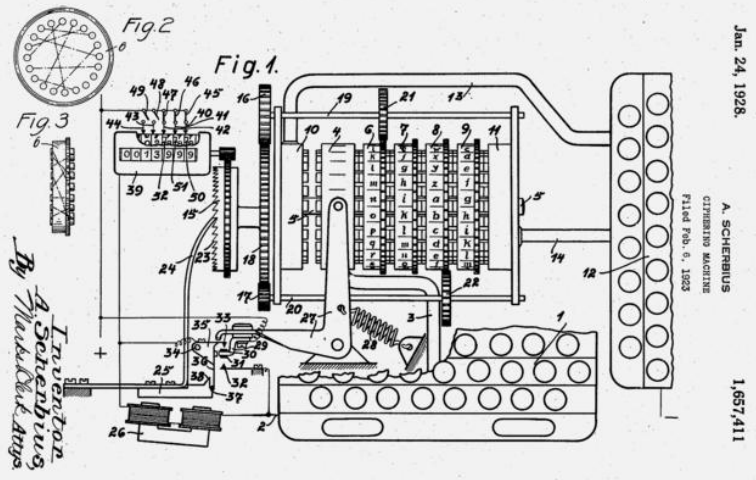
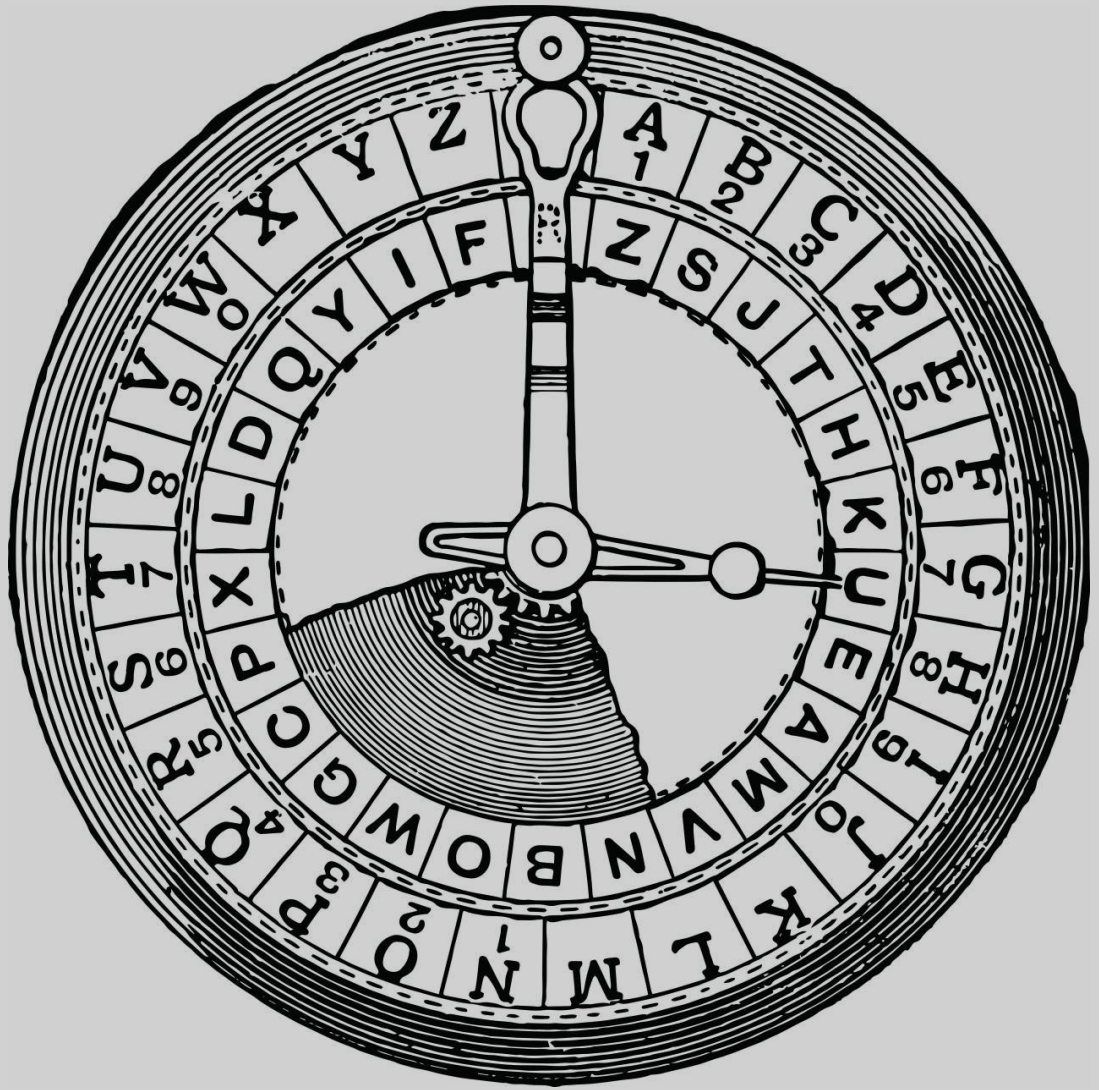
Cifras Autenticadas

Fundamentos da Criptografia

Nicollas Mocelin Sdroievski



Na última aula



Sintaxe de MACs

- Digamos que quero enviar uma mensagem m
  - E tenho uma chave privada $k \leftarrow \mathrm{Gen}(1^n)$
- Primeiro, computo uma **tag** t a partir de m e k
  - $t \leftarrow \mathrm{Mac}_k(m)$
- E envio (m, t)
- Ao receber (m, t) a outra parte verifica se t é uma *tag* válida para m
  - $\mathrm{Vrfy}_k(m, t)$
- Deve valer que para todo n, toda chave k produzida por $\mathrm{Gen}(1^n)$ e $m \in \{0,1\}^*$, $\mathrm{Vrfy}_k\big(m, \mathrm{Mac}_k(m)\big) = 1$.

