90/100

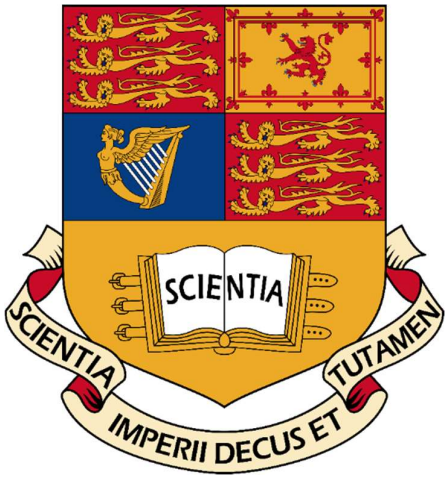
70009 11

Sole coursework for C70009

Submitters

tw723	Tee Wen Hui
gea23	Guillermo Arcos Ibanez
ky523	Kangle Yuan
wh1923	Wai Hooi

Emarking



Imperial College London

70009 Cryptography Engineering

Sole Assessed Coursework

Name	CID
Guillermo Arcos Ibáñez	02541386
Kangle Yuan	02472956
Hooi Wai Kit, Marcus	02542289
Tee Wen Hui	02535839

Q1(a)

p(C=2)

= p(K=4) * p(P=a) + p(K=5) * p(P=a) + p(K=1) * p(P=b) + p(K=3) * p(P=c) + p(K=2) * p(P=d)

= 0.2 * (0.18 + 0.18 + 0.19 + 0.22 + 0.21)

= 0.196

Q1(b)

p(C=4 | P=e) = p(K=3) = 0.2

Q1(c)

Compute p(C=3| P=c) = p(K=5) = 0.2

Compute p(C=3)

= p(K=3) * p(P=b) + p(K=5) * p(P=c) + p(K=1) * p(P=d) + p(K=2) * p(P=e) + p(K=4) * p(P=e)

= 0.2 * (0.19 + 0.22 + 0.21 + 0.2 + 0.2)

= 0.204

So:

p(P=c | C=3)

= (p(P=c) * p(C=3| P=c)) / p(C=3) by Bayes Theorem

= (0.22 * 0.2) / 0.204

= 0.2157

Q1(d)

This is not perfectly secure, because p(C=3| P=c) = 0.2 and p(C=3) = 0.204. They are inequivalent, which doesn't satisfy the criteria of perfect secrecy that p(C=c| P=m) = p(C=c). Besides, both k4 and k5 can encrypt plaintext a to ciphertext 2, which doesn't meet the criteria of S2 in the definition of perfect secrecy. Consequently, this is not a perfect secrecy.

Q2(a)

generateRSAPrime returns a randomly generated prime number that is of size between 1024 bits to 4096 bits. The first assert statement ensures that the prime number generated falls within 1024 bits to 4096 bits. This is important to ensure that the number is sufficiently large. The second assert statement ensures that r is always a positive value. All prime numbers should be positive.

Q2(b)

generateRSAKey returns the prime numbers p and q, their product p*q, and the private key d for RSA encryption. The first assert statement in the code validate that the input number_of_bits is between 2048 and 8192 bits. This range is twice that of the input for generateRSAPrime, because a floor division will be performed before passing this variable into the generateRSAPrime function. The second assert statement ensures that that p and q are distinct primes. If p and q are not distinct, it is easier to factorise the product pq and compromise the security of the encryption.

Q2(c)

The definition of the variable 'k' is considered secure for generating a random element 'r' in the RSA plaintext space because it calculates the number of bits required to represent the modulus 'N' securely. Logarithm base 2 (log2) of 'N' provides the number of bits required to represent 'N' in binary. Taking the floor of this value ensures that 'k' is an integer, representing the exact number of bits needed for secure representation. k is then used to create a random value of r using randbits of the correct bit size, ensuring that r is random.

The derivation of the key 'K' in the function is secure because element 'r' is processed with a double hash function (SHA3-256) to create 'K'. This enhances the security by introducing cryptographic hash functions to obfuscate the relationship between 'r' and 'K'. It is harder and near impossible to reverse engineer the key from the derived value.

