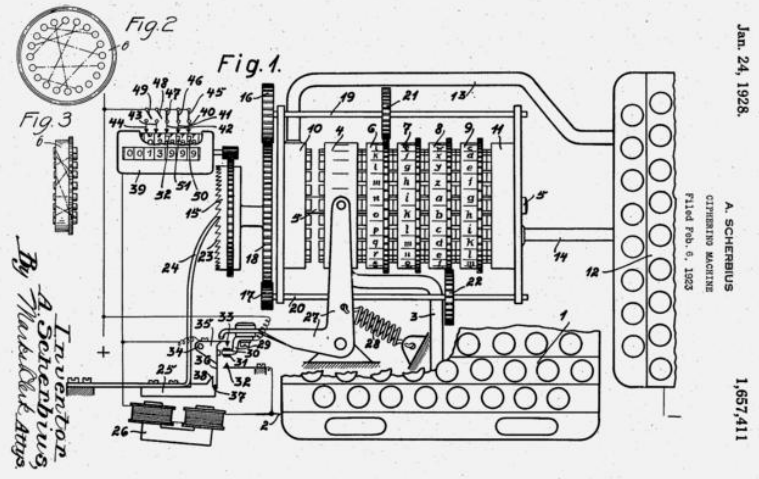
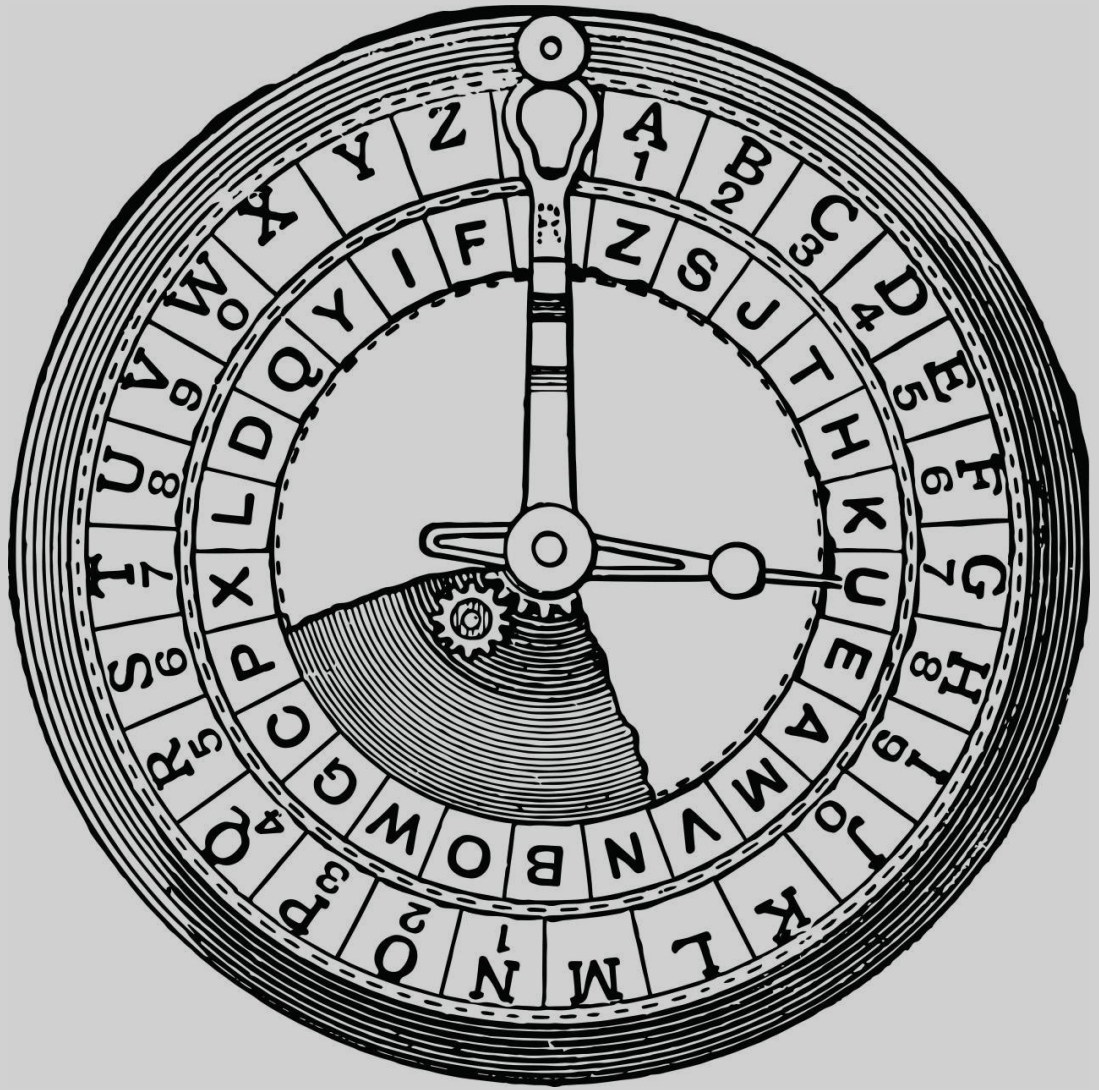
Mais Sobre Funções Hash