Experimento $\text{MAC-forge}_{\mathcal{A},\Pi}(n)$:

Desafiante



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



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

$m' \in \{0,1\}^*$

$\text{Mac}_k(m')$

(m, t) (não vale repetir m')

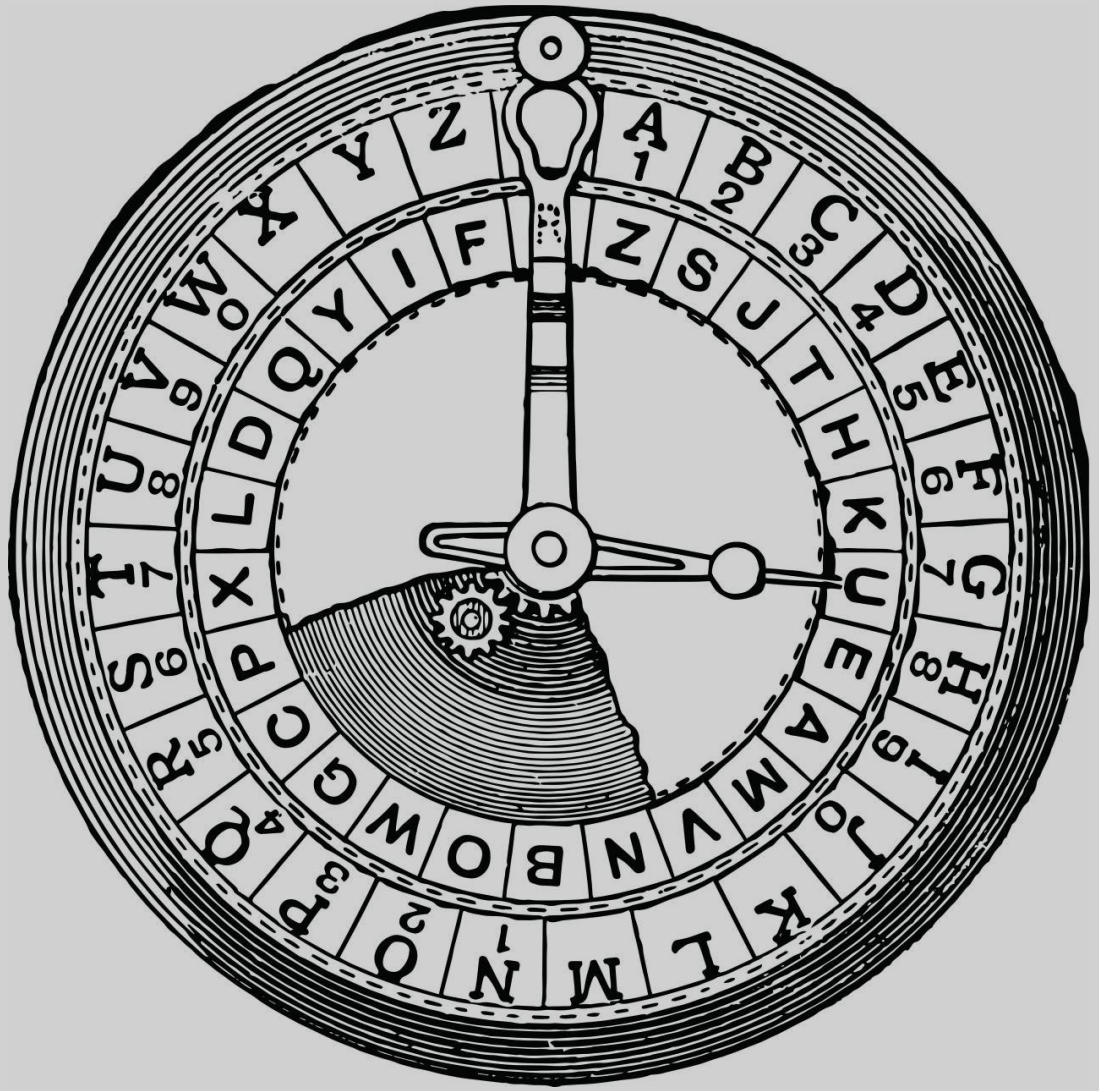
- Seja Q o conjunto de mensagens consultadas por $\mathcal{A}$
- $\mathcal{A}$ obtém sucesso se e somente se $m \notin Q$ e $\text{Vrfy}_k(m, t) = 1$
- Neste caso escrevemos $\text{MAC-forge}_{\mathcal{A},\Pi}(n) = 1$

O MAC

- **Construção 4.5.** Seja F uma função pseudoaleatória. Definimos um MAC para mensagens de tamanho n a seguir:
  - Mac: ao receber uma chave $k \in \{0,1\}^n$ e uma mensagem $m \in \{0,1\}^n$, produza a tag $t := F_k(m)$.
  - Vrfy: ao receber uma chave $k \in \{0,1\}^n$, uma mensagem $m \in \{0,1\}^n$ e uma tag $t \in \{0,1\}^n$, responda 1 se e somente se $t = F_k(m)$.

- **Teorema 4.6.** Se F é uma função pseudoaleatória, então a Construção 4.5 é um MAC seguro para mensagens de tamanho n.

