COMP4901W Homework 1

KIM, Jaehyeok (20550178)

The Hong Kong University of Science and Technology

Exercise 1

1. Secure Protocol Design

Step 1. Alice first makes her choice $c_A \in \{r, p, s\}$ (rock, paper, and scissors) and chooses a nonce n_A that is a random large string.

Step 2. Alice computes $h(c_A \mid n_A)$ where $h()$ is a hash function and $\mid$ is just a separator.

Step 3. Alice computes her key pair (e_A, d_A) such that $Dec_{d_A}(Enc_{e_A}(m)) = m$ where e_A is her public key and d_A is her private key for digital signature.

Step 4. Alice signs $h(c_A \mid n_A)$ using d_A by computing $m_A = Enc_{e_A}(h(c_A \mid n_A))$ and sends this with $h(c_A \mid n_A)$ to Bob. This will take at most 1 minute.

Step 5. After Bob receives m_A and $h(c_A \mid n_A)$ from Alice, he decrypts m_A using Alice's public key e_A by computing $Dec_{d_A}(m_A)$ and compare with $h(c_A \mid n_A)$ to verify that it is sent from Alice.

Step 6. Bob then makes his choice c_B and computes (e_B, d_B) such that $Dec_{d_B}(Enc_{e_B}(m)) = m$ for digital signature.

Step 7. Bob signs c_B using e_B by computing $m_B = Enc_{e_B}(c_B)$ and sends m_B and c_B to Alice. This will take at most 1 minute as well.

Step 8. After Alice receives m_B, she can decrypt it using Bob's public key d_B by computing $Dec_{d_B}(m_B)$ and compare it with c_B to verify if it is sent from Bob.

Step 9. Alice then sends c_A and n_A to Bob and this takes at most 1 minute. Bob can use these values to compute $h(c_A \mid n_A)$ by himself to verify the received c_A. If the hash values are different, Alice will lose the game immediately.

Step 10. After at most 3 minutes, the winner will be decided based on the shared choices.

2. Can either party cheat? Can cheating be detected?

The secure protocol proposed above prevents both parties to cheat appreciating the useful properties of the hash function. Bob cannot cheat since he cannot guess Alice's choice given $h(c_A \mid n_A)$ due to the hiding property of the hash function. Alice also cannot cheat since she cannot modify her choice as the hashed values has already been sent. After Alice checks that she lost in step 8, she still has to send her original choice. Otherwise, Bob will notice that she has sent a different choice value by having two different hash values due to the collision-resistance property of the hash function. Moreover, Alice cannot figure out a new random large string nonce that is mapped to the previous hash value along with the modified choice due to the collision-resistance property.

3. Prove that the protocol is immune to tampering by the players or any third party

The designed protocol is immune to tampering by Alice, Bob, or any third party on the network. Both Alice and Bob cannot modify their choices as proven above, otherwise it will be detected due to the collision-resistance property of the hash function. In addition to that, tampering by third party can easily be detected since the protocol utilizes the non-forgeable property of RSA digital signature. When both Alice and Bob send any information to each other, they send the information along with a signed version. The signature is created by using their own private key that cannot ideally owned by any third party on the network. Therefore, by verifying the information delivered from each other using public key, the players can assure that the information is not tampered by any third parties.

With these following properties, no party can cheat or tampering the result in the proposed protocol or all cheats can be detected assuming the properties of the hash function hold.

Exercise 2

1. Is this a valid symmetric encryption scheme?

Since the encryption scheme uses the same shared key k for both encryption and decryption, this is a symmetric encryption scheme. Moreover,

$$\begin{aligned}
Dec_k(Enc_k(m)) &= (((m+k) \bmod p) - k) \bmod p \\
&\text{Since } k < p, \\
&\equiv (((m+k) \bmod p) - (k \bmod p)) \bmod p \\
&\text{Based on the modular arithmetic rules,} \\
&\equiv ((m+k-k) \bmod p) \bmod p \\
&\equiv (m+k-k) \bmod p \\
&\equiv m \bmod p \\
&\text{Since } m < p, \\
&\equiv m
\end{aligned}$$

Therefore, it is a valid symmetric encryption scheme.

2. Is this scheme secure for a one-time communication?

From a one-time communication, Eve will have access to the public information Enc, Dec, p, e. Based on the assumption that $m \leq p$, $m \in \{1, \ldots, p\}$, e $(= (m+k) \bmod p)$ will also be $\in \{0, 1, \ldots, p-1\}$. Thus, $\{m, m+1, \ldots, m+p-1\} \equiv \{0, 1, \ldots, p-1\} \bmod p$. Similar to the discrete algorithm in the lecture, it is going to take $O(p)$ where p is a huge prime number, thus it will be impossible to compute m in the lifetime of the universe. The probability of guessing the correct message stays at $1/p$ which is close to 0. In other words, knowing a single encrypted message does not disclose any information about the message.

3. Is it secure for multiple rounds of communication?

The suggested scheme is not as secure as used for a one-time communication if it is used for multiple rounds of communication with the same secret key k. When Eve collects two encoded messages $e_1 = m_1 + k$ and $e_2 = m_2 + k$, she can acquire narrowed-down information about the messages by subtracting those two: $e_1 - e_2 = (m_1 + k) - (m_2 + k) = m_1 - m_2$. This information can potentially be used for figuring out the original messages using some sort of statistical analysis. Therefore, multiple rounds of communication is not secure.

Moreover, since p is a huge prime number, it is likely to be $m \ll p$ and there exists more security vulnerability in this case. When sufficient number of $\{e_t\}_{t=1}^{N} = \{k + m_t\}_{t=1}^{N}$ are collected by Eve, she can roughly deduce that the secret key k would be the lowest value of them. The reason is that the encryption is simply a translation by k. By having the rough prediction for the secret key k value, the potential range of the encrypted message gets narrowed down by a great scale. This definitely discloses some information about the message to Eve.

To prevent this to happen, Alice and Bob can update or change the secret key k in every single communication. For example, they can utilize the encrypted message e and the current secret key

k to create a new secret key using the same secret rule. If the secret key is continuously changed, each communication will be one-time communication, thus there would never be sufficient number of encrypted messages to make the rough deduction.

Exercise 3

1. Protocol

The following protocol is based on zero-knowledge proof.

1. Alice chooses a random number r from a range $[0, p-2]$ to satisfy the condition of being the primitive root.
2. Alice computes $m^* = g^r \bmod p$ and disclose it to Bob.
3. Bob then randomly asks Alice either r or $(k+r)\ mod\ (p-1)$.
   – If Bob asked for r,
       he can verify it by comparing whether $g^r \bmod p$ is equal to m^*
   – If Bob asked for $(k+r)\ mod\ (p-1)$,
       he can verify it by checking whether the equations below hold or not.

$$\begin{aligned} g^{(k+r)\ mod\ (p-1)} \bmod p &= (g^{k\ mod\ (p-1)} