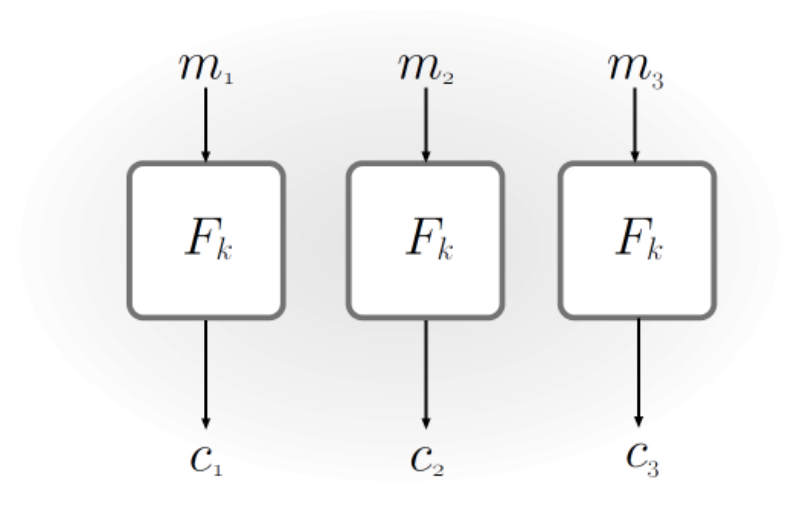
Permutações pseudoaleatórias

- Ou cifras de bloco
- Funções chaveadas $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ tal que
  - $F_k(\cdot) = F(k,\cdot)$ é sempre uma permutação
  - Tanto F_k quanto F_k^{-1} são computáveis em tempo polinomial
  - Indistinguíveis de permutações aleatórias: existe uma função negligenciável...

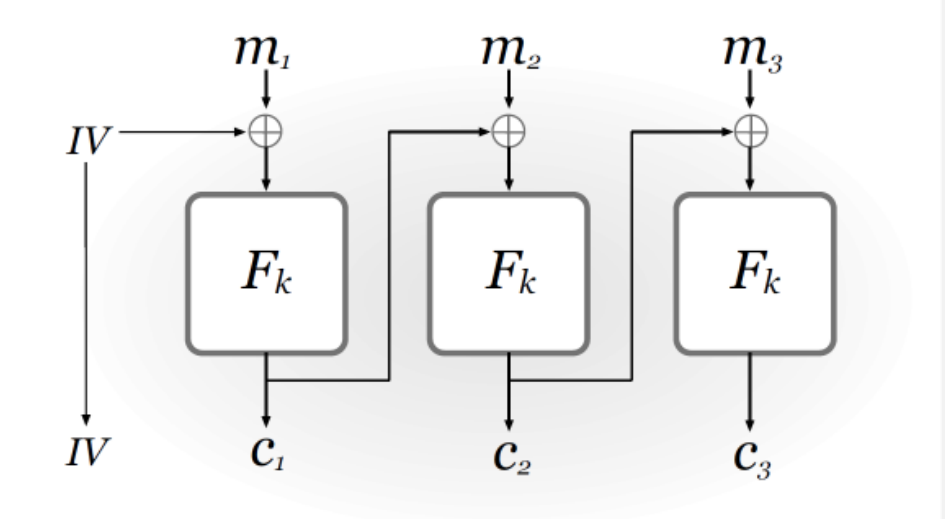
$$\left|\Pr\left[D^{F_k(\cdot)}(1^n) = 1\right] - \Pr\left[D^{f(\cdot)}(1^n) = 1\right]\right| \leq \mathrm{negl}(n)$$

Modos de operação

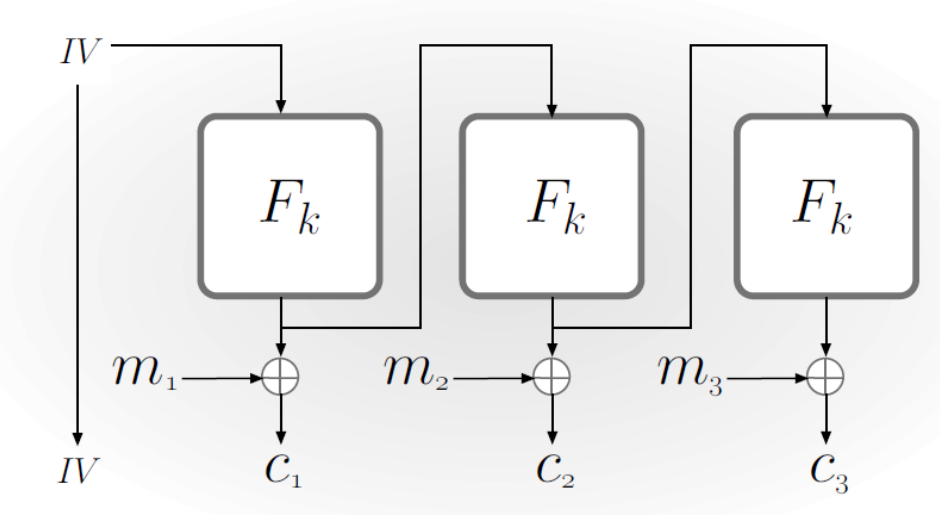
ECB



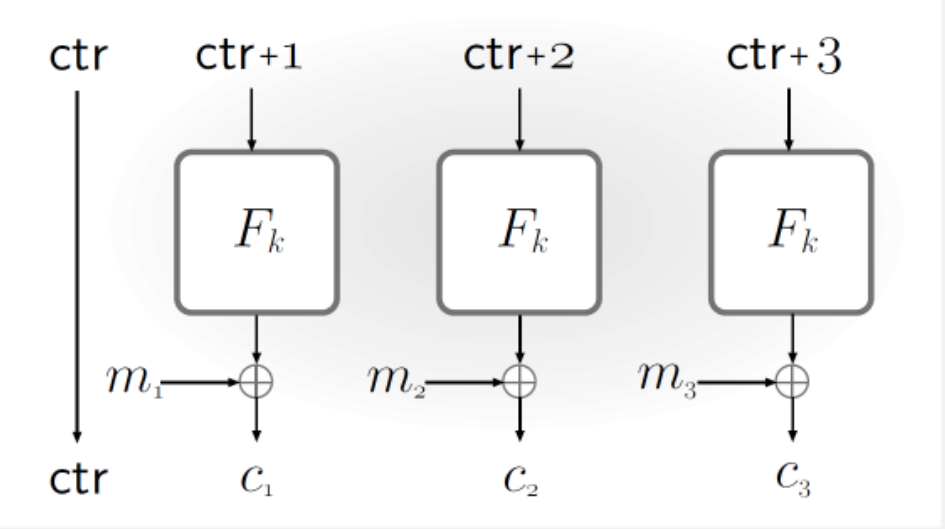
CBC



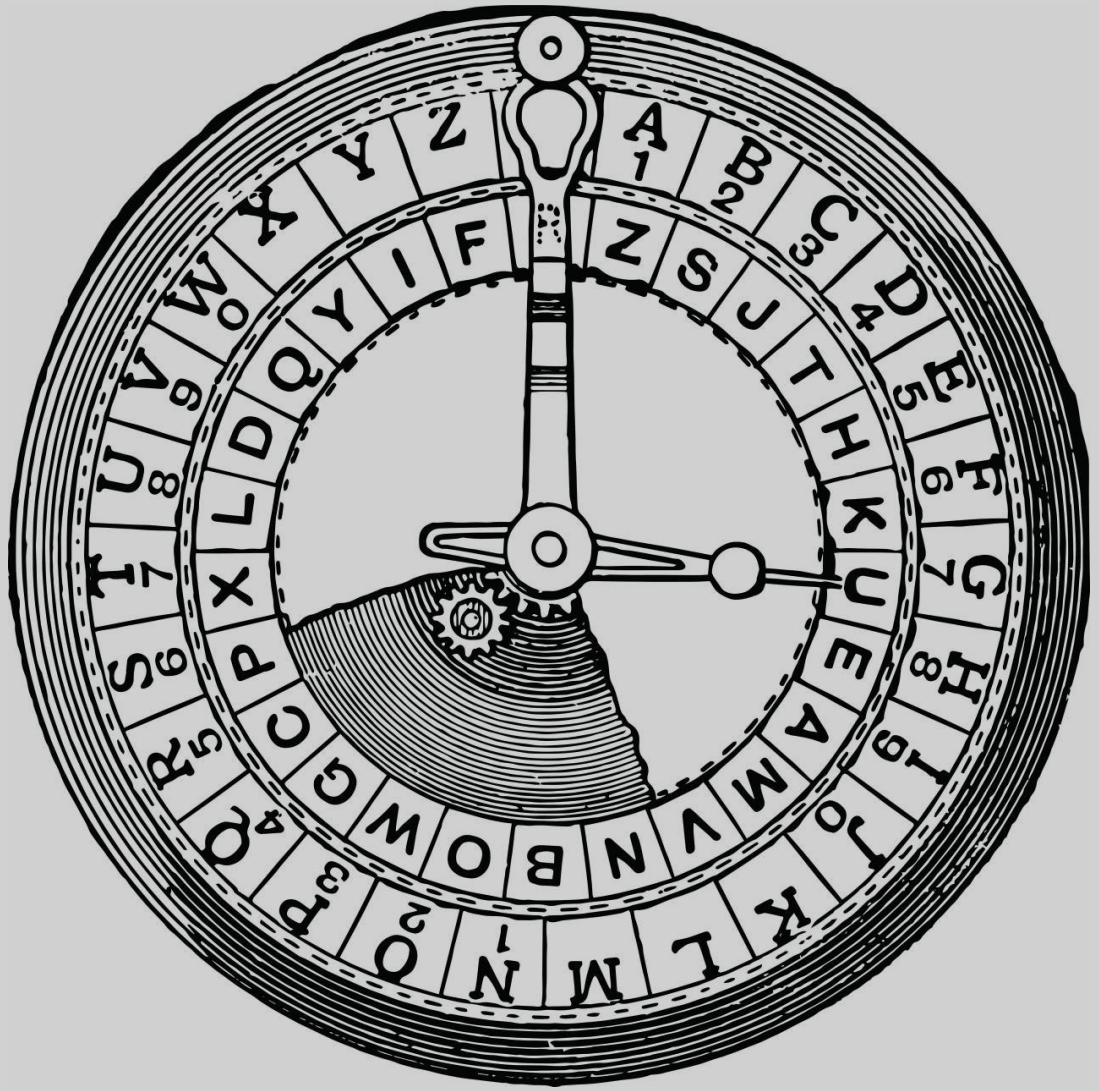
OFB



CTR



Segurança CCA



Segurança CCA

- Até então, dois tipos de ataques
  - O adversário apenas enxerga o texto cifrado (EAV)
  - O adversário pode influenciar as mensagens que são cifradas (CPA)
- Agora, um ataque ainda mais forte
  - O adversário ainda pode influenciar nas mensagens que são cifradas (CPA)
  - Porém, também pode influenciar nas mensagens que são decifradas!
  - Formalmente, o adversário possui um oráculo de **decifragem** (além do oráculo de cifragem)

O experimento $\mathrm{PrivK}^{\mathrm{cca}}_{\mathcal{A},\Pi}(n)$

Desafiante



Adversário $\mathcal{A}$
Tempo poli(n)



$k \leftarrow \mathrm{Gen}(1^n)$

$m \in \mathcal{M}$ **OU** $c' \in \mathcal{C}$

$\mathrm{Enc}_k(m)$ **OU** $\mathrm{Dec}_k(c')$ ⟳

$m_0, m_1 \in \mathcal{M}$ com $|m_0| = |m_1|$

$b \leftarrow \{0,1\}$

$c \leftarrow \mathrm{Enc}_k(m_b)$

$m \in \mathcal{M}$ **OU** $c' \in \mathcal{C}$ (não vale $c = c'$)

$\mathrm{Enc}_k(m)$ **OU** $\mathrm{Dec}_k(c)$ ⟳

$b' =$ chute para b

O experimento $\mathrm{PrivK}^{\mathrm{cca}}_{\mathcal{A},\Pi}(n)$

- Como antes, a saída do experimento é 1 se $b' = b$ e 0 caso contrário
- Escrevemos $\mathrm{PrivK}^{\mathrm{cca}}_{\mathcal{A},\Pi}(n) = 1$ se a saída do experimento é 1
- **Definição 3.22.** Uma cifra de chave privada $\Pi = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ possui cifragens indistinguíveis sob ataque de texto cifrado conhecido, ou é CCA-segura, se para todos os algoritmos probabilísticos polinomiais $\mathcal{A}$ existe uma função negligenciável negl tal que,

$$\Pr\left[\mathrm{PrivK}^{\mathrm{cca}}_{\mathcal{A},\Pi}(n) = 1\right] \leq \frac{1}{2} + \mathrm{negl}(n),$$