Segredo + integridade



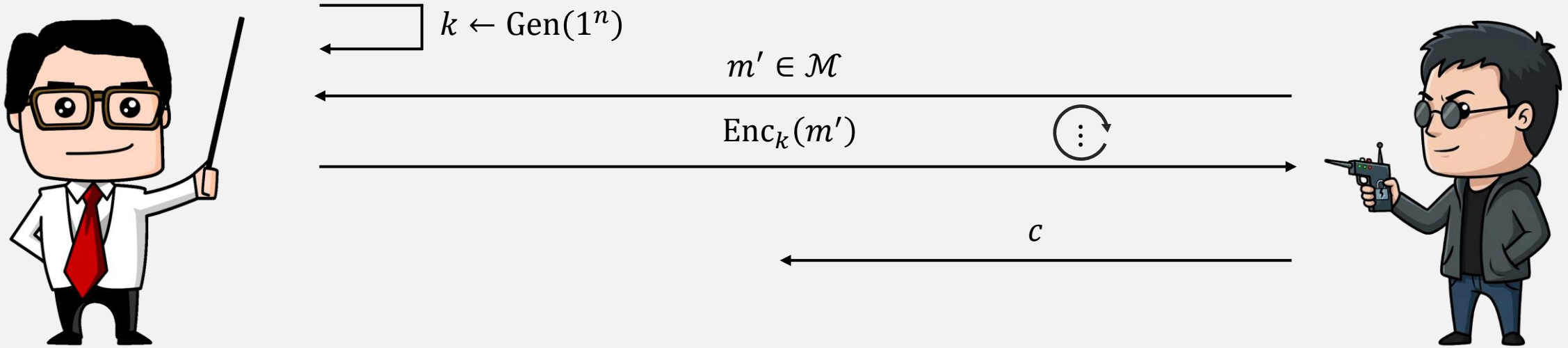
Ideia

- Seja $\Pi = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ uma cifra de chave privada
- Queremos que Π seja:
  - CCA-segura
    - Por que?
    - Porque já estamos assumindo que o adversário pode tentar modificar um texto cifrado
  - E “infalsificável”
    - Parecido com a definição de segurança de um MAC
    - Vamos definir através do experimento $\mathrm{Enc\text{-}Forge}_{\mathcal{A},\Pi}(n)$

Experimento $\text{Enc-Forge}_{\mathcal{A},\Pi}(n)$

Desafiante

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



- Sejam $m := \text{Dec}_k(c)$ e Q o conjunto de todas as consultas que $\mathcal{A}$ fez para o oráculo de cifragem
- A saída do experimento é 1 se e somente se
  - $m \neq \perp$
  - $m \notin Q$