Fundamentos da Criptografia

Nicollas Mocelin Sdroievski



Na última aula



Função hash

- Uma função H que comprime a sua entrada
- Objetivo principal: evitar colisões
  - Colisão = duas entradas distintas x e y tal que $H(x) = H(y)$
  - Um adversário eficiente só pode conseguir encontrar uma colisão com probabilidade negligenciável
- **Definição 5.1.** Uma *função hash* (com tamanho de saída ℓ) é um par (Gen, H) de algoritmos probabilísticos polinomiais que satisfazem as seguintes condições:
  - Gen é um algoritmo probabilístico que recebe um parâmetro de segurança 1^n e tem como saída uma chave s. Assumimos que n pode ser determinado a partir de s
  - H recebe uma chave s e uma string $x \in \{0,1\}^*$ e tem como saída $H^s(x) \in \{0,1\}^{\ell(n)}$

O experimento $\text{Hash-coll}_{\mathcal{A},H}(n)$

Desafiante

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



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

s

x, x'



- $\mathcal{A}$ obtém sucesso se e somente se $x \neq x'$ e $H^s(x) = H^s(x')$
- Neste caso escrevemos $\text{Hash-coll}_{\mathcal{A},H}(n) = 1$

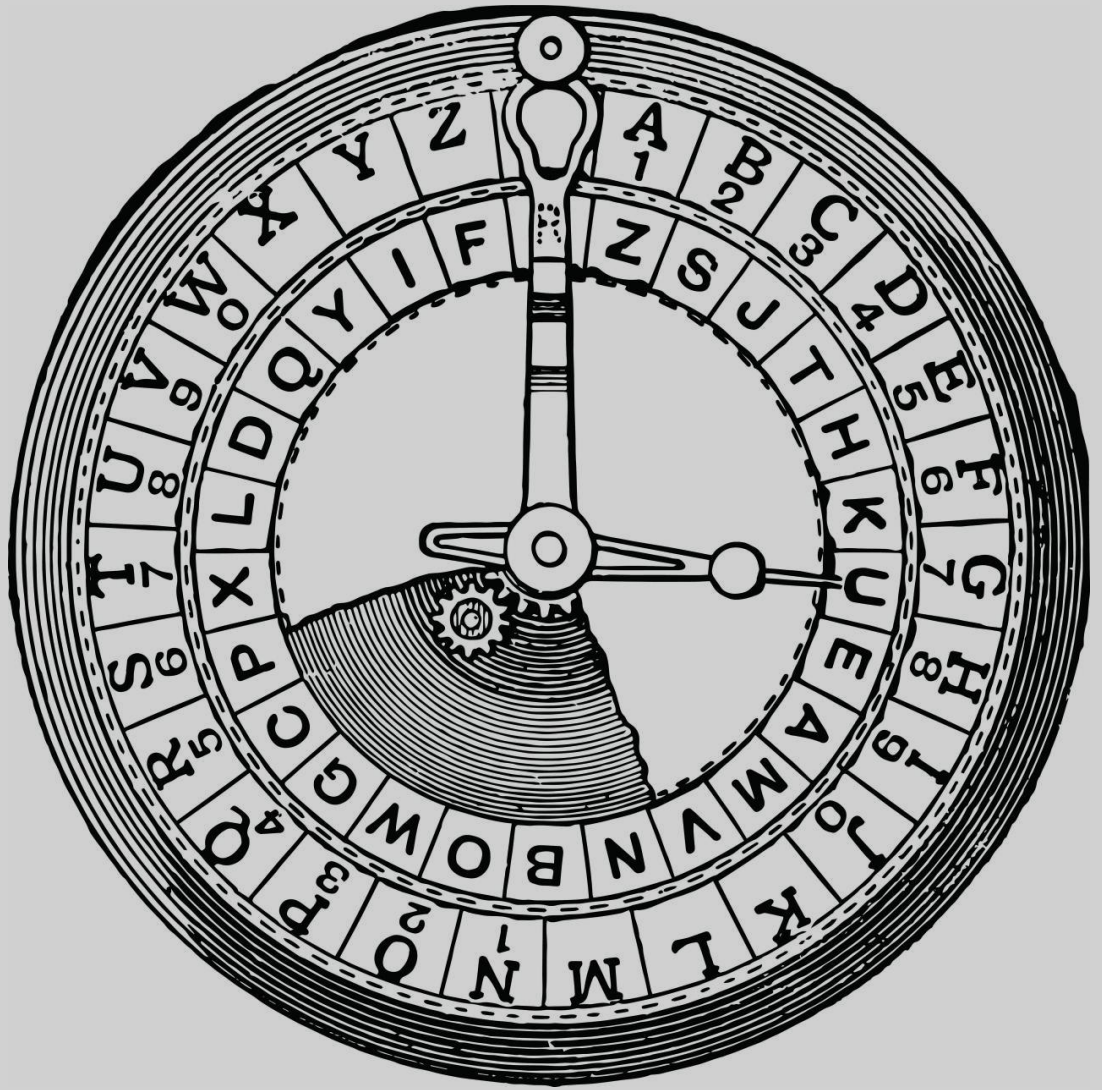
Definição de segurança