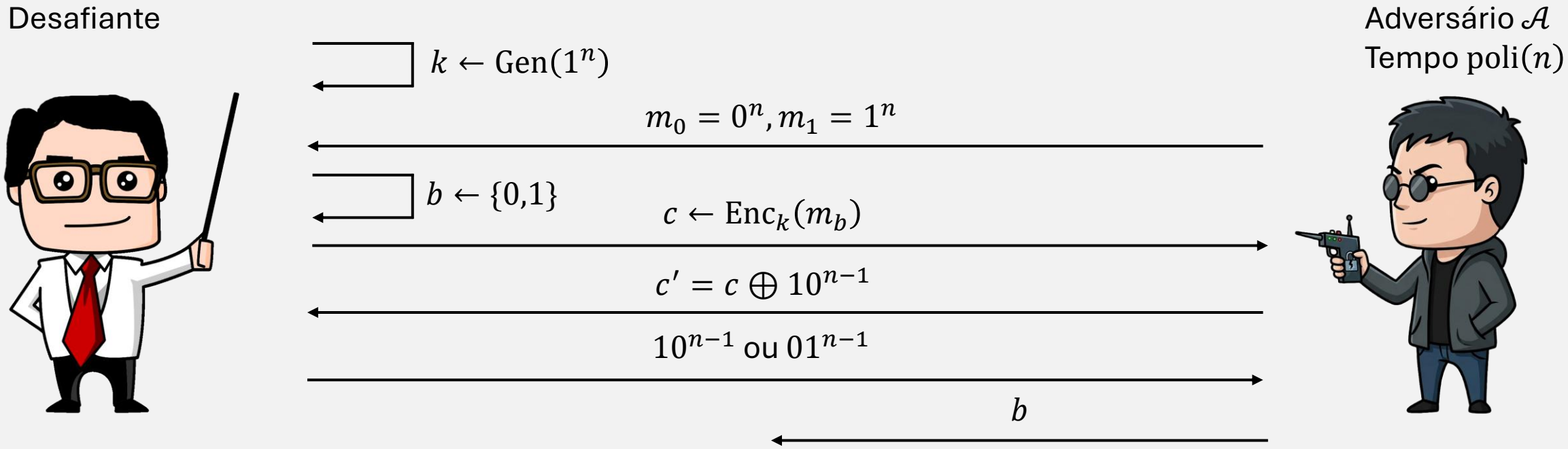
onde as probabilidades são sobre a aleatoriedade de $\mathcal{A}$ e do Desafiante (para escolher a chave, o bit b, e de Enc , caso necessário)

Segurança CCA é muito forte?

- Nenhum dos esquemas que vimos até agora é CCA seguro
  - A construção 3.30 é “quase” CCA segura, porém (exercício)
- Esses ataques fazem sentido no mundo real?
- No geral, não esperamos de fato que as partes honestas decifrem qualquer texto cifrado enviado pelos adversários
  - Midway
  - Textos cifrados para mensagens mal formadas
    - Padding-Oracle Attack
  - Cifragem como uma etapa de um protocolo mais complexo

Um ataque CCA

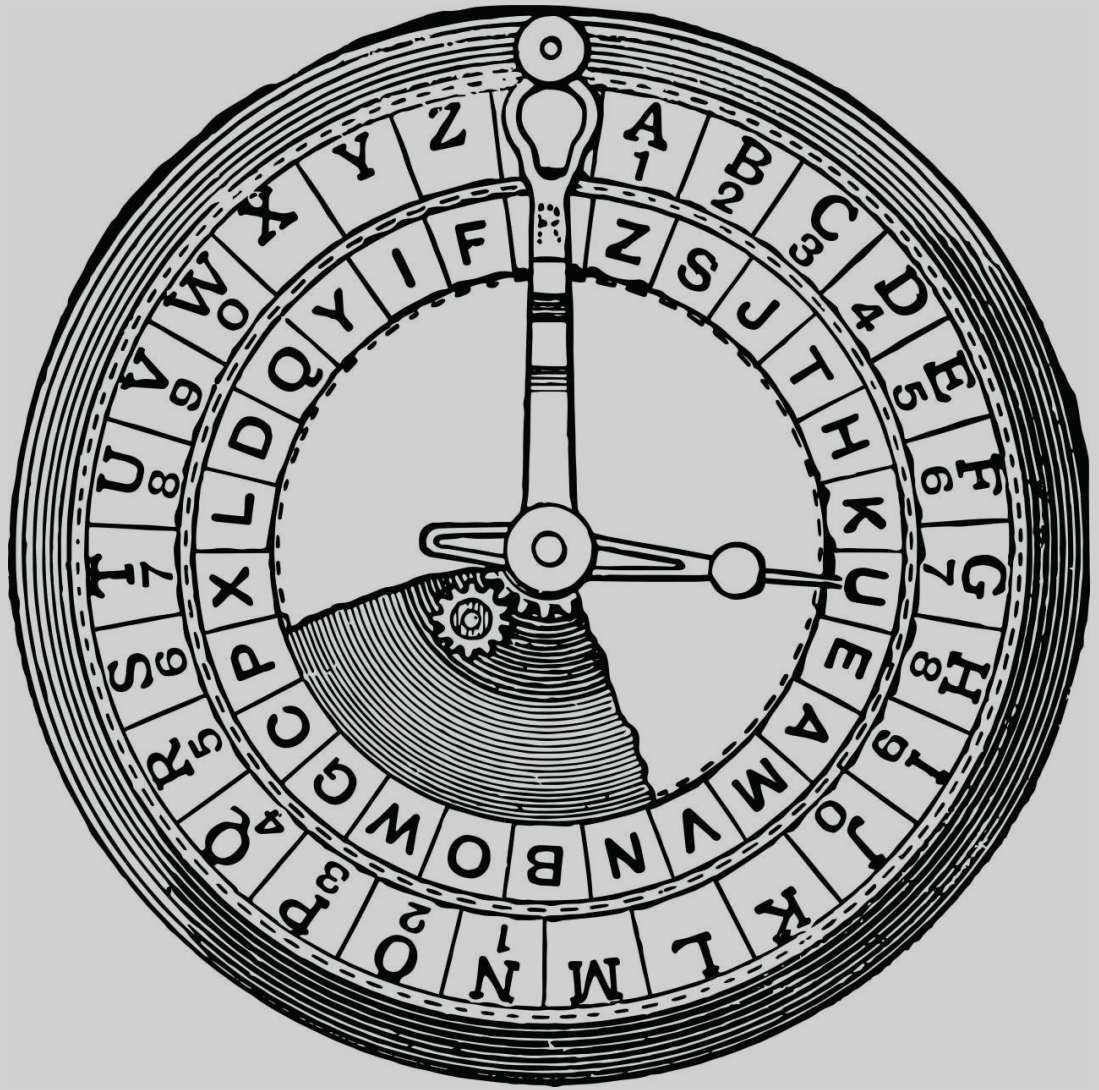
- Construção 3.30
  - $\mathrm{Enc}_k(m) = \langle r, F_k(r) \oplus m \rangle$
  - Ataque CCA?