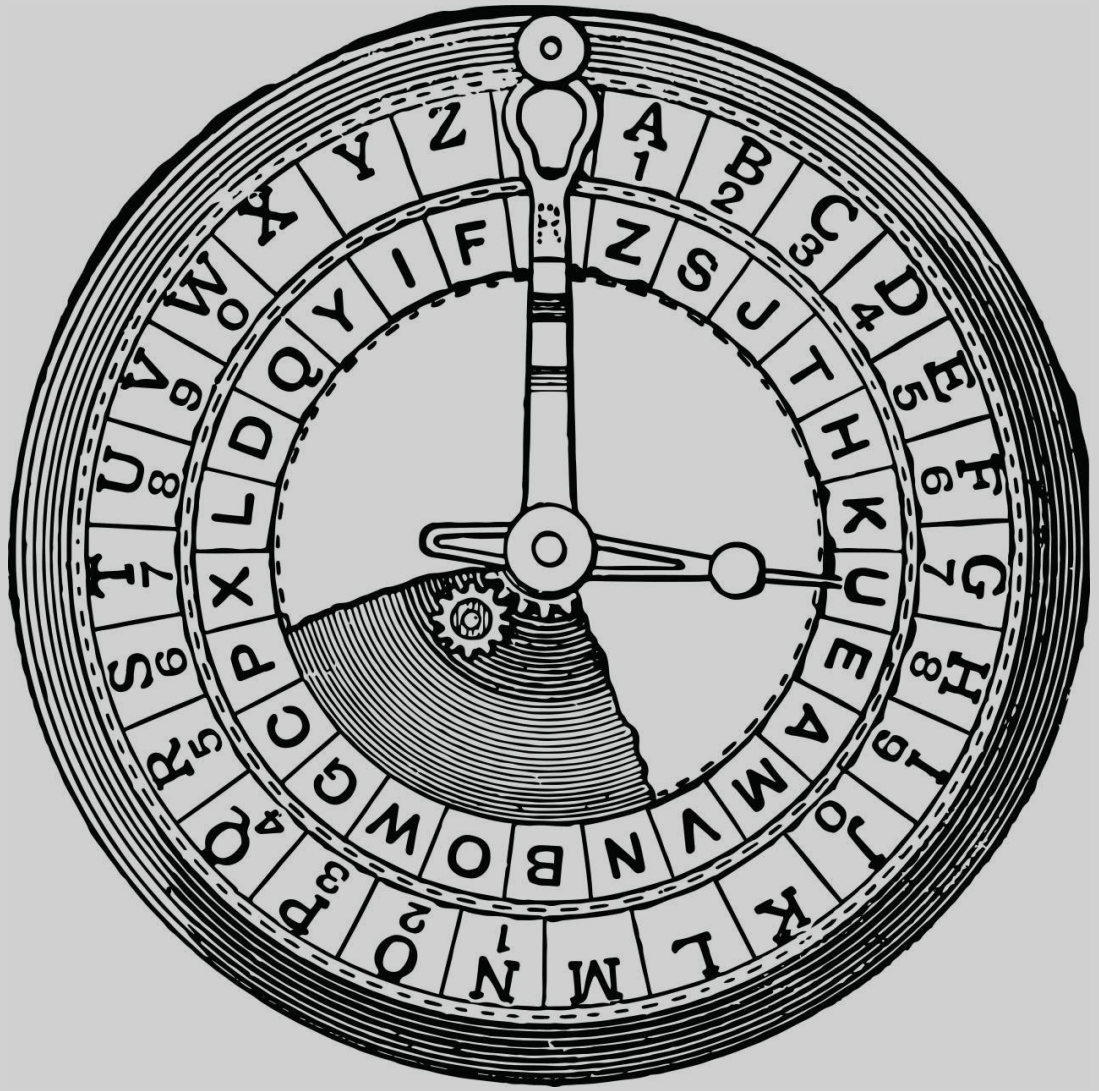
Definição

- **Definição 4.16.** Uma cifra simétrica Π é *infalsificável* se para todo adversário probabilístico polinomial $\mathcal{A}$, existe uma função negligenciável negl tal que:

$$\Pr\left[\text{Enc-Forge}_{\mathcal{A},\Pi}(n) = 1\right] \leq \mathrm{negl}(n)$$

- **Definição 4.17.** Uma cifra simétrica é uma *cifra autenticada* se é CCA-segura e infalsificável.
- **Observação:** na construção que veremos, o adversário pode ter um oráculo de decifragem também.

Construções genéricas



Ideia

- Combinar uma cifra segura com um MAC seguro
  - Sempre com chaves independentes!
- Nem todas as combinações funcionam

1. Cifrar-e-autenticar
   - Envia $c \leftarrow \mathrm{Enc}_{k_E}(m)$ e $t \leftarrow \mathrm{Mac}_{k_M}(m)$
2. Autenticar-e-então-cifrar
   - Computa $t \leftarrow \mathrm{Mac}_{k_M}(m)$ e envia $c \leftarrow \mathrm{Enc}_{k_E}(m||t)$
3. Cifrar-e-então-autenticar
   - Envia $c \leftarrow \mathrm{Enc}_{k_E}(m)$ e $t \leftarrow \mathrm{Mac}_{k_M}(c)$

Análise

- Vamos analisar cada uma das construções com componentes seguros “genéricos”
  - Cifra CPA-segura
  - MAC (fortemente) seguro
- Queremos uma construção que funcione com **qualquer** escolha dos componentes
  - Algumas só funcionam com componentes específicos
    - E serão rejeitadas como “inseguras” na nossa análise
    - Por que?
    - Principalmente pela possibilidade de troca de componentes (atualização de bibliotecas criptográficas)

Cifrar-e-autenticar

- Pode não ser nem EAV-seguro
  - Existem MACs que são seguros porém revelam a mensagem
  - Nossa própria construção seria assim ao concatenar o primeiro bit da mensagem na tag (continua seguro)
- Pode não ser CPA-seguro mesmo com “componentes padrão”
  - Por exemplo, um MAC determinístico
    - A tag t para uma mensagem é sempre a mesma
    - Permite ao adversário perceber quando uma mensagem é mandada duas vezes

Autenticar-e-então-cifrar

- Seguro para algumas escolhas, mas não todas
- Exemplo: modo CBC com padding
  - Vulnerável à ataque padding-oracle
- $\mathrm{Dec}'_{k_E,k_M}(c)$:
  1. Compute $\tilde{m} := \mathrm{Dec}_{K_E}(c)$. Se houver erro de padding, retorne “erro de padding”
  2. Interprete $\tilde{m}$ como $m||t$. Se $\mathrm{Vrfy}_{k_M}(m,t) = 1$, retorne m, senão retorne “erro de autenticação”
- Se o adversário consegue distinguir entre os dois tipos de erro, pode fazer o mesmo ataque padding-oracle de antes
- Uma maneira (não ideal) de mitigar: retornar apenas um tipo de erro