Q2(d)

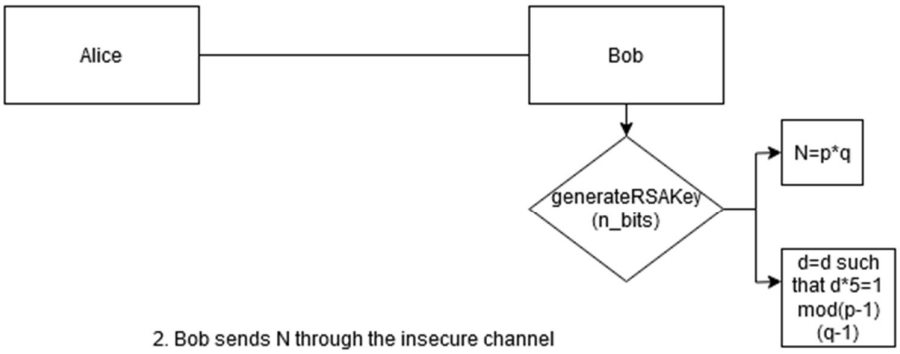
The assertion makes sure that the cypher text has a value between [0,N) to make sure that we are dealing with a correct cypher text. We first decrypt the cypher text, by doing c^d modulus N,as the RSA cypher decrypts cyphertext by doing this operation, necessitating the knowledge of d and c. We also assume that d has to be a value that satisfies d*5=1 mod (p-1)*(q-1) according to the RSA cypher, and thanks to Lagrange's theorem we know that the decryption will be correct.

This operation returns m, which in this case is the plaintext encrypted in the previous function. However, in the previous function the plaintext is derived from creating a random integer with log2(k) random bits(This variable is r). Therefore, r and m represent the same value in both functions so if we apply the same hashing operation to m that was applied to r in encryptRandomKeyWithRSA we will get the same key K.

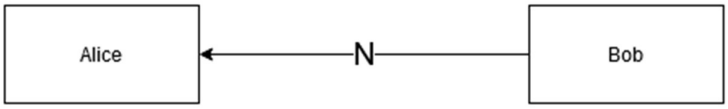
Q2(e)

Alice and Bob want to share a key through an insecure channel

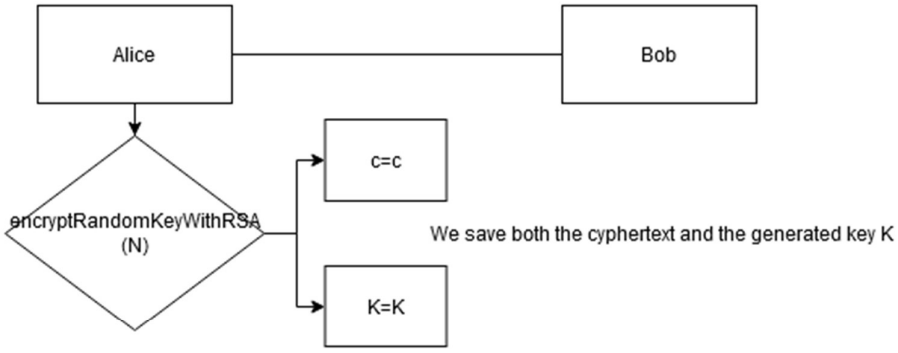
1.Bob generates a new RSA key with his desired number of bits.
He saves N(p*q) and assigns d to the value that compliments e
(In this implementation e=5)



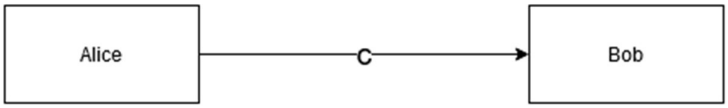
2. Bob sends N through the insecure channel



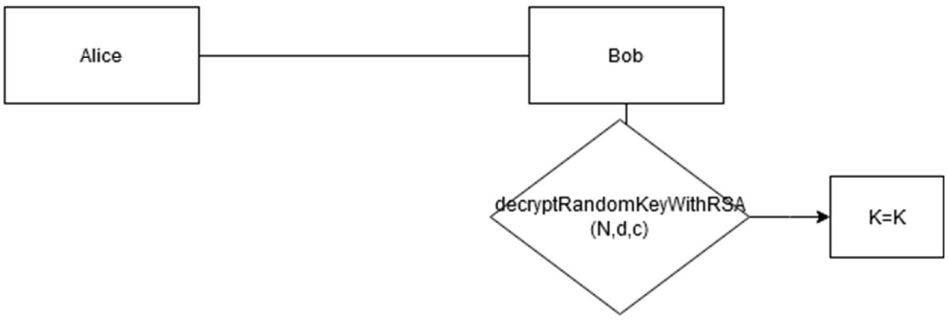
3. Alice creates and encrypts a random key with encryptRandomKeyWithRSA and the received N as parameter