Não-maleabilidade

- Os textos cifrados da construção 3.30 (e de todas que vimos até agora) são **maleáveis**
- Modificações em um texto cifrado c têm efeitos previsíveis
  - E em particular geram outro texto cifrado c' válido
- Todas as cifras CCA seguras são também **não-maleáveis**
- Pequeno spoiler:
  - Na construção CCA segura que veremos, o adversário só vai conseguir "usar" o oráculo de decifragem com probabilidade negligenciável

Ataques do tipo Padding-Oracle

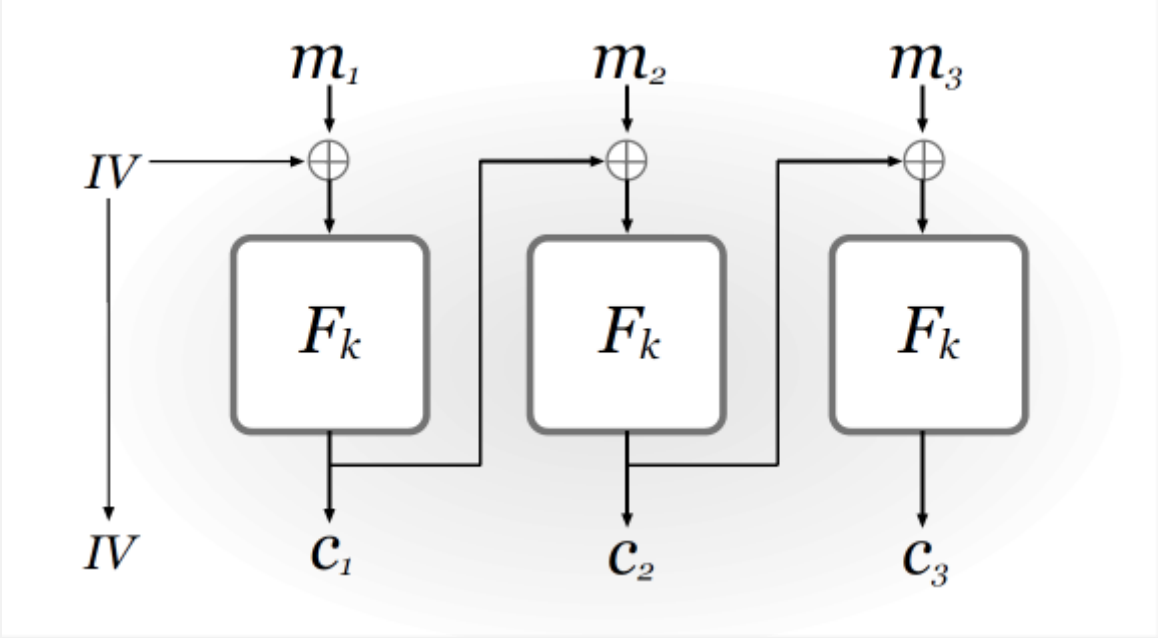


Ataque Padding-Oracle

- Veremos um ataque “no estilo CCA”
  - Que se aplica ao modo CBC com um esquema de padding (preenchimento) específico
- Porém o ataque nem precisa que as partes honestas de fato decifrem qualquer texto cifrado
- Basta que as partes honestas indiquem se o texto cifrado decifra para uma mensagem válida ou não
- Mensagem válida = quantidade de padding correta

Relembrando CBC

- Para cifrar $(m_1, \dots, m_\ell)$
  - $c_0 \coloneqq IV$ (uma string aleatória de tamanho n)
  - $c_1 \coloneqq F_k(c_0 \oplus m_1)$
  - $c_2 \coloneqq F_k(c_1 \oplus m_2)$
  - $\vdots$
  - $c_i \coloneqq F_k(c_{i-1} \oplus m_i)$
- Para decifrar $(c_0, c_1, \dots, c_\ell)$
  - $m_i \coloneqq F_k^{-1}(c_i) \oplus c_{i-1}$



Padding PKCS #5

- Assuma que a mensagem original m possui um número inteiro de bytes
- E que L é o tamanho de bloco da cifra de bloco subjacente
- Seja b o número de bytes que devem ser adicionados ao fim da mensagem para que tenha tamanho múltiplo de L
  - $1 \leq b \leq L$
  - $b = 0$ não é permitido (ambíguo)
- O padding é a própria string b (em hexadecimal) repetida b vezes

Exemplos

- Se é necessário apenas um byte
  - Adiciona-se 0x01 ao fim da string
- Se são necessários quatro bytes
  - Adiciona-se 0x04040404 ao fim da string
- ⋮

Processo de decifragem

- A parte honesta/servidor recebe $c = (IV, c_1, \ldots, c_\ell)$ e então
  - Decifra c em $(m_1, \ldots, m_\ell)$
  - Verifica se m_ℓ possui o padding correto
  - Se sim, retorna uma mensagem: “valid message”
  - Se não, retorna um erro: “bad padding”
- Na prática:
  - Qualquer diferença nos dois casos que seja detectável
    - Tipos de erro
    - Tempo

Ideia inicial

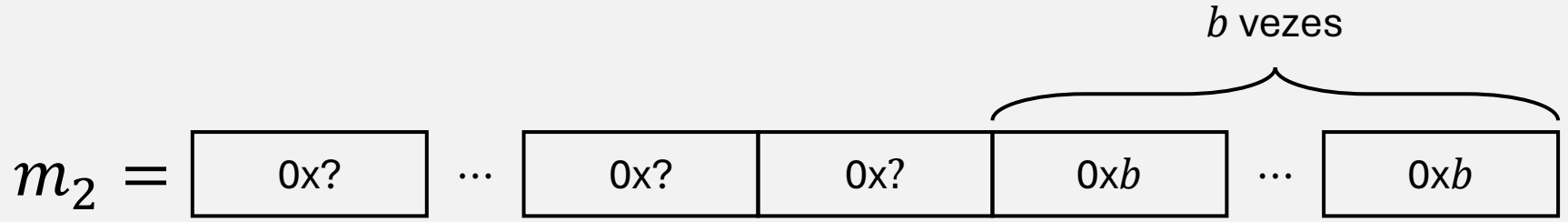
- Considere um texto cifrado (IV, c_1, c_2)
  - E sejam m_1, m_2 as mensagens originais
  - Lembre que cada bloco possui L bytes
- Porque estamos usando CBC
  - $m_2 = F_k^{-1}(c_2) \oplus c_1$
  - Onde k é a chave (desconhecida)
- Sabemos que m_2 termina em 0xb ⋯0xb (b vezes)
- Propriedade chave: mudanças no texto cifrado resultam em mudanças previsíveis nas mensagens

Maleabilidade

- Considere c_1' idêntico à c_1 exceto pelo último byte
- Decifrando (IV, c_1', c_2), obtemos m_1', m_2', onde
  - $m_2' = F_k^{-1}(c_2) \oplus c_1'$
  - m_2' é idêntico à m_2, exceto pelo último byte
  - m_1' é completamente imprevisível, mas não importante
- De maneira semelhante, se c_1' é idêntico à c_1 exceto pelo i-ésimo byte
  - m_2' é idêntico à m_2, exceto pelo i-ésimo byte
- De maneira geral, se $c_1' = c_1 \oplus \Delta$
  - $m_2' = m_2 \oplus \Delta$

Aprendendo b

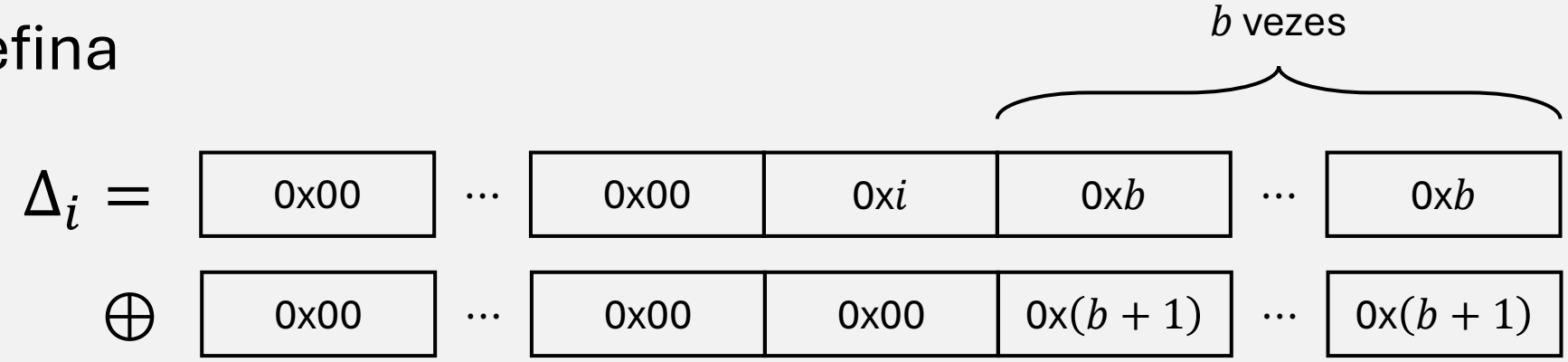
- Vamos usar essa ideia para aprender b (a quantidade de padding)



- Primeiro, o adversário modifica o primeiro byte de c_1 e envia o resultado (IV, c_1', c_2)
  - Lembre que isso modifica o primeiro byte de m_2
- O que acontece se houver erro de decifragem?
  - b deve ser igual a L
- Se não houver erro?
  - Continua com o segundo byte, terceiro byte, ...
  - O primeiro que levar à um erro revela o valor de b

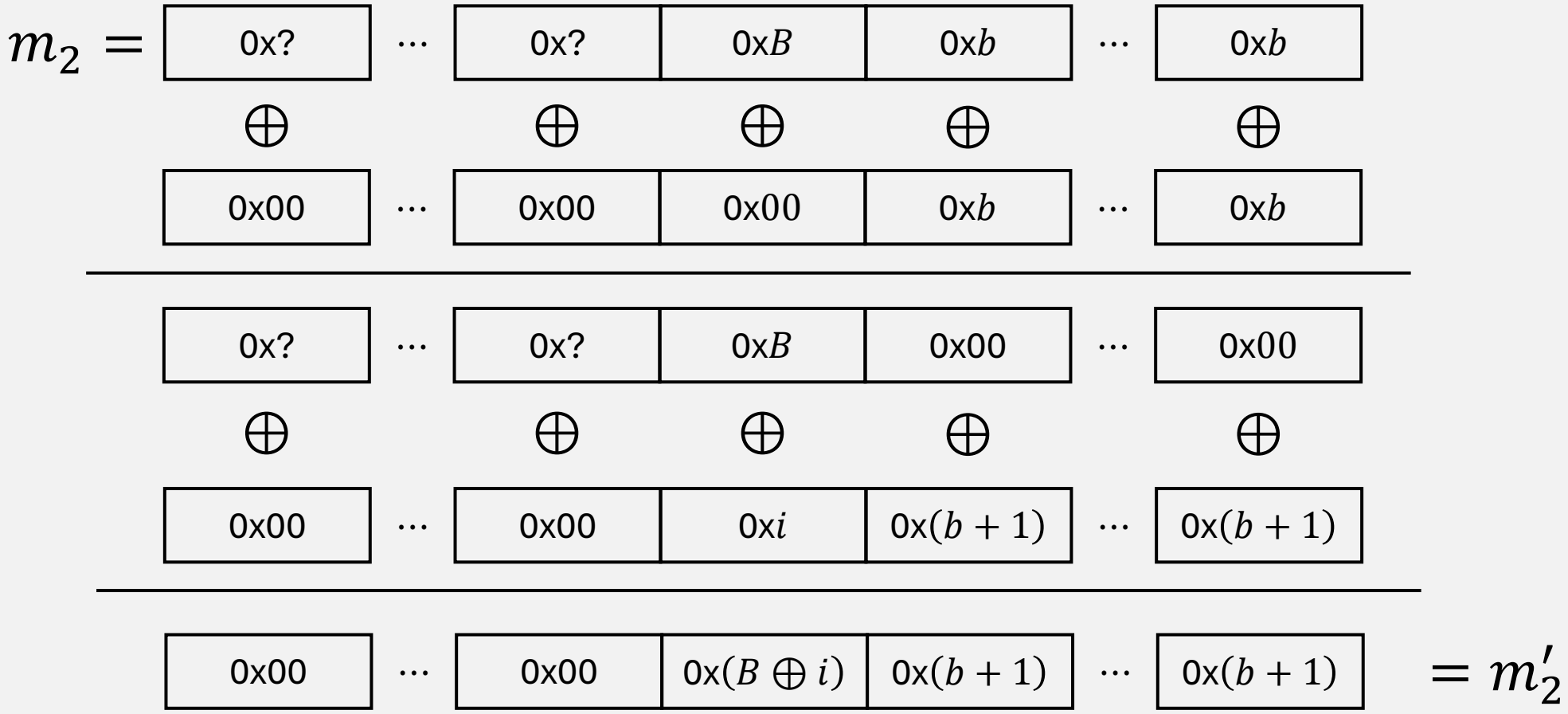
Aprendendo a mensagem

- Agora que sabemos b, vamos revelar os bytes da mensagem um-por-um
- Agora o adversário sabe que m_2 termina em 0xB0xb ⋯ 0xb (com 0xb repetido b vezes), e quer descobrir B
- Defina

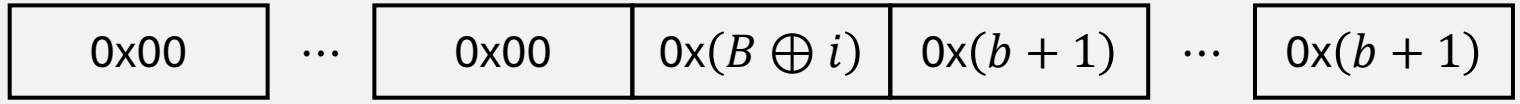


Para $0 \leq i < 2^8 = 256$

O valor de m_2'