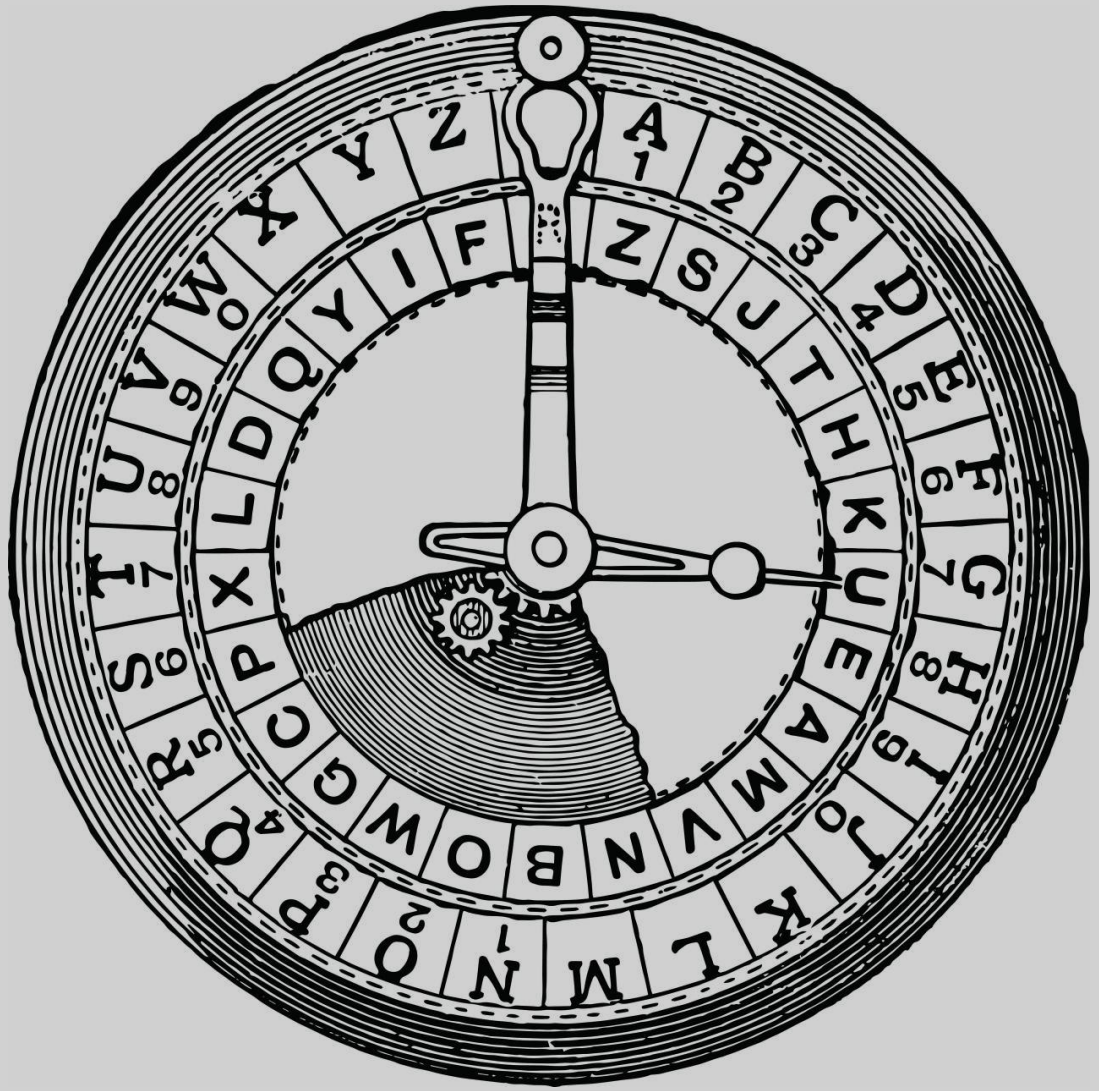
Cifrar-e-então-autenticar

- Primeiro cifra e depois computa o MAC
  - Envia $\langle c, t \rangle$ onde
    - $c \leftarrow \mathrm{Enc}_{k_E}(m)$
    - $t \leftarrow \mathrm{Mac}_{k_M}(c)$
- É seguro contanto que o MAC seja fortemente seguro
- Intuição para a segurança
  - Um MAC fortemente seguro garante que um adversário não consegue gerar **qualquer** texto cifrado válido
  - Garante a infalsificabilidade, mas também segurança CCA pois o oráculo de cifragem é “inútil”

Construção 4.18

- Seja $\Pi_E = (\mathrm{Enc}, \mathrm{Dec})$ uma cifra simétrica e $\Pi_M = (\mathrm{Mac}, \mathrm{Vrfy})$ um MAC, ambos com chaves de tamanho n escolhidas de maneira aleatória. Defina uma cifra simétrica $(\mathrm{Gen}', \mathrm{Enc}', \mathrm{Dec}')$ como a seguir:
  - Gen': na entrada 1^n, escolhe chaves $k_E, k_M \in \{0,1\}^n$ de maneira aleatória e independente, e retorna (k_E, k_M)
  - Enc': ao receber uma chave (k_E, k_M) e uma mensagem m, computa $c \leftarrow \mathrm{Enc}_{k_E}(m)$ e $t \leftarrow \mathrm{Mac}_{k_M}(c)$. Tem como saída $\langle c, t \rangle$
  - Dec': ao receber uma chave (k_E, k_M) e um texto cifrado $\langle c, t \rangle$, primeiro verifica se $\mathrm{Vrfy}_{k_M}(c, t) = 1$. Se sim, retorna $\mathrm{Dec}_{k_E}(c)$; senão, retorna $\perp$

Prova de segurança



Prova de segurança

- **Teorema 4.19.** Seja Π_E uma cifra simétrica CPA-segura, e seja Π_M um MAC fortemente seguro. Então a construção 4.18 é uma cifra autenticada.
- **Demonstração:** Seja $\mathcal{A}$ um adversário probabilístico polinomial que ataca a construção 4.18 no experimento CCA.
- Dizemos que um texto cifrado $\langle c, t \rangle$ é **novo** se $\mathcal{A}$ não recebeu $\langle c, t \rangle$ de seu oráculo de cifragem ou como o texto cifrado de desafio.
- Seja $\mathrm{ConsultaValida}$ o evento de que $\mathcal{A}$ realiza uma consulta ao oráculo de decifragem com um texto cifrado novo tal que o resultado não seja $\perp$

Consultas válidas são raras

- **Afirmação 4.20.** $\Pr[\mathrm{ConsultaValida}]$ é negligenciável.
- **Demonstração (afirmação):** Intuitivamente, se $\mathrm{ConsultaValida}$ ocorre, então o adversário conseguiu obter um par (c, t) para o experimento MAC-sforge.
- Seja $q(n)$ um limite polinomial para o número de consultas de decifragem de $\mathcal{A}$
- Vamos construir um adversário $\mathcal{A}_M$ para o experimento MAC-sforge

