Assignment 3: COMP6453 2024 T2

This set consists of 3 questions with a total marks of 60. Before you start read about Chinese Remainder Theorem, if you have forgotten it.

Change log

- Question 2, **Verify**: (C, x, π): Check if $C = g^x \cdot h^r$.
  h^r instead of b^r.
- Question 3, pk and sk has been explicitly specified and inputs to the functions now include sk and pk.

1 RSA Encryption

Three users have RSA public keys $<N_1, 3>$, $<N_2, 3)$, and $<N_3, 3>$ (they all use $e = 3$), with $N_1 < N_2 < N_3$. Consider the following method for sending the same message $m \in \{0,1\}^l$ to each of these parties: choose a uniform $r \leftarrow Z^*_{N_1}$, and send to everyone the same ciphertext

$$\langle [r^3 mod N_1], [r^3 mod N_2], [r^3 mod N_3], H(r) \oplus m \rangle,$$

where $H : Z^*_{N_1} \to \{0,1\}^l$. Assume $l >> n$, n is the security parameter.

(a) Show that this is not CPA-secure, and an adversary can recover m from the ciphertext even when H is modeled as a random oracle.

(b) Show a simple way to fix this and get a CPA-secure method that transmits a ciphertext of length $3l + O(n)$.

(c) Improve your approach such that the encryption scheme is still CPA-secure but with a ciphertext of length $l + O(n)$.

Marks

- Part (a): 5
- Part (b): 4
- Part (c) : 5
- Total: 14 marks

2 Commitment Scheme

A *commitment scheme* consists of the algorithms **Setup, Commit, Reveal, Verify** and enables a committer to commit a value x to a verifier. The scheme is hiding if the commitment C does not reveal any information about x. Later, the committer may present/reveal a proof convincing the verifier value committed in C is x. The scheme is binding if the committer cannot convince the verifier that the committed value in C is some $x' \neq x$. Here is an example scheme:

Setup: Publicly output a group G with $|G| = q$ where q is an odd prime. Two generators $g, h \in G$ are also made public.

Commit(x): To commit to $x \in \mathbb{Z}_q$, choose a random $r \in \mathbb{Z}_q$ and output $C = g^x \cdot h^r \in G$ where $\cdot$ represents the group operation.

Reveal(x): output the proof $\pi = (x, r)$

Verify(C, x, π): Check if $C = g^x \cdot h^r$.

(a) Show this scheme is both binding and hiding.

(b) Show this scheme satisfies the following *homomorphic property*: given commitments C_x, C_y to $x, y \in \mathbb{Z}_q$ respectively, one can compute a commitment C_z, $z = ax + by$ for any $a, b \in Z_q$ without knowledge of the secret randomness r used to generate C_x and C_y.

Marks

- Part (a): (4 + 3)= 7 marks
- Part (b): 5
- Total: 12 marks

3 RSA Accumulators

A cryptographic accumulator is a compact data structure for representing a set of elements belonging to some domain. An example of an accumulator is a Merkle tree. Recall that a Merkle tree can commit to an ordered tuple of elements $(x_1, x_2, \cdots, x_n) \in \mathcal{S}$. One can later prove that x_i is the value at position i, for any $1 \leq i \leq n$, where each such proof is a sequence of $\log_2 n$ hashes. An accumulator allows for a compact proof of membership of element x in the data structure.

The aim of this exercise is to implement an RSA accumulator.

The accumulator's public key pk is an RSA modulus $n = pq$ ($sk = (p, q)$ are large primes and secret). An initial value, $acc_{\varnothing} \leftarrow Z_n^*$ that corresponds to the empty set is also picked. The value $acc_{\mathcal{S}} = acc_{\varnothing}^{\Pi_{x \in \mathcal{S}} x}$ represents the accumulator for a set $\mathcal{S}$. For now, let us think of elements of $\mathcal{S}$ as positive integers. We say that $acc_{\mathcal{S}}$ is the accumulator for $\mathcal{S}$. The witness w_x that $x \in \mathcal{S}$ is the value $w_x = acc_{\varnothing}^{\Pi_{x' \in \mathcal{S}, x' \neq x} x'}$. To verify that $x \in \mathcal{S}$ using this witness, check that $(w_x)^x = acc_{\mathcal{S}}$.

Part A

Your task is to write the pesudocodes for the following functions. After you have written pseudocodes, implement these functions taking the help of OPENSSL library.

(a) createAccumulator($\mathcal{S}, pk$): Creates an accumulator $acc_{\mathcal{S}}$ for the set of elements $\mathcal{S}$.

(b) genWitness ($\mathcal{S}, x, acc_{\mathcal{S}}, sk$): Takes a set of elements $\mathcal{S}$, the corresponding accumulator $acc_{\mathcal{S}}$ and generates an witness w_x for the element x.

(c) verifyWitness ($x, acc_{\mathcal{S}}, w_x, pk$): Takes the accumulator $acc_{\mathcal{S}}$, witness w_x for the element x and returns 1 if the witness is correct and 0 otherwise.

(d) addElement (($\mathcal{S}, x, acc_{\mathcal{S}}, pk$)): Takes a set of elements $\mathcal{S}$, the corresponding accumulator $acc_{\mathcal{S}}$, a new element x and outputs the new accumulator value $acc_{\mathcal{S} \bigcup \{x\}}$.

(e) updateWitness (($\mathcal{S}, x, acc$), sk): Takes a set of elements $\mathcal{S}$, the corresponding accumulator $acc_{\mathcal{S}}$, a new element x and update witnesses w_y, $\forall y \in \mathcal{S} \bigcup \{x\}$.

(f) batchUpdate (($\mathcal{S}, \mathcal{X}, acc$), sk): Takes a set of elements $\mathcal{S}$, the corresponding accumulator acc, adds new set of element $\mathcal{X}$ and updates accumulator value $acc_{\mathcal{S} \bigcup \mathcal{X}}$ and witnesses for all elements in $x \in \mathcal{S} \bigcup \mathcal{X}$.

Part B

Compare the (1) asymptotic time complexity to verify membership, (2) size (in Bits) of the accumulation value and (3) size of witness (in Bits) of RSA accumulator with Merkle Tree accumulator.

(4) How easy is it to update a Merkle tree accumulator with a new node? Compute the time complexity for update when a new element is added.

Marks

- Part A (a-c): (1 + 2)X3 = 9, (1 for pseudocode, 2 for coding)
- Part A (d-f): (2+3)X3 = 15 (2 for pseudocode, 3 for coding)

- Part B: $2 + 2 + 2 + 4 = 10$ marks.
- Total: 34 marks

Full Submission

1. Q1: Write answer in a file q1. You can use pdf or txt format.
2. Q2: Write answer in a file q2. You can use pdf or txt format.
3. Q3: Put the answers of the Parts A and B, in a pdf or txt file, and code and headers Part A all in a folder.
4. Upload a zip file with all three answers. $\langle zid \rangle \langle ass3 \rangle .zip$