$m_2 =$
0x?
⋯
0x?
0xB
0xb
⋯
0xb
⊕
0x00
⋯
0x00
0x00
0xb
⋯
0xb
0x?
⋯
0x?
0xB
0x00
⋯
0x00
⊕
0x00
⋯
0x00
0x00
0xi
0x$(b+1)$
⋯
0x$(b+1)$
0x00
⋯
0x00
0x$(B \oplus i)$
0x$(b+1)$
⋯
0x$(b+1)$
$= m_2'$

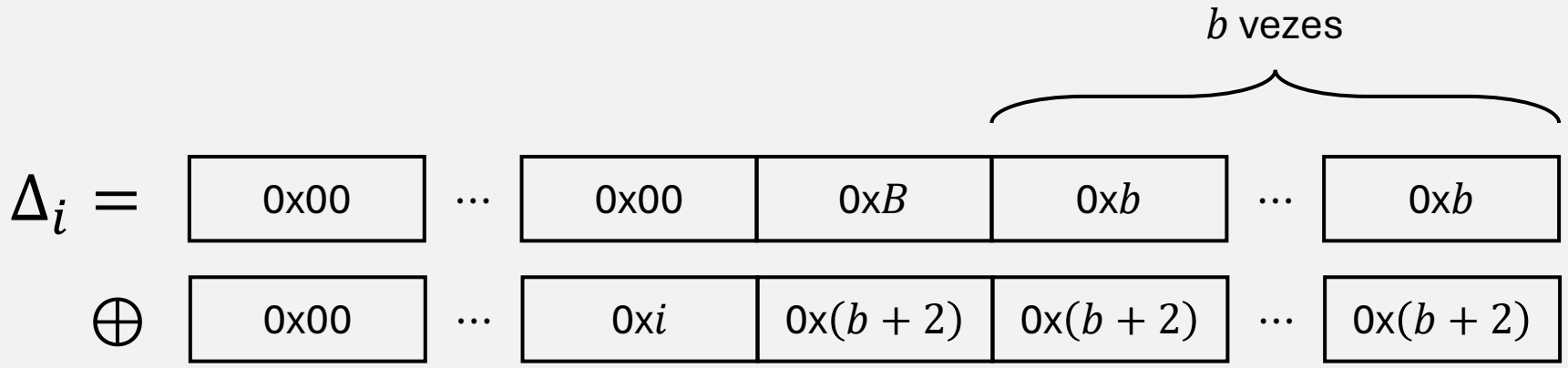
O valor de m_2'

$m_2' =$ 

- Quando que m_2' possui padding correto?
  - Se e somente se $0x(B \oplus i) = 0x(b+1)$
  - Nesse caso, podemos recuperar $B = 0x\big((b+1) \oplus i\big)$
- Testando todos os valores de i (256 deles), recuperamos o último byte de m_2

Aprendendo o penúltimo byte

- Como aprender o penúltimo byte?

