Números Inteiros e Criptografia, PLE 2020

Trabalho Final

Submeta as soluções das questões marcadas com *
até 16 de novembro às 12:30 salvando um arquivo na sua pasta no Google Drive†

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

Como sempre, justifique todas as questões.

Testes de Primalidade

***Questão 1.** Seja $n = 3 \times 2^k + 1$ com $k > 1000$. O teste de Miller–Rabin em todas as bases possíveis $b < 20$ detecta que n é composto. Use esse fato para provar que $256^{1536} \not\equiv -1 \pmod{n}$.

***Questão 2.** Mostre que se um número n é pseudoprimo no teste de Miller–Rabin para a base b, então n é um pseudoprimo (no teste de Fermat) para esta mesma base.

***Questão 3.** Sejam $p < q$ dois primos. Suponha $n = pq$ e digamos que $p - 1$ e $q-1$ ambos dividem $n-1$. Mostre que $n-1 \equiv p-1 \mod (q-1)$ e obtenha uma contradição. Conclua que um número de Carmichael não pode ser o produto de apenas dois primos.

RSA: Questões teóricas

Questão 4. Seja $n = 938957$ a chave pública do RSA.

*** a.** Determine a fatoração de n sabendo-se que $\phi(n) = 937020$ (não pode usar um algoritmo de fatoração para resolver essa questão).

*** b.** Qual o menor valor de $d > 0$ para o qual d pode servir como chave secreta para uma implementação do RSA?

Questão 5. Considere $n = 19291$.

*** a.** Construa a menor chave pública e possível para o RSA usando o valor de n acima e determine a chave secreta correspondente.

†Link recebido por email em 1/9/2020 ou 17/9/2020. A pasta tem um nome similar a `Cripto - Submissões e Feedback - <seu nome>`; em caso de qualquer dúvida entre em contato com os professores.

*** b.** Codifique a mensagem `12345` usando sua chave pública.

***Questão 6.** Sejam p e q primos ímpares e digamos que estamos trabalhando com uma implementação do RSA com chave de codificação (n, e), onde $n = pq$. Pode acontecer que um bloco b de uma mensagem seja codificado como ele próprio nesta implementação. Ou seja, pode acontecer que $C(b) = b$. Um tal bloco será chamado de *invariante* pelo RSA com chave (n, e). Determine quantos são os blocos invariantes pelo RSA quando $p = 3, q > 3$ e $e = 3$ (Dica: use o exercício 8 da Lista 9).

***Questão 7.** O expoente $e = 2$ nunca deveria ser usado como chave pública. Por quê?

Teorema Chinês dos Restos

Questão 8.

*** a** (TCR, versão simplificada)**.** Sejam p e q primos entre si e $n = pq$.

Prove que, para quaisquer inteiros a e b, o sistema

$$\begin{cases} x \equiv a \mod p \\ x \equiv b \mod q \end{cases}$$

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

$$x \equiv (a \cdot q \cdot q') + (b \cdot p \cdot p'),$$

onde p' é o inverso de p em módulo q e q' é o inverso de q módulo p.

*** b** (TCR, versão completa)**.** Suponha que $p_1, p_2, \ldots, p_k$ sejam primos entre si, i.e., $\text{mdc}(p_i, p_j) = 1$ para todos $1 \leq i < j \leq k$, e seja $n = \prod_{i=1}^{k} p_i$.

Prove que, para quaisquer inteiros $a_1, a_2, \ldots, a_k$, o sistema

$$\begin{cases} x \equiv a_1 \pmod{p_1} \\ x \equiv a_2 \pmod{p_2} \\ \quad \vdots \\ x \equiv a_k \pmod{p_k} \end{cases}$$

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

$$x \equiv \sum_{i=1}^{k} (a_i \cdot q_i \cdot q_i'),$$

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

$$q_i = \frac{n}{p_i} = \prod_{\substack{j \in \{1,2,\ldots,k\} \\ j \neq i}} p_j$$

$$q_i' = \text{o inverso multiplicativo de } q_i \text{ em módulo } p_i.$$

***Questão 9** (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 suas chaves pública $n = pq$ & e, privada d, o usuário também calcula e guarda os seguintes valores:

- 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.

Ataques ao RSA

***Questão 10.** A mensagem `6803, 205, 1126, 1421, 1658` foi codificada pelo método RSA usando a senha $n = 7597$ e $e = 4947$. Além disso, sabe-se que $\phi = 7420$. Decodifique a mensagem.

***Questão 11.** Mostre que se você conhece as chaves públicas n, e e a chave privada d (e temos $e, d > 1$), então você consegue encontrar os primos p e q tais que $n = p \cdot q$, sem precisar fatorar n explicitamente.

***Questão 12.** Três pares (n, e) de chaves públicas,

$$(32333464105158123139761850953 9503, 3),$$

$$(37554017468380006506803029920 1351, 5),$$

$$\text{e } (42265968263874274411577354568 9701, 5)$$

foram geradas usando somente 5 números primos. Com essa informação, você consegue quebrar pelo menos uma das 3 chaves?

Questão 13.

*** a.** Como vimos, a segurança do RSA está, em parte, baseada no fato de que é difícil calcular *raízes* modulares em geral: dados $a \mod n$ e um natural e, é difícil 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 $x^e = a$.

Mostre que, se alguma solução $x^e \equiv a \mod n$ satisfaz

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

então é **fácil** encontrar x.

*** b.** Considere a situação em que k pessoas, $P_1, P_2, \dots, P_k$, 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 coprimos entre si. Agora suponha que Maria codifique a mesma mensagem m para cada pessoa: temos $0 \leq m \leq \min\{n_1, n_2, \dots, n_k\}$ 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).

RSA: Implementação

Questão 14. 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, sendo p com aproximadamente 50 algarismos e q com aproximadamente 100 algarismos, e retornar:

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

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 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 9 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 durante a aula prática de 16/11.

Teste suas funções!

Por exemplo, se você usou `gera_chaves` e obteve `n`, `e`, `d` como chaves pública e privada, então você deve obter, no interpretador do Python:

```
>>> descriptar(encriptar('cósmicos',n,e),n,d)
'cósmicos'
```

cód.	símb.	cód.	símb.	cód.	símb.	cód.	símb.
111	0	141	m	171	B	211	Â
112	1	142	n	172	C	212	Ã
113	2	143	o	173	D	213	É
114	3	144	p	174	E	214	Ê
115	4	145	q	175	F	215	Í
116	5	146	r	176	G	216	Ó
117	6	147	s	177	H	217	Ô
118	7	148	t	178	I	218	Õ
119	8	149	u	179	J	219	Ú
121	9	151	v	181	K	221	Ç
122	=	152	w	182	L	222	,
123	+	153	x	183	M	223	.
124	−	154	y	184	N	224	!
125	/	155	z	185	O	225	?
126	∗	156	á	186	P	226	;
127	a	157	à	187	Q	227	:
128	b	158	â	188	R	228	_
129	c	159	ã	189	S	229	(
131	d	161	é	191	T	231	)
132	e	162	ê	192	U	232	"
133	f	163	í	193	V	233	#
134	g	164	ó	194	W	234	$
135	h	165	ô	195	X	235	%
136	i	166	õ	196	Y	236	@
137	j	167	ú	197	Z	237	(espaço)
138	k	168	ç	198	Á	238	(nova linha)
139	l	169	A	199	À		