Consultas de cifragem



- Sempre que o adversário $\mathcal{A}$ consulta o seu oráculo de cifragem, $\mathcal{A}_M$ usa o desafiante para computar $\langle c, t \rangle$

Desafio



- As mensagens m_0, m_1 de desafio são tratadas de maneira similar

Consultas de cifragem



- Se $\langle c_j, t_j \rangle$ foi o resultado de alguma consulta de cifragem, $\mathcal{A}_M$ retorna a mensagem original m
- Senão, $\mathcal{A}_M$ retorna $\perp$
- Se $j = i$, então $\mathcal{A}_M$ produz $\langle c_i, t_i \rangle$ como saída para o experimento

Análise

- $\mathcal{A}_M$ tenta "adivinhar" que a i-ésima consulta de decifragem de $\mathcal{A}$ será para uma mensagem nova e válida
- Se isso acontecer, de fato $\mathcal{A}_M$ vence!
- Ponto chave: a visão de $\mathcal{A}$ na redução é idêntica à sua visão no experimento $\mathrm{PrivK}^{\mathrm{cca}}_{\mathcal{A},\Pi'}(n)$ até que o evento $\mathrm{ConsultaValida}$ ocorra
  - Consultas de cifragem são respondidas perfeitamente
  - Consultas de decifragem também!
    - Ou a redução já sabia a mensagem original
    - Ou (como o evento ainda não ocorreu) a consulta é inválida e a resposta seria $\perp$ mesmo
    - Lembrando que $\mathcal{A}$ não pode enviar o texto cifrado de desafio para seu oráculo de decifragem

Mais análise

- A visão de $\mathcal{A}$ na redução é idêntica à sua visão no experimento $\mathrm{PrivK}^{\mathrm{cca}}_{\mathcal{A},\Pi'}(n)$ até que o evento $\mathrm{ConsultaValida}$ ocorra
- Segue que a probabilidade do evento $\mathrm{ConsultaValida}$ no experimento $\mathrm{MAC\text{-}sforge}_{\mathcal{A}_M,\Pi_M}(n)$ é a mesma que sua probabilidade no experimento $\mathrm{PrivK}^{\mathrm{cca}}_{\mathcal{A},\Pi'}(n)$
- Se $\mathcal{A}_M$ adivinha o índice i em que o evento ocorre, então $\mathcal{A}_M$ vence no experimento $\mathrm{MAC\text{-}sforge}_{\mathcal{A}_M,\Pi_M}(n)$
  - Como a probabilidade de adivinhar i corretamente é $1/q(n)$

$$\Pr\left[\mathrm{MAC\text{-}sforge}_{\mathcal{A}_M,\Pi_M}(n) = 1\right] \geq \frac{\Pr[\mathrm{ConsultaValida}]}{q(n)}$$

- Como Π_M é fortemente segura, segue que $\Pr[\mathrm{ConsultaValida}]$ é negligenciável

Infalsificável e CCA seguro

- Resta provar que nosso esquema é infalsificável e CCA seguro
- Que o esquema é infalsificável segue diretamente da Afirmação 4.20
  - O adversário $\mathcal{A}'$ do experimento $\mathrm{Enc\text{-}Forge}$ é “mais fraco” que o adversário $\mathcal{A}$ que estamos considerando
    - Não tem acesso à um oráculo de decifragem, apenas de cifragem
    - Vence se e somente se consegue produzir um texto cifrado válido
    - O que vimos que só ocorre com probabilidade negligenciável
- Provar que nosso esquema Π' é CCA-seguro é mais trabalhoso

Segurança CCA

- Perceba o seguinte sobre o adversário $\mathcal{A}$

$$\Pr\left[\mathrm{PrivK}^{\mathrm{cca}}_{\mathcal{A},\Pi'}(n) = 1\right] \leq \Pr[\mathrm{ConsultaValida}] + \Pr\left[\mathrm{PrivK}^{\mathrm{cca}}_{\mathcal{A},\Pi'}(n) = 1 \wedge \overline{\mathrm{ConsultaValida}}\right]$$

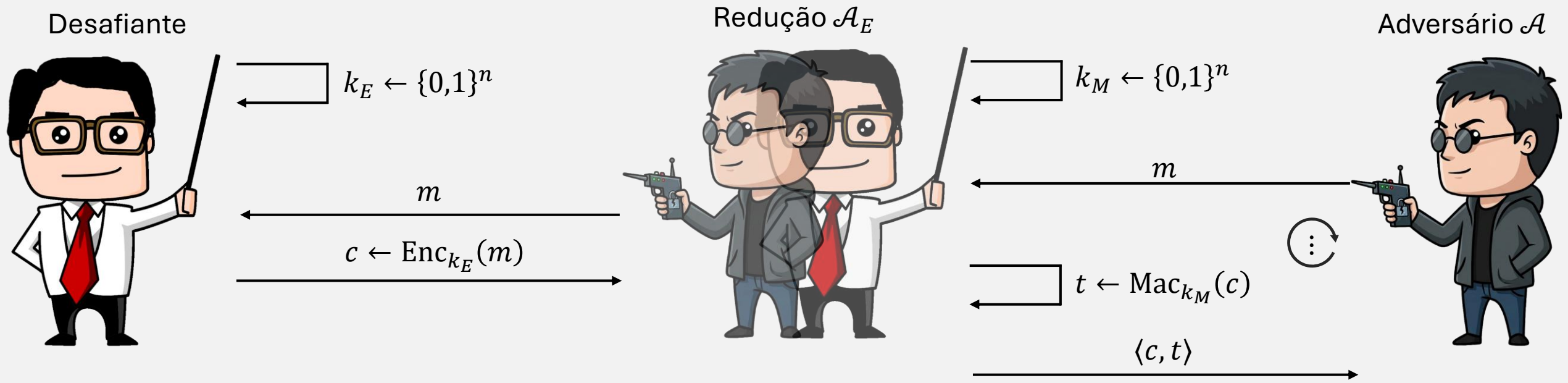
- Nos resta mostrar o seguinte
- **Afirmação 4.21.** Existe uma função negligenciável negl tal que