- **Definição 5.2.** Uma função hash $\Pi = (\mathrm{Gen}, H)$ é *resistente à colisões* se para todos os adversários probabilísticos polinomiais $\mathcal{A}$ existe uma função negligenciável negl tal que

$$\Pr[\text{Hash-coll}_{\mathcal{A},H}(n) = 1] \leq \mathrm{negl}(n)$$

- Por simplicidade, vamos nos referir à H ou H^s como uma função hash resistente à colisões, quando tecnicamente só poderíamos dizer isso sobre (Gen, H)

Um ataque genérico à funções hash



Nosso modelo de análise

- Fixamos $H : \{0,1\}^* \to \{0,1\}^\ell$
- Ataque trivial?
  - Força bruta
  - Tempo $O(2^\ell)$
- Assumimos que o valor de $H(x)$ é uniforme
  - Essencialmente o pior caso
  - Mais fácil encontrar colisões quando H não é “tão aleatória”

Ataque do aniversário

- Escolhemos q valores $x_1, \dots, x_q$ distintos para tentar encontrar uma colisão
- Lembre que o valor de $H(x)$ é aleatório uniforme
- Qual a probabilidade de que uma colisão ocorra?
  - A análise é equivalente à análise do problema do aniversário
  - Se q pessoas estão em uma sala, qual a probabilidade de que duas tenham o mesmo aniversário (com cada data escolhida dentre 365 possíveis)?
    - Com 23 pessoas, aproximadamente $1/2$
  - No nosso caso
    - Sorteamos q imagens $y_1, \dots, y_q$ em $\{0,1\}^{\ell}$
    - E queremos saber a probabilidade de que existam $i \neq j$ tal que $y_i = y_j$

