COMP 70009 Cryptography Engineering: Spring 2024
Sole Assessed Coursework

To be completed and submitted in groups of 1-4 students

Due: Wednesday 14 February 2024, 19:00

Marking scheme: Questions1(a) has 5 marks. Questions 1(b)-(c) have 7 marks each. Question 1(d) has 6 marks. Questions 2(a)-(b) have 8 marks each. Questions 2(c)-(e) have 9 marks each. Questions 3(a)-(c) have 8 marks each. Questions 3(d)i. and 3(d)ii. have 4 marks each.

1. This question is about Shannon's notion of perfect secrecy and conditional probabilities needed for that notion.

   Let $\mathbb{P} = \{a, b, c, d, e\}$, $\mathbb{C} = \{1, 2, 3, 4, 5\}$, and $\mathbb{K} = \{k_1, k_2, k_3, k_4, k_5\}$. Let $p(P = a) = 0.18$, $p(P = b) = 0.19$, $p(P = c) = 0.22$, $p(P = d) = 0.21$, and $p(P = e) = 0.2$. Let $p(K = k_i) = 0.2$ for all $1 \leq i \leq 5$. Finally, the encryption $e_k(m)$ for this is given in the table

	a	b	c	d	e
k_1	4	2	1	3	5
k_2	5	1	4	2	3
k_3	1	3	2	5	4
k_4	2	4	5	1	3
k_5	2	5	3	4	1

   (a) Compute $p(C = 2)$.

   (b) Compute $p(C = 4 \mid P = e)$.

   (c) Compute $p(P = c \mid C = 3)$.

   (d) Is the above crypto-system perfectly secure? Justify your answer.

2. This question is about understanding practical ways of making RSA encryption non-deterministic.

   To answer this questions, first watch the videos "Naive-RSA-And-Its-Correctness" and "RSA-security" in the sub-folder *optional-videos* of our recorded mini-lectures; these videos explain the naive RSA public-key cryptosystem and discuss its security, respectively.

   Then consider the Python code in Figure 1, as a practical solution to the key-distribution problem for a session key $\mathtt{K}$ that will be used in symmetric-key encryption subsequently.

   The modulus $N = p \cdot q$ is the product of large primes p and q, the public key e equals 5.

(a) State what function `generateRSAPrime` returns, and explain the roles of the assert statements in its code.

(b) State what function `generateRSAKey` returns, and explain the roles of assert statements in its code.

(c) Function `encryptRandomKeyWithRSA` returns a session key and a ciphertext. Explain why you think the definition of `k` is secure enough for generating a random element `r` in RSA plaintext space, and why the derivation of key `K` is secure.

(d) Function `decryptRandomKeyWithRSA` assumes knowledge of the secret key d and of the ciphertext returned by function `encryptRandomKeyWithRSA`. Explain the role of the assertion, and why this decryption successfully recovers the session key `K` for the correct values of N, d, and c.

(e) *Briefly* sketch how the functions in Figure 1 could be used to share a session key across an insecure communication channel.[1]

3. This question is about gaining a first, *non-technical*, understanding of Post-Quantum Cryptography Standards.

(a) In 1997, the US National Institute of Standards and Technology (NIST) held a *competition* for its Advanced Encryption Standard (AES). In 2016, its call for proposals for post-quantum cryptography (PQC) standards spoke about a *competition-like* process. Research online and briefly report why NIST wanted this to be *like* a competition but not a competition as we normally understand the term.

(b) Explain briefly why, in the PQC call for proposals, NIST did focus on public-key encryption, key-exchange mechanism, and digital signature and not also on hash functions and symmetric encryption algorithms.

(c) In your brief assessment, will the adoption of NIST PQC standards such as Module-Lattice-based Key-Encapsulation Mechanism (ML-KEM) require that existing internet protocols such as Transport Layer Security (TLS) are replaced with completely new security protocols? Related to that, describe briefly how NIST seems to see this.

(d) In the PQC call for proposals, NIST speaks of 5 security levels (1 through to 5) of increasing strength:

i. Explain briefly why levels are increasing in that an attacker is expected to invest more computational resources for compromising the security of larger levels.

ii. Explain briefly why the security levels 2 and 4 are relevant for digital signature schemes, e.g., by relating this to a security property of digital signatures.

[1] This is not asking about details of the used communication protocol; this is about understanding which parties would use which functions to generate what data, and who sends which data to whom.

```
def generateRSAPrime(number_of_bits:int):
    assert 1024 <= number_of_bits & number_of_bits <= 4096
    r = 100 * number_of_bits
    x = 4
    while rabin_miller_primality_test(x,50) == False or x % 5 == 1:
        r = r-1
        assert r > 0
        x = randbits(number_of_bits)
    return x

def generateRSAKey(number_of_bits:int):
    assert 2048 <= number_of_bits & number_of_bits <= 8192
    gcd = 2
    while gcd != 1:
        p = generateRSAPrime(number_of_bits // 2)
        q = generateRSAPrime(number_of_bits // 2)
        assert p != q
        t = (p-1) * (q-1)
        gcd, _, _ = extended_Euclid(5,t)
    u = multiplicative_inverse_modulo(5,t)
    d = u % t
    return p, q, p*q, d

from hashlib import sha3_256 as h
from math import floor, log2

def encryptRandomKeyWithRSA(N:int):
    k = floor(log2(N))
    r = randbits(k)
    K = h(h(str(r).encode()).hexdigest().encode()).hexdigest()
    c = pow(r,5,N)
    return K, c

def decryptRandomKeyWithRSA(N:int, d:int, c:int):
    assert 0 <= c & c < N
    i = pow(c,d,N)
    K = h(h(str(i).encode()).hexdigest().encode()).hexdigest()
    return K
```

Figure 1: Python code for generation of RSA primes, RSA keys, and RSA decryption/encryption. You can run this code in a notebook available in our notebook space; that notebook contains implementations of auxiliary functions such as `rabin_miller_primality_test`