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

Demonstração da Afirmação 4.21

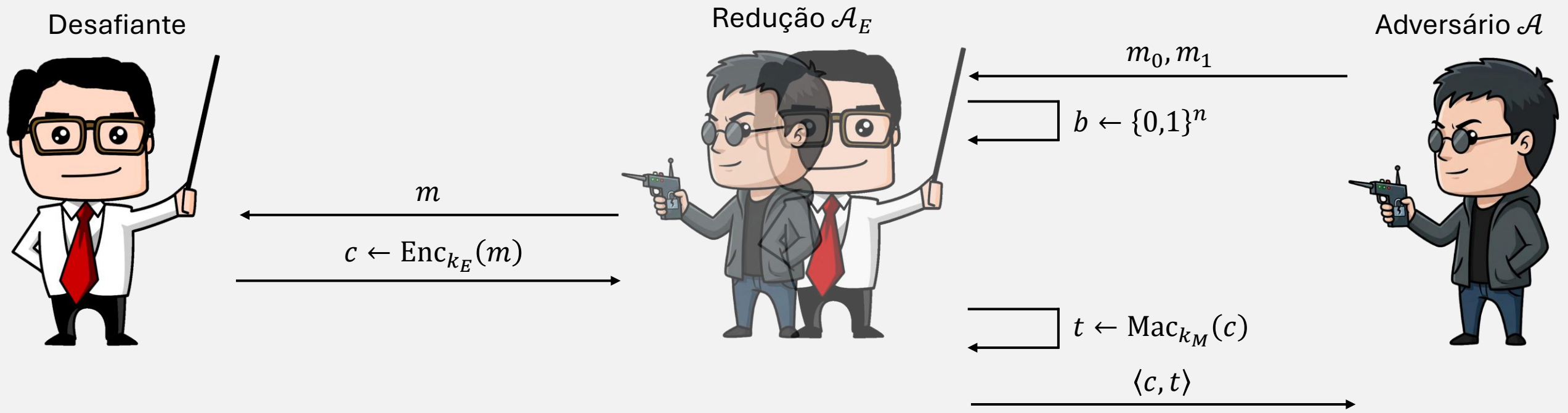
- Por redução à segurança CPA de Π_E
- Pontos importantes
  - A redução usa o desafiante CPA para cifrar a mensagem, porém computa a tag sozinha (sorteando uma chave k_M)
  - A redução precisa responder consultas de decifragem também
    - Mas como?
    - Basta responder $\perp$ sempre pois estamos assumindo que $\mathrm{ConsultaValida}$ não acontece
- Detalhes da redução (que chamamos de $\mathcal{A}_E$) nos próximos slides

Consultas de cifragem



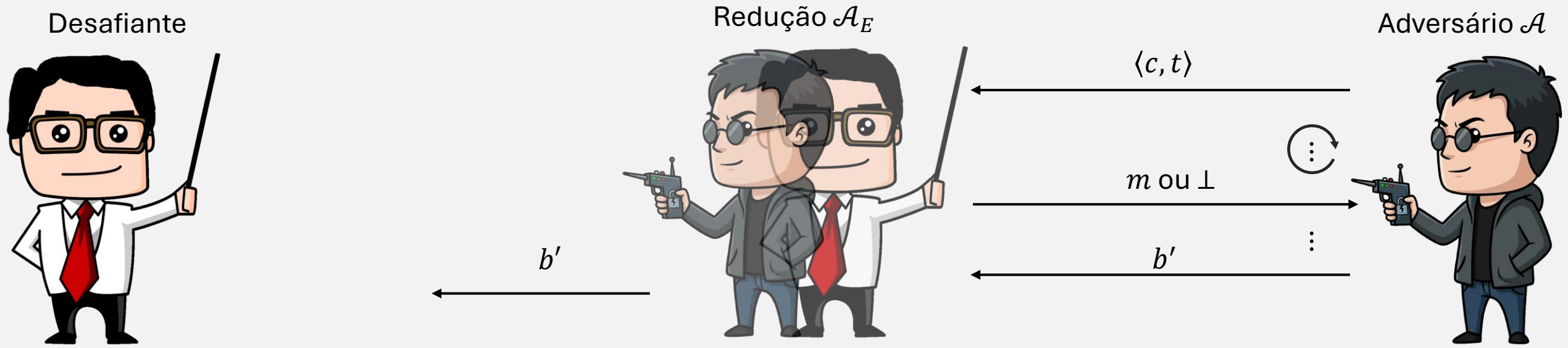
- Sempre que o adversário $\mathcal{A}$ consulta o seu oráculo de cifragem, $\mathcal{A}_M$ usa o desafiante para computar c, e computa t sozinho

Desafio



- As mensagens m_0, m_1 de desafio são tratadas de maneira similar

Consultas de cifragem



- Se $\langle c, t \rangle$ foi o resultado de alguma consulta de cifragem, $\mathcal{A}_E$ retorna a mensagem original m
- Senão, $\mathcal{A}_E$ retorna $\perp$
- No fim, $\mathcal{A}_E$ repete a saída de $\mathcal{A}$