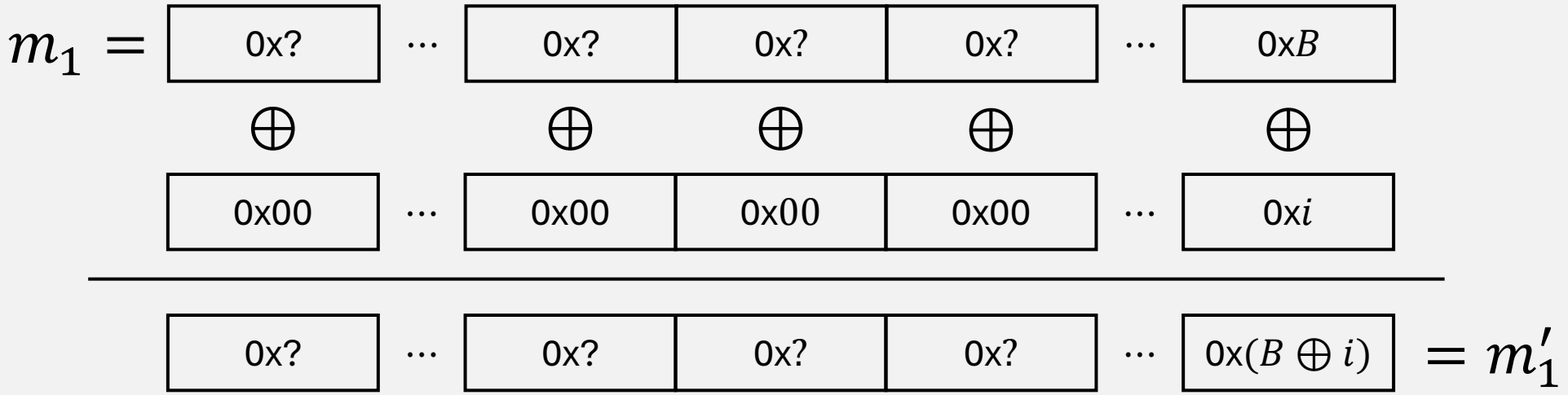


- E assim por diante

Outros blocos da mensagem

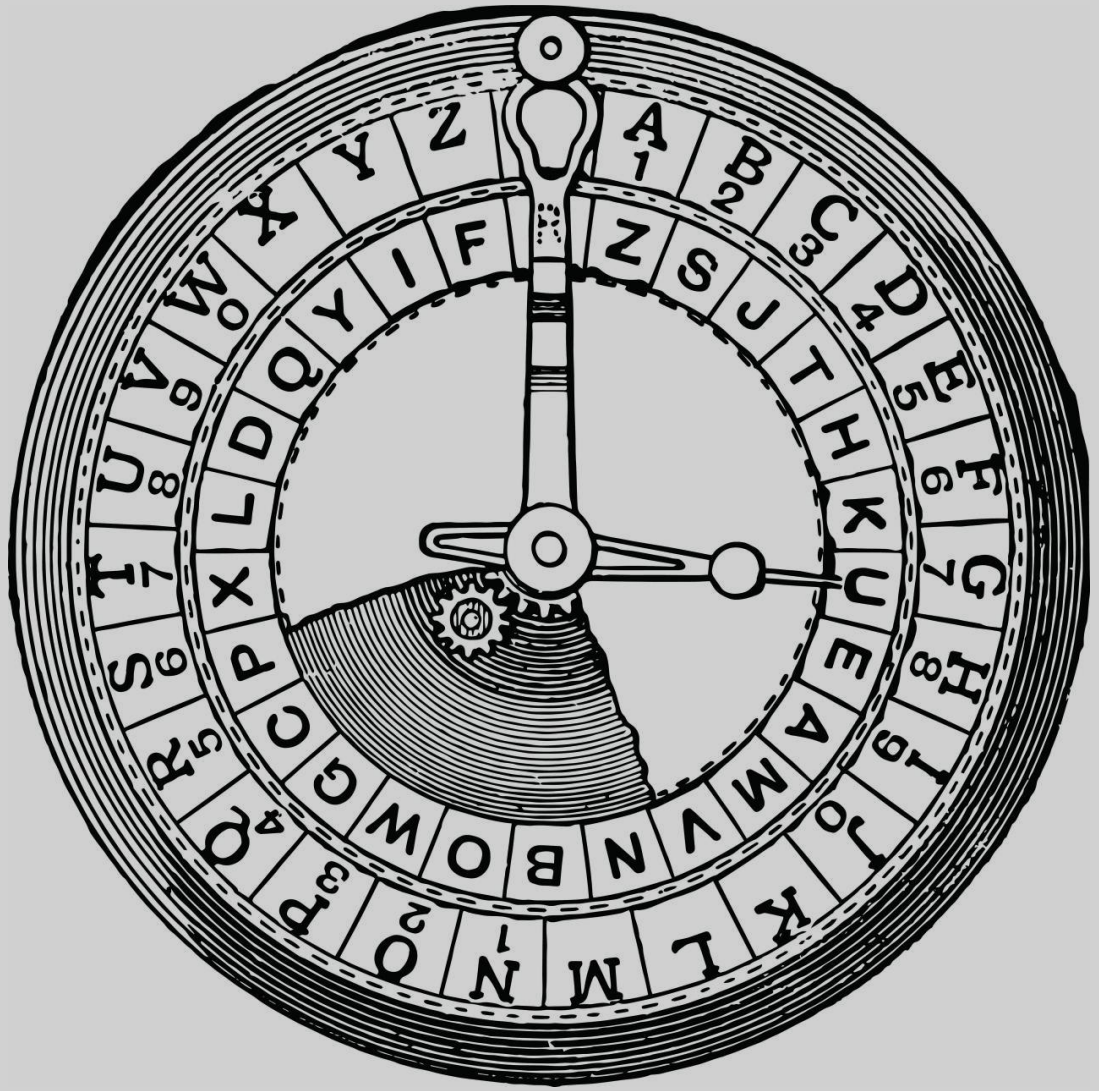
- Agora conseguimos recuperar o último bloco inteiro, m_2
- Como fazemos para recuperar todos os blocos da mensagem?
- De certa forma, o trabalho ficou um pouco mais fácil!
  - Por que?
  - Porque agora sabemos que os outros blocos não possuem padding
- Vamos tratar (IV, c_1) como um "texto cifrado" para m_1
- E modificar IV (byte a byte) para recuperar m_1

Recuperando m_1



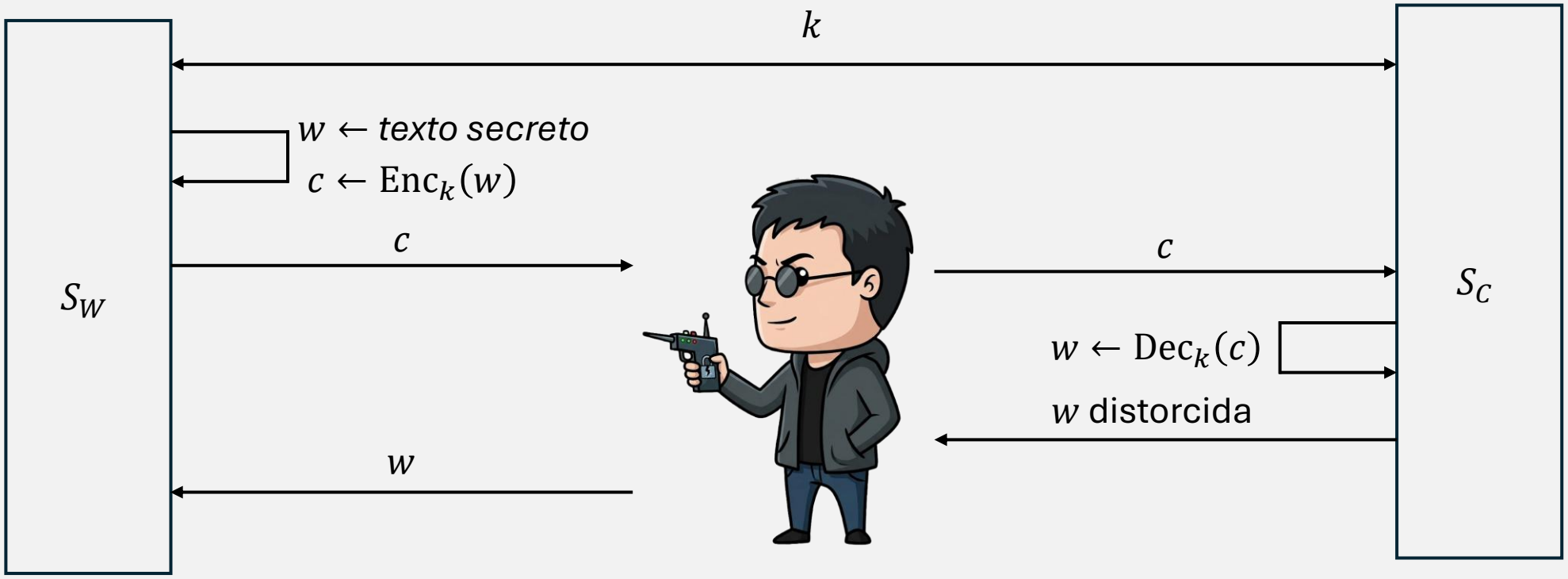
- Testando todos os valores de i, encontramos o último byte
- E então prosseguimos como antes
- Uma dificuldade: ambiguidade

CAPTCHAs



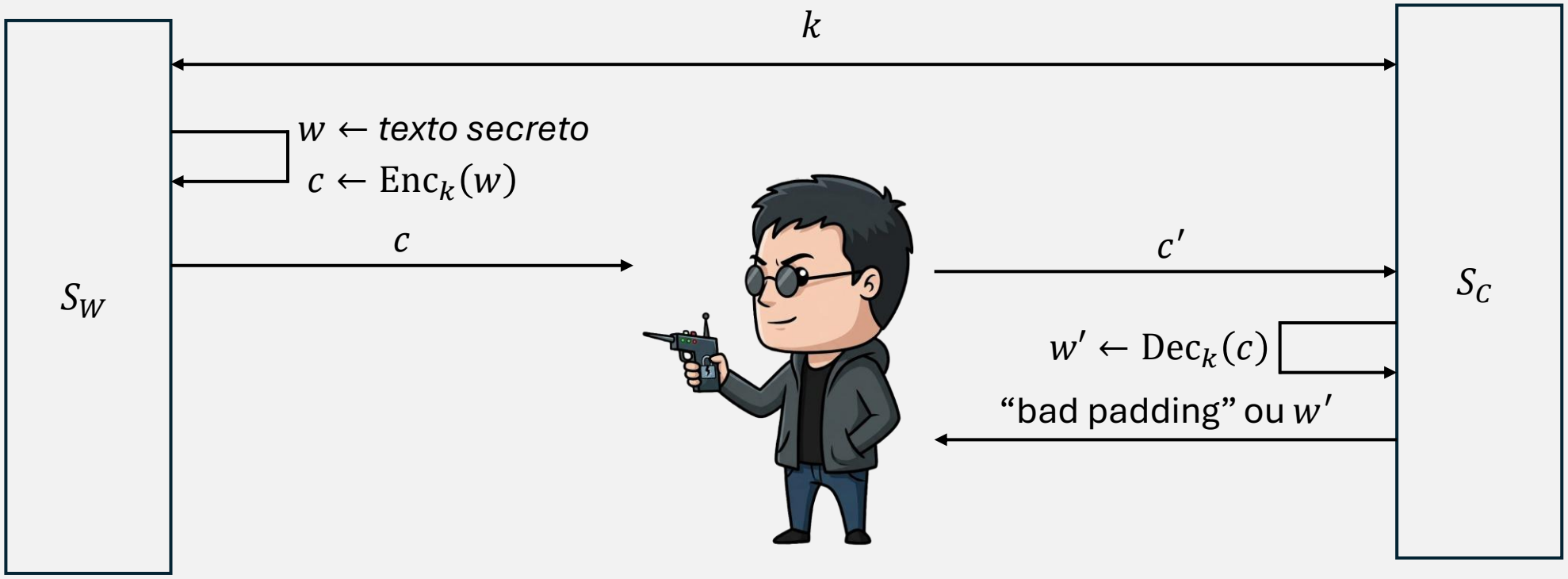
Uma maneira de implementar CAPTCHAs

- Um usuário $\mathcal{A}$ interage com um servidor S_W, que usa um serviço externo S_C para gerar os CAPTCHAs

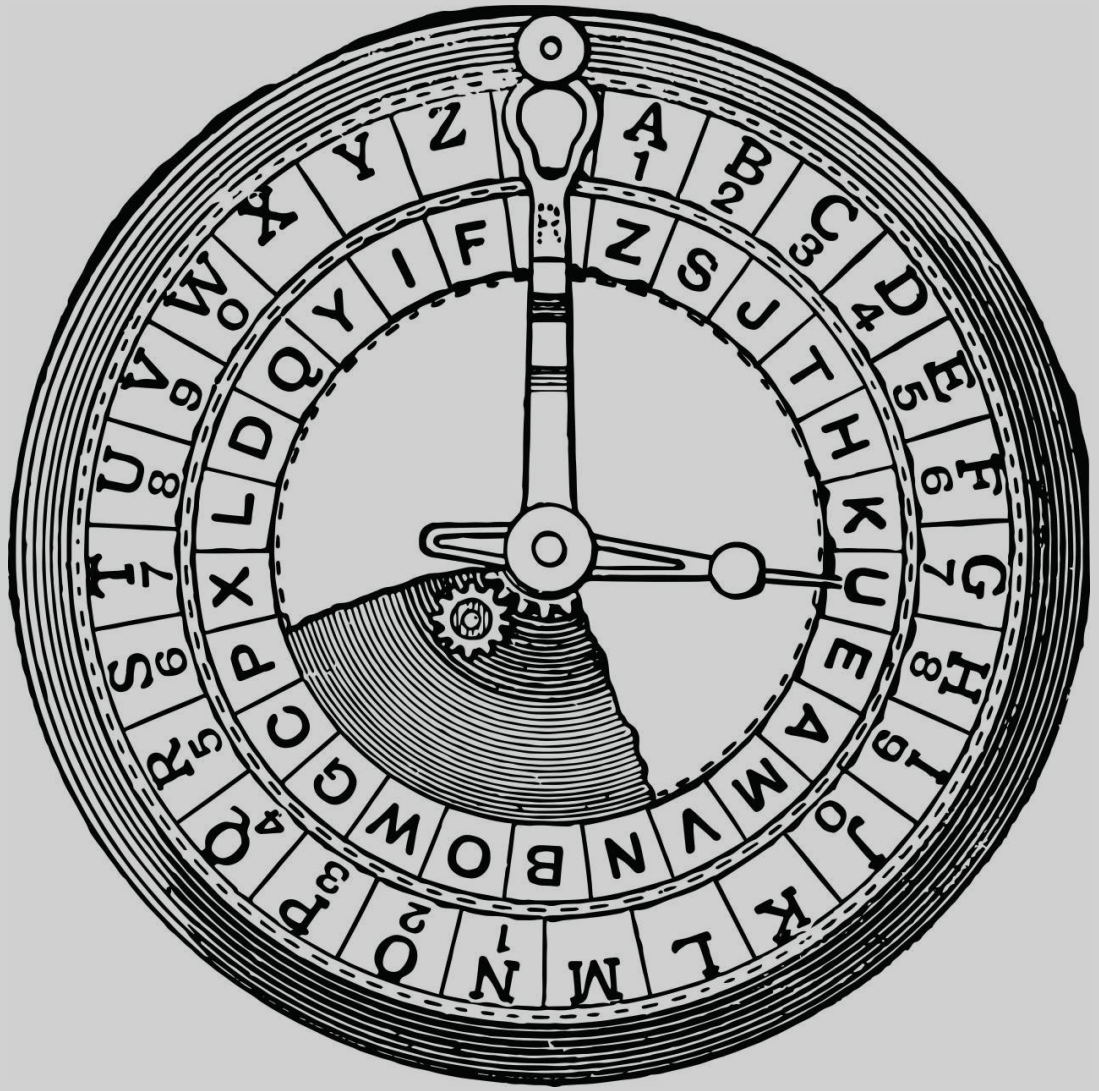


Como realizar o ataque?

- $\mathcal{A}$ pode mandar o texto cifrado c' que quiser para S_C
- E em particular implementar um ataque paddding-oracle



Para fechar



Para fechar

- Hoje
  - Segurança CCA
    - Adversário tem acesso à oráculos de cifragem e decifragem
    - Não vale usar o oráculo na mensagem de desafio
  - Ataques do tipo padding-oracle
    - Usam um oráculo parcial de decifragem para recuperar a mensagem original
- Próxima aula
  - Códigos de autenticação de mensagem
    - Foi você mesmo que mandou a mensagem?
    - Ela não foi modificada no caminho?