Análise

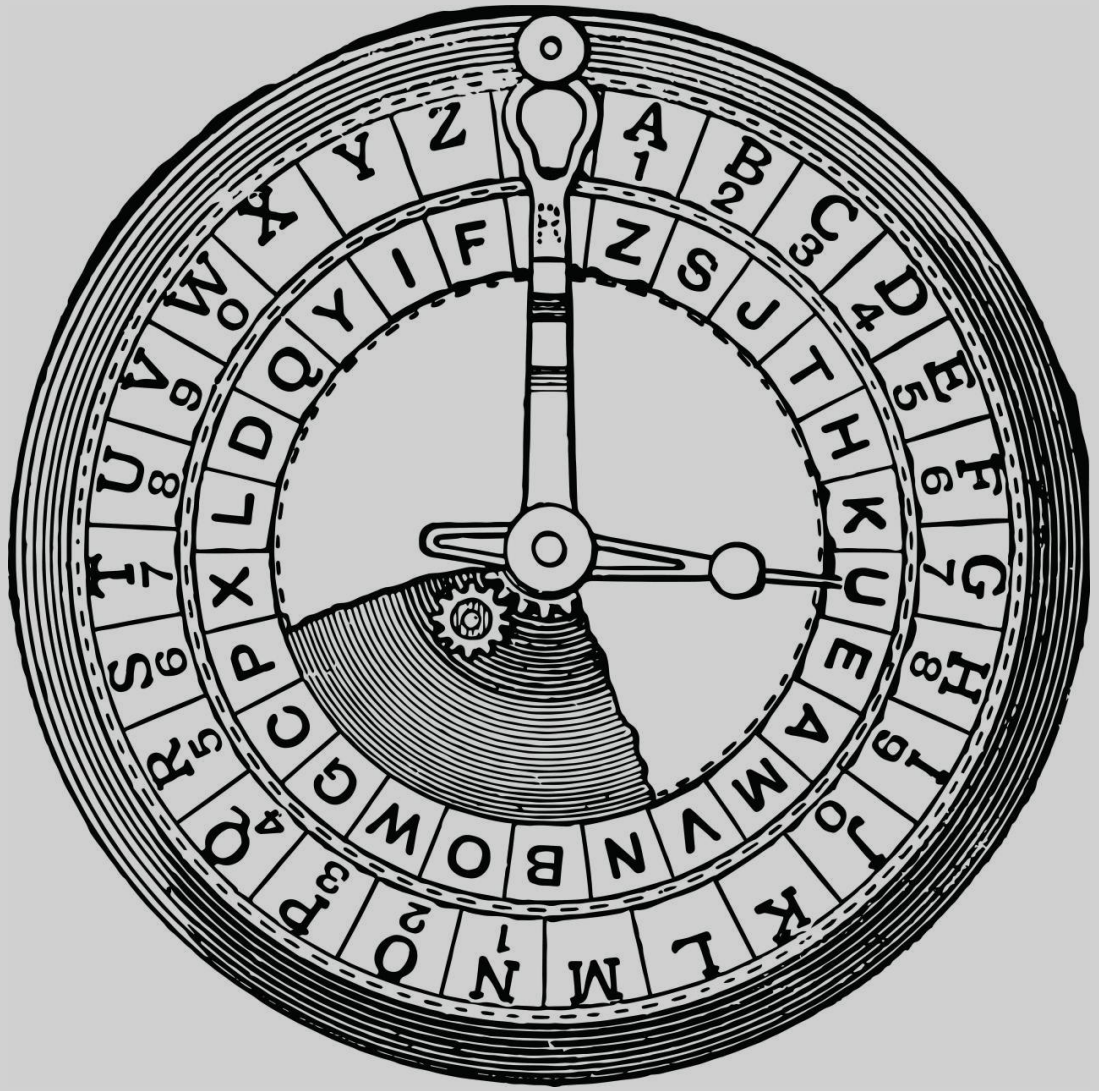
- **Lema A.16.** Fixe um inteiro positivo N e considere escolher $q \leq \sqrt{2N}$ elementos $y_1, \ldots, y_q$ de maneira aleatória e uniforme de um conjunto de tamanho N. Então a probabilidade de que existam $i \neq j$ tal que $y_i = y_j$ é de pelos menos $\frac{q(q-1)}{4N}$.

- Usando o Lema, basta escolher $q = \Theta\left(2^{\ell/2}\right)$ para garantir probabilidade de colisão aproximadamente $1/2$
- Se queremos que nossa função hash seja resistente à colisões contra adversários de tempo T, precisamos que $\ell \geq 2 \log T$

Encontrando colisões mais interessantes

- No geral, queremos encontrar colisões entre duas mensagens x_1 e x_2 que "façam sentido"
- Lembramos que o ataque só pede que as entradas sejam distintas, não aleatórias!
- Basta então, gerar $q = \Theta(2^{\ell/2})$ mensagens "similares" à x_1 e $q = \Theta(2^{\ell/2})$ mensagens "similares" à x_2
  - Uma análise semelhante ao Lema A.16 garante uma probabilidade próxima de $1/2$ de sucesso
- **Observação:** uma mensagem com 64 palavras que admitem um sinônimo já permite 2^{64} variações "similares"

Hash-and-MAC



MACs para mensagens de tamanho arbitrário

- Vimos algumas opções:

1. Usar uma função pseudoaleatória com tamanho de entrada arbitrário
2. Incluir várias informações na própria mensagem, como número de blocos, índice de cada bloco e um ID aleatório
3. CBC-MAC

- Veremos uma ideia nova usando hashes
  - E bem natural

Ideia

- Temos um MAC para mensagens de tamanho $\ell(n)$ e queremos estendê-lo para mensagens de tamanho arbitrário
- Também temos uma função hash que produz hashes de tamanho $\ell(n)$
- Como combinar os dois?
  - Primeiro computamos o hash $H(m)$ da mensagem m
  - E então o MAC de $H(m)$

Construção 5.5

- Seja $\Pi = (\mathrm{Mac}, \mathrm{Vrfy})$ um MAC para mensagens de tamanho $\ell(n)$, e $\Pi_H = (\mathrm{Gen}_H, H)$ uma função hash com tamanho de saída $\ell(n)$. Defina um MAC $\Pi' = (\mathrm{Gen}', \mathrm{Mac}', \mathrm{Vrfy}')$ para mensagens de tamanho arbitrário a seguir:
  - Gen': na entrada 1^n, escolhe $k \leftarrow \{0,1\}^n$ de maneira uniforme e $s \leftarrow \mathrm{Gen}_H(1^n)$; a chave é $k' = \langle k, s \rangle$.
  - Mac': ao receber como entrada uma chave $k' = \langle k, s \rangle$ e uma mensagem $m \in \{0,1\}^*$, retorna $t \leftarrow \mathrm{Mac}_k\big(H^s(m)\big)$.
  - Vrfy': ao receber como entrada uma chave $k' = \langle k, s \rangle$, uma mensagem $m \in \{0,1\}^*$ e uma tag t, retorna 1 se e somente se $\mathrm{Vrfy}_k(H^s(m), t) = 1$.

Intuição da segurança

- Como H^s é resistente à colisões, autenticar $H^s(m)$ é "equivalente" a autenticar m de fato
- Imagine que um adversário $\mathcal{A}$ consegue forjar uma tag para uma mensagem $m^* \notin Q$ que não foi consultada antes, então, um de dois casos aconteceu

1. Existe uma mensagem $m \in Q$ tal que $H^s(m) = H^s(m^*)$
   - Então $\mathcal{A}$ encontrou uma colisão em H^s
2. Para toda mensagem $m \in Q, H^s(m) \neq H^s(m^*)$
   - Então $\mathcal{A}$ forjou uma tag para a "mensagem" $H^s(m^*)$, que não foi consultada antes.

Demonstração de segurança

- **Teorema 5.6.** Se Π é um MAC seguro para mensagens de tamanho ℓ e Π_H é resistente à colisões, então a Construção 5.5 é um MAC seguro.
- **Demonstração:** Sejam
  - Π' o MAC da construção,
  - $\mathcal{A}'$ um adversário no experimento $\text{MAC-forge}_{\mathcal{A}',\Pi'}(n)$,
  - Q o conjunto de mensagens consultadas por $\mathcal{A}'$,
  - (m^*, t) a saída final de $\mathcal{A}'$, assumindo que $m^* \notin Q$,
  - Colisão o evento em que, no experimento $\text{MAC-forge}_{\mathcal{A}',\Pi'}(n)$, existe $m \in Q$ tal que $H^s(m) = H^s(m^*)$.

Demonstração de segurança

- É verdade que:

$$
\begin{aligned}
&\Pr\left[\text{MAC-forge}_{\mathcal{A}',\Pi'}(n) = 1\right] \\
&= \Pr\left[\text{MAC-forge}_{\mathcal{A}',\Pi'}(n) = 1 \wedge \text{Colisão}\right] + \Pr\left[\text{MAC-forge}_{\mathcal{A}',\Pi'}(n) = 1 \wedge \overline{\text{Colisão}}\right] \\
&\leq \Pr[\text{Colisão}] + \Pr\left[\text{MAC-forge}_{\mathcal{A}',\Pi'}(n) = 1 \wedge \overline{\text{Colisão}}\right]
\end{aligned}
$$

- Vamos provar que os dois termos da última equação são negligenciáveis
  - O primeiro através de uma redução para a segurança da função hash
  - E o segunda através de uma redução para a segurança do MAC

Primeira redução



- $\mathcal{A}_H$ usa as chaves s e k para responder às consultas de $\mathcal{A}'$
- Se a mensagem m^* produzida por $\mathcal{A}'$ colide com alguma mensagem m_i anterior
  - $H^s(m^*) = H^s(m_i)$
  - Então $\mathcal{A}_H$ retorna (m^*, m_i)

Pr[Colisão] é negligenciável

- A visão de $\mathcal{A}'$ na redução é idêntica à sua visão no experimento $\text{MAC-forge}_{\mathcal{A}',\Pi'}(n)$
- Como $\mathcal{A}_H$ produz uma colisão exatamente quando o evento Colisão ocorre, segue que

$$\Pr[\text{Colisão}] = \Pr\left[\text{Hash-coll}_{\mathcal{A}_H,\Pi_H}(n)\right]$$

Que é negligenciável pois Π_H é resistente à colisões

Segunda redução



- $\mathcal{A}_M$ usa a chave s e o desafiante para responder às consultas de $\mathcal{A}'$
- E tem como saída $(H^s(m^*), t)$, onde (m^*, t) é a saída de $\mathcal{A}'$

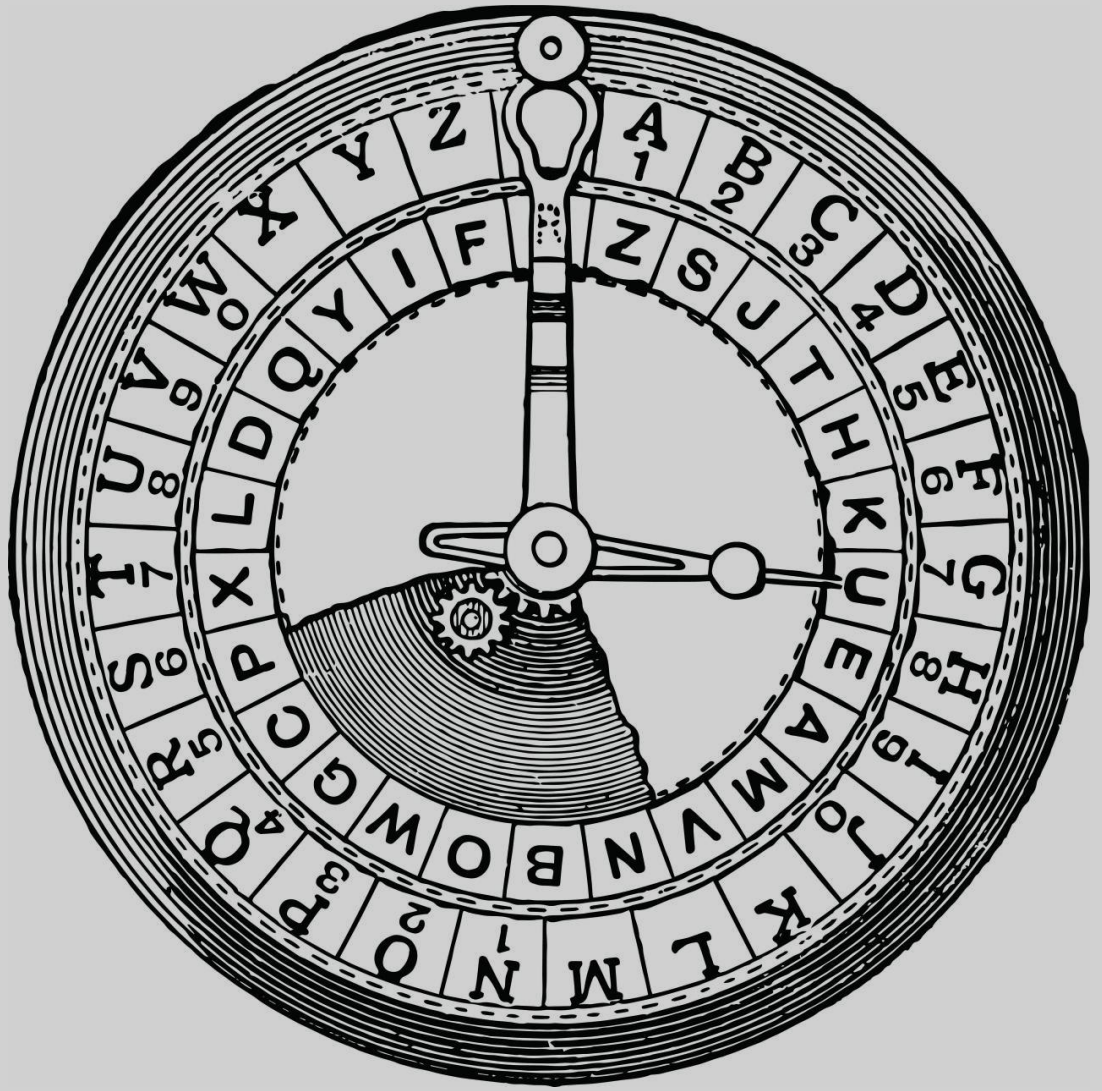
$\Pr\left[\text{MAC-forge}_{\mathcal{A}',\Pi'}(n) = 1 \wedge \overline{\text{Colisão}}\right]$ é negligenciável

- Novamente, a visão de $\mathcal{A}'$ na redução é idêntica à sua visão no experimento $\text{MAC-forge}_{\mathcal{A}',\Pi'}(n)$
- Quando os eventos $\text{MAC-forge}_{\mathcal{A}',\Pi'}(n) = 1$ e $\overline{\text{Colisão}}$ ocorrem:
  - $\mathcal{A}'$ nunca consultou seu oráculo com uma mensagem m_i tal que $H^s(m_i) = H^s(m^*)$
    - Logo, $\mathcal{A}_M$ nunca consultou o seu oráculo com uma "mensagem" igual a $H^s(m^*)$
  - t é uma tag válida (de acordo com Π) para a mensagem $H^s(m^*)$
  - Segue que

$$\Pr\left[\text{MAC-forge}_{\mathcal{A}',\Pi'}(n) = 1 \wedge \overline{\text{Colisão}}\right] = \Pr\left[\text{MAC-forge}_{\mathcal{A}_M,\Pi}(n) = 1\right]$$

Como Π é um MAC seguro, segue que essa probabilidade é negligenciável

Modelo de oráculo aleatório



Motivação

- Várias aplicações usando funções hash não possuem provas de segurança baseadas na resistência à colisões da função utilizada
  - Em muitos casos, provar a segurança parece bastante complicado
- Algumas soluções
  - Encontrar construções que podem ser provadas seguras
    - **Problema:** o que fazer enquanto não encontramos?
  - Usar a construção mesmo sem uma prova de segurança
    - **Problema:** vai contra a nossa disciplina inteira!
  - Introduzir um modelo idealizado em que conseguimos provar a segurança
    - **Problema:** talvez não modele a realidade de maneira precisa

Modelo de oráculo aleatório

- Exemplo de modelo idealizado
- Trata a função hash H como uma função verdadeiramente aleatória
  - Existe uma função **pública** e **aleatória** H que só pode ser avaliada ao realizar uma consulta à um oráculo que retorna $H(x)$ ao receber x como entrada
  - Lembrando: funções pseudoaleatórias são seguras **apenas** quando sua chave é mantida em segredo!
- Ideia
  - Desenvolver (e provar a segurança de) um protocolo criptográfico usando um oráculo aleatório
  - Na prática, **instanciar** o oráculo através de uma função hash criptográfica

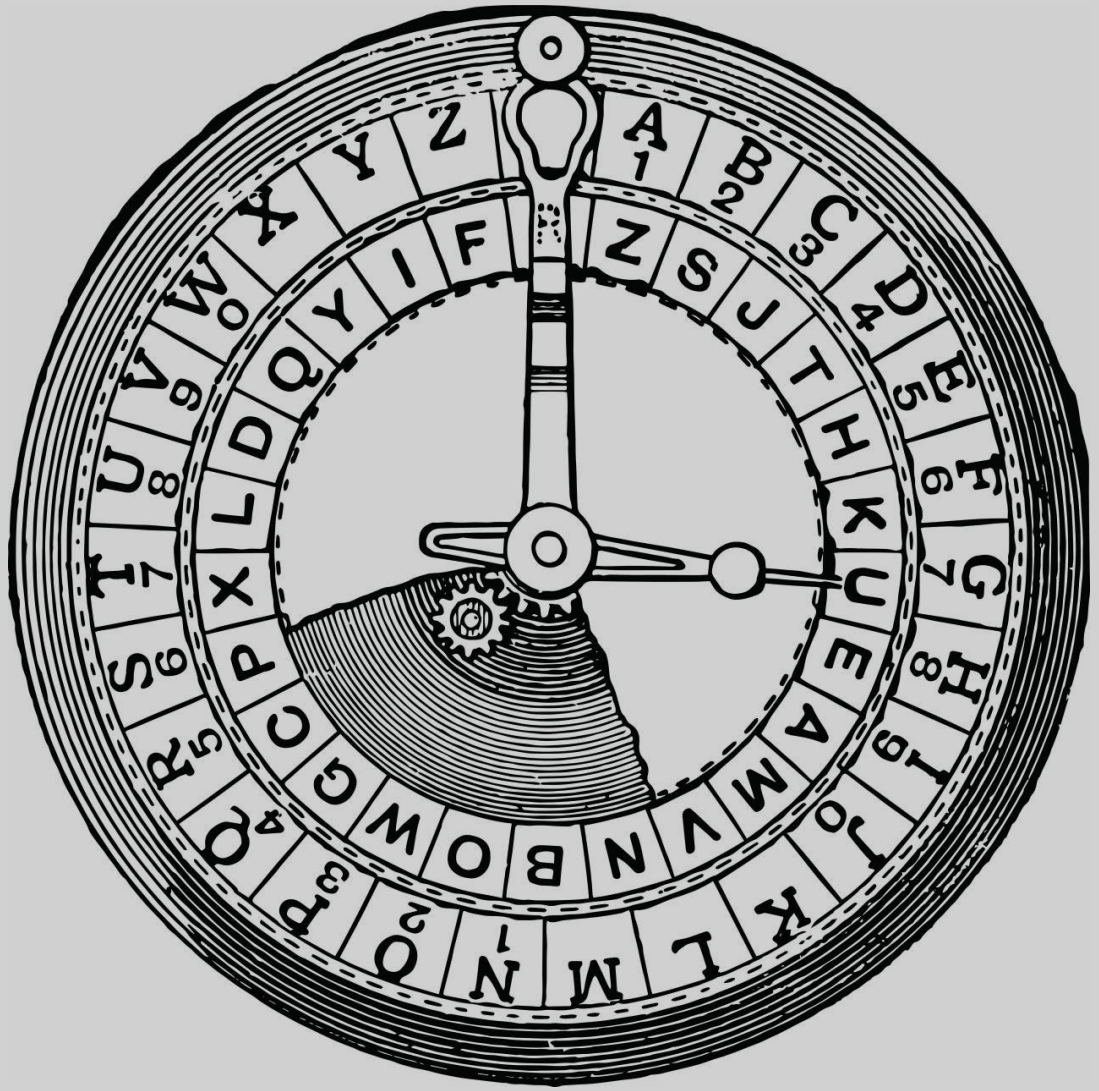
Agumentos contra

- Não existe justificativa formal ou rigorosa para acreditar que uma prova no modelo se traduz para segurança no mundo real
- Impossível instanciar o oráculo a partir de uma função hash
  - Uma vez escolhida a função:
    - Todos os valores já estão definidos (só são “desconhecidos”)
    - Um adversário pode usar o código de H no seu ataque
- Não sabemos quando uma função hash é “boa o suficiente” para instanciar um oráculo aleatório
- Existem construções (esquisitas) que são seguras no modelo porém inseguras quando se usa uma função hash criptográfica

Argumentos a favor

- Permite construções muito mais eficientes do que as do modelo “tradicional” (com primitivas criptográficas)
  - Especialmente para criptografia de chave pública
  - Várias construções usadas no mundo real com provas de segurança deste tipo
- “Uma prova de segurança no modelo é melhor do que nenhuma prova”
  - Uma prova deste tipo indica que a ideia geral da construção é “boa”, e apenas alguma fraqueza da própria função hash poderia ser um problema
  - Até então, nenhum ataque no mundo real para construções seguras de acordo com o modelo

Para fechar



Para fechar

- Hoje
  - Alguns ataques genéricos à funções hash
  - Funções Hash e MACs
  - O modelo de oráculo aleatório (Random-Oracle model)
- Próxima aulas
  - Objetivo: criptografia de chave pública (assimétrica)
  - Na próxima aula, especificamente
    - Teoria de números
    - Teoria de grupos