Finalizando

- Sabendo que ConsultaValida não ocorre, a visão de $\mathcal{A}$ é idêntica na redução e no experimento $\mathrm{PrivK}^{\mathrm{cca}}_{\mathcal{A},\Pi'}(n)$
- Segue que

$$\Pr\left[\mathrm{PrivK}^{\mathrm{cca}}_{\mathcal{A},\Pi'}(n) = 1 \wedge \overline{\mathrm{ConsultaValida}}\right] = \Pr\left[\mathrm{PrivK}^{\mathrm{cpa}}_{\mathcal{A}_E,\Pi_E}(n) = 1 \wedge \overline{\mathrm{ConsultaValida}}\right]$$

- Sabendo que

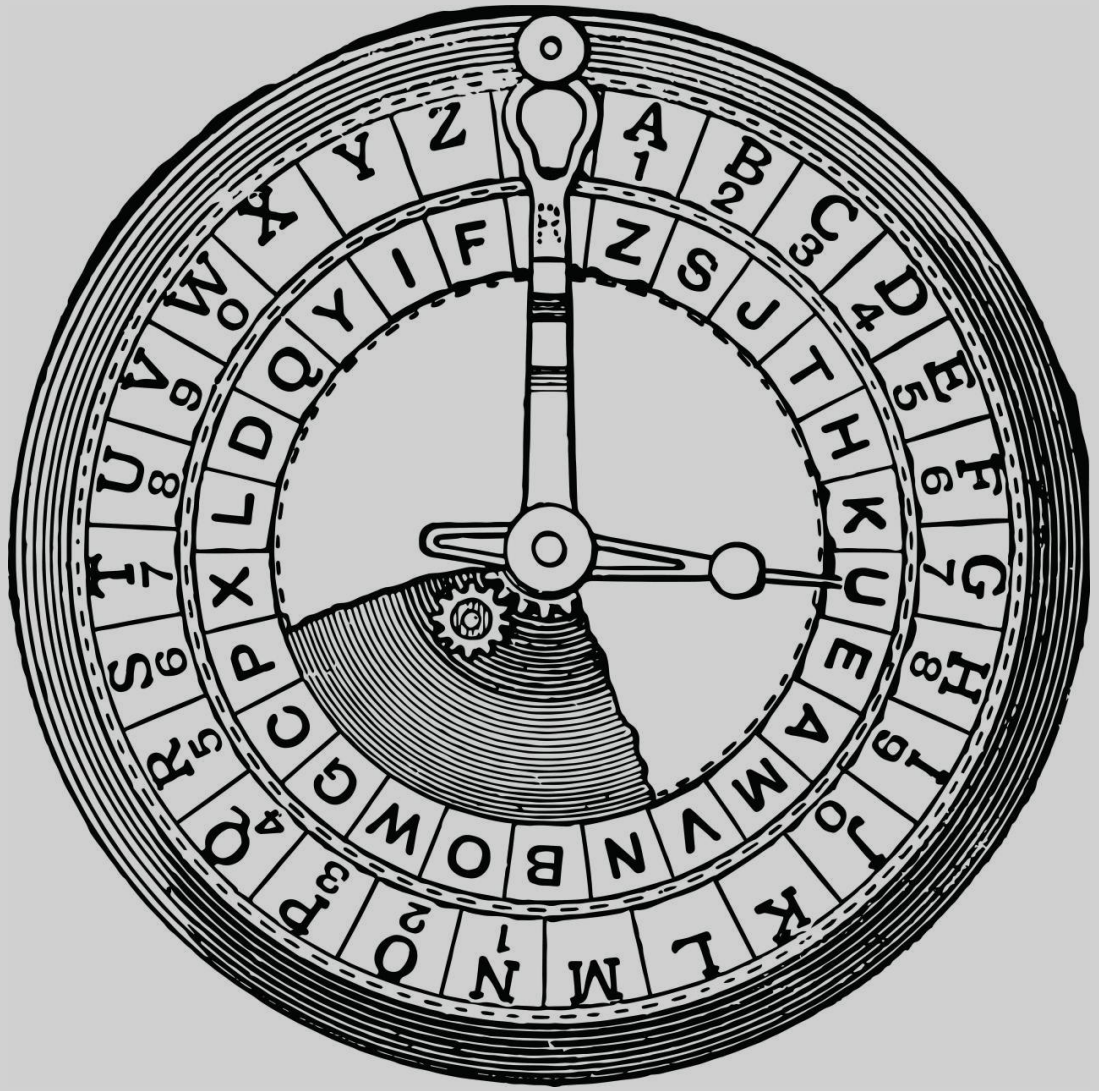
$$\Pr\left[\mathrm{PrivK}^{\mathrm{cpa}}_{\mathcal{A}_E,\Pi_E}(n) = 1 \wedge \overline{\mathrm{ConsultaValida}}\right] \leq \Pr\left[\mathrm{PrivK}^{\mathrm{cpa}}_{\mathcal{A}_E,\Pi_E}(n) = 1\right]$$

- Como Π_E é CPA-segura, existe uma função negligenciável negl tal que

$$\Pr\left[\mathrm{PrivK}^{\mathrm{cpa}}_{\mathcal{A}_E,\Pi_E}(n) = 1\right] \geq \frac{1}{2} + \mathrm{negl}(n)$$

- Completando a prova

Para fechar



Para fechar

- Hoje
  - Cifras autenticadas
    - Segredo
    - Integridade
  - Construção genérica de cifra autenticada
- Próxima aula
  - Funções hash criptográficas
    - Mapeiam strings para strings menores
    - De maneira que seja difícil encontrar colisões