4. Alice sends the generated cyphertext through the channel to Bob



5.Bob decrypts the cyphertext using his secret d parameters to obtaing the session key



Now both Alice and Bob have access to the same session key K
and since parameter d was never shared,a would be attacker cannot feasibly decrypt it

Q3(a)

NIST structured the call for post-quantum cryptography (PQC) standards as a competition-like process to encourage the development of new cryptographic algorithms that are resistant to quantum attacks. However, it wasn't a traditional competition with a single winner; instead, it was a collaborative effort to explore various candidate algorithms and select those that demonstrate security against quantum threats. The intention was to foster a diverse range of approaches and solutions, ensuring that the selected algorithms would be robust and effective in different scenarios. This approach aimed to avoid a monoculture in cryptographic standards, where a single algorithm could become a target for potential vulnerabilities or attacks.

Q3(b)

The focus on public-key encryption, key-exchange mechanisms, and digital signatures in the PQC call for proposals is because these cryptographic primitives are most vulnerable to attacks by quantum computers. Shor's algorithm, for example, has the potential to efficiently break widely used public-key cryptosystems like RSA and ECC when executed on a large enough quantum computer. Therefore, finding post-quantum alternatives for these specific primitives is crucial to maintaining the security of communication systems.

NIST did not explicitly focus on hash functions and symmetric encryption algorithms in the PQC competition because these primitives are believed to be less susceptible to quantum attacks. Quantum computers are not expected to provide a significant advantage in breaking hash functions or symmetric encryption compared to classical computers. This is because one-way hash functions, such as SHA-256, are designed to be computationally infeasible to reverse, making them less susceptible to quantum algorithms like Shor's. Symmetric encryption relies on shared secret keys, and while quantum computers may provide a speedup for certain key search algorithms like Grover's, the impact is generally less severe than the threat posed to public-key systems.

Q3(c)

NIST is actively working on soliciting, evaluating, and selecting post-quantum cryptographic algorithms, including key exchange and digital signature schemes, as part of their Post-Quantum Cryptography Standardisation project. While some internet protocols will need to be replaced, it is more likely that most will undergo some degree of transition either involving a gradual replacement or upgrade of cryptographic algorithms within existing protocols rather than a complete overhaul.

TLS relies on the handshake process and key exchange algorithms for secure communication, and current algorithms such as RSA and ECC (Elliptic Curve Cryptography) are vulnerable to quantum attacks. Such an illustration is Shor's algorithm, a quantum algorithm, can efficiently factorise large numbers and solve the discrete logarithm problem, which are the underlying hard problems for RSA and ECC, respectively. The integration of PQC algorithms, like those based on lattice cryptography, hash-based cryptography, or code-based cryptography, may necessitate modifications to the TLS handshake process, key exchange mechanisms, and signature schemes.

NIST's approach is to ensure a smooth transition to post-quantum cryptography by providing guidance and standards for implementing quantum-resistant algorithms in existing protocols. The goal is to avoid disruptions to the security of internet communication when quantum computers become a practical threat. NIST is likely considering a phased approach to integration. The adoption process may involve developing hybrid solutions that support both classical and post-quantum algorithms during the

transition period. This approach allows systems to maintain compatibility with existing infrastructure while preparing for the quantum era.

Q3(d)

(i)

In this case, NIST defines 5 levels of security strength that a system can reach. Each level corresponds to an amount of computational power that a would be attacker would need to successfully break relevant information. Level 1 only requires the attacker to be able to break AES128 with a 128-bit key, this is possible to do with a computational complexity of 2^126.. Meanwhile, the second level requires to break a hash function of 256 bits and so on and so forth, alternating between hash function and block cipher each with their number of required bits increasing . Its therefore obvious why they are ordered this way,as the number of bits in both the hash function and key grow,so does the computational complexity of finding the key. Its also important to know that for SHA-2 there exist only theoretical attacks.

(ii)

Hash based digital signature scheme are built to be resilient to pre-image and second pre-image attack. We know that collision resistance implies second pre-image resistance(Since collision resistance says that it is infeasible to find x and x' such that h(x)=h(x') and a second pre-image attack requires to find that). Therefore a digital security scheme that seeks to be resilient to these kinds of attacks must necessarily have a hash function that is collision resistant.