Números Inteiros e Criptografia, 2021.2

Trabalho Final[†]

Submeta as soluções de todas as questões do trabalho até **10 de março às 9:00** salvando os arquivos na sua pasta chamada "Trabalho Final" no Google Drive

Em todas as questões que envolverem codificação de texto para número ou vice-versa (incluindo a sua implementação do RSA), usaremos a tabela de correspondência entre números e símbolos dada na última página deste PDF.

Lembre-se: você pode usar tudo o que foi visto em aula, em listas anteriores, ou mesmo qualquer questão do trabalho para responder outras questões (mesmo que você não tenha feito a questão que está citando!), desde que você seja claro na sua citação do que está usando. A única exceção é a Questão 2, na qual você não pode usar o item **b** na solução do item **a**, a não ser que você faça o item **b**.

Como sempre, justifique todas as questões!

Questão 1 (Critérios de divisibilidade)**.** *Lembrete de Fundamentos da Computação Digital.* Dado um natural $b > 0$, dizemos que um natural n tem *expansão* $(d_k d_{k-1} \cdots d_2 d_1 d_0)_b$ *na base* b se cada d_i é um natural menor do que b e

$$
\begin{aligned}
n &= (d_k \cdot b^k) + (d_{k-1} \cdot b^{k-1}) + \cdots + (d_2 \cdot b^2) + (d_1 \cdot b^1) + (d_0 \cdot b^0) \\
&= \sum_{i=0}^{k} d_i \cdot b^i
\end{aligned}
$$

a. Sejam $a, b, n \in \mathbb{N}$ tais que $b \equiv 1 \pmod{a}$ e n tem expansão $(d_k d_{k-1} \cdots d_2 d_1 d_0)_b$ na base b.

Mostre que n é divisível por a se, e somente se, $\left(\sum_{i=0}^{k} d_i\right)$ é divisível por a.

b. Use o item **a** para mostrar que $(123412341)_8$ é divisível por 7.

c. Prove que um natural é divisível por 5 se, e somente se, ao ser escrito em base 10, seu último algarismo (i.e., o seu algarismo da casa das unidades) é divisível por 5.

[†]Publicado em 21/02.

Questão 2 (Teorema Chinês dos Restos)**.**

a (TCR, versão simplificada)**.** Sejam m_0 e m_1 naturais coprimos e $n = m_0 \cdot m_1$. Prove que, para quaisquer inteiros a e b, o sistema

$$\begin{cases} x \equiv a \pmod{m_0} \\ x \equiv b \pmod{m_1} \end{cases}$$

tem *uma única* solução em módulo n, e esta solução é dada por

$$x \equiv (a \cdot m_1 \cdot m_1') + (b \cdot m_0 \cdot m_0') \pmod{n},$$

onde m_0' é o inverso de m_0 no módulo m_1 e m_1' é o inverso de m_1 no módulo m_0.

b (TCR, versão completa)**.** Suponha que $m_0, m_1, \ldots, m_{k-1}$ sejam primos entre si, i.e., $\mathrm{mdc}(m_i, m_j) = 1$ para todos $0 \leq i < j \leq k-1$, e seja $n = \prod_{i=0}^{k-1} m_i$.

Prove que, para quaisquer inteiros $a_0, a_1, \ldots, a_{k-1}$, o sistema

$$\begin{cases} x \equiv a_0 \pmod{m_0} \\ x \equiv a_1 \pmod{m_1} \\ \quad \vdots \\ x \equiv a_{k-1} \pmod{m_{k-1}} \end{cases}$$

possui *uma única solução* módulo n, e esta solução é dada por

$$x \equiv \sum_{i=0}^{k-1} (a_i \cdot \ell_i \cdot \ell_i') \pmod{n},$$

onde para cada i com $0 \leq i \leq k-1$ definimos

$$\ell_i = \frac{n}{m_i} = \prod_{\substack{j \in \{0,1,\ldots,k-1\} \\ j \neq i}} m_j$$

$$\ell_i' = \text{o inverso multiplicativo de } \ell_i \text{ no módulo } m_i.$$

Questão 3. O número $e = 10$ nunca pode ser usado como expoente público no RSA. Por quê?

Questão 4 (Aceleração de descriptação no RSA com o TCR)**.** Uma das aplicações práticas do Teorema Chinês dos Restos é para acelerar a etapa de descriptação de mensagens. O procedimento é o seguinte: ao gerar seu módulo público $n = pq$, expoente público e e expoente privado d, o usuário também calcula e guarda (em segredo!) os seguintes valores:

- p
- q
- o inverso de p módulo q
- o inverso de q módulo p
- a forma reduzida de d módulo $p-1$
- a forma reduzida de d módulo $q-1$.

Lembrando que a tarefa básica na etapa de descriptação é, ao receber um bloco encriptado m, calcular a forma reduzida da potência modular $m^d \mod pq$, explique como usar o Teorema Chinês dos Restos (e o Pequeno Teorema de Fermat) e os dados calculados acima para tornar essa tarefa mais fácil (e explique em que sentido a tarefa fica mais fácil).

Questão 5. Três pares (n, e) de chaves públicas do RSA,

$$(9159429724473829475821613283436298601833, 11)$$

$$(1242030202519254059474455275778009437469, 3)$$

$$(1828982866109630997434176303059685778909, 5)$$

foram geradas usando somente 5 números primos distintos no total. Usando apenas o conhecimento obtido nessa disciplina, explique como a fatoração em primos de dois desses módulos públicos pode ser encontrada rapidamente (e dê a fatoração).

Questão 6.

a. Como vimos, a segurança do RSA está, em parte, baseada no fato de que é difícil calcular *raízes* modulares em geral: dados a natural com $a < n$ e um natural e, é dificil encontrarmos x tal que $x^e \equiv a \mod n$. Note que isso é diferente em aritmética comum, onde até mesmo calculadoras simples de bolso são capazes de encontrar x tal que $x^e = a$ (assuma esse fato como verdadeiro).

Explique por que se alguma solução x para a congruência $x^e \equiv a \mod n$ satisfaz

$$0 \leq x^e < n,$$

então é **fácil** encontrar x.

b. Usando o item **a**, explique a importância de que, no RSA com expoente público e pequeno e módulo n grande, cada bloco b da mensagem a ser encriptada seja razoavelmente *grande* (apesar de menor do que n, é claro).

c. Considere a situação em que k pessoas, $P_0, P_1, \ldots, P_{k-1}$, tenham, cada uma, sua própria chave pública para o módulo, mas a mesma chave pública e para o expoente. Seja n_i o módulo da chave pública da pessoa P_i e assuma que todos esses módulos são primos entre si. Agora suponha que Maria codifique a mesma mensagem m para cada pessoa: suponha que tenhamos $0 \leq m \leq \min\{n_0, n_1, \ldots, n_{k-1}\}$ e Maria manda $c_i = m^e \mod n_i$ para pessoa P_i. Finalmente, suponha que $k \geq e$. Mostre que um invasor que escuta todos os textos codificados pode recuperar a mensagem m (Dica: use o Teorema Chinês dos Restos).

Questão 7. Implemente o RSA em Python! Sua implementação deve ter (pelo menos) os seguintes componentes.

a. Uma função para gerar números primos. Sua função deve receber como entrada um natural n e gerar um número (provavelmente) primo p satisfazendo $10^n < p < 10^{n+2}$, sorteando p aleatoriamente no intervalo desejado e rodando 10 testes de Miller–Rabin com bases b aleatórias no intervalo $1 < b < p - 1$. (Naturalmente, p só deve ser aceito como provavelmente primo se todos os testes forem inconclusivos.)

b. Uma função chamada `gera_chaves` (por favor use este nome) para gerar chaves do RSA. Sua função deve usar sua função da letra **a** para gerar primos p e q, cada um com aproximadamente 75 algarismos, e retornar:

- $n = pq$
- algum número e inversível módulo $\phi(n) = (p - 1)(q - 1)$
- o inverso d de e módulo $\phi(n)$

Para uma solução realmente *completa*, sua função deve retornar também:

- p
- q
- o inverso de p módulo q
- o inverso de q módulo p
- a forma reduzida de d módulo $p - 1$
- a forma reduzida de d módulo $q - 1$.

c. Uma função chamada `encriptar` (por favor use este nome) que recebe como entrada uma string `texto` e números `n` e `e`, e retorna uma lista de números que seja uma sequência válida dos blocos numéricos resultantes da encriptação do `texto` com chave pública de módulo `n` e expoente `e`.

d. Uma função chamada `descriptar` (por favor use este nome) que recebe como entrada uma lista `blocos` e números `n` e `d`, e retorna a string resultante da descriptação da sequência de blocos usando a chave privada de módulo `n` e expoente `d`.

Para uma solução realmente *completa*, implemente a versão rápida de descriptação, usando a Questão 4 e os valores adicionais retornados pela função `gera_chaves`.

Use a transformação de símbolos em números dados na tabela ao final deste documento; você encontra a tabela em versões de dicionários de Python, um para conversão de símbolos em números e outra na direção contrária, em

`https://www.hugonobrega.com/codigo.py`

Como todos usaremos a mesma tabela de conversão, faremos uma troca de mensagens encriptadas ao vivo no horário da aula de 10/03 (começando às 9:00)! A participação é completamente livre e não conta para a avaliação de nenhuma forma.

Teste suas funções! Por exemplo, descripte a mensagem

```
[3323713707, 1319300010, 1144229290, 1660290264, 2194193588,
734058623, 2217413390, 3225084774, 781294944, 2728321168,
2631341137, 1922680560, 651540169, 3030894670, 41880099,
 300780045, 559287950, 1767066193, 2787757960, 520879703,
1993872416, 1386565567, 2460441503, 2766908703]
```

usando os parâmetros abaixo que forem relevantes:

```
n = 3401574593
e = 7
d = 971773783
p = 9533
q = 356821
inv_p_mod_q = 339154
inv_q_mod_p = 472
d_mod_p_menos_1 = 5447
d_mod_q_menos_1 = 152923
```

cód.	símb.
111	0
112	1
113	2
114	3
115	4
116	5
117	6
118	7
119	8
121	9
122	=
123	+
124	−
125	/
126	∗
127	a
128	b
129	c
131	d
132	e
133	f
134	g
135	h
136	i
137	j
138	k
139	l

cód.	símb.
141	m
142	n
143	o
144	p
145	q
146	r
147	s
148	t
149	u
151	v
152	w
153	x
154	y
155	z
156	á
157	à
158	â
159	ã
161	é
162	ê
163	í
164	ó
165	ô
166	õ
167	ú
168	ç
169	A

cód.	símb.
171	B
172	C
173	D
174	E
175	F
176	G
177	H
178	I
179	J
181	K
182	L
183	M
184	N
185	O
186	P
187	Q
188	R
189	S
191	T
192	U
193	V
194	W
195	X
196	Y
197	Z
198	Á
199	À

cód.	símb.
211	Â
212	Ã
213	É
214	Ê
215	Í
216	Ó
217	Ô
218	Õ
219	Ú
221	Ç
222	,
223	.
224	!
225	?
226	;
227	:
228	_
229	(
231	)
232	"
233	#
234	$
235	%
236	@
237	(espaço)
238	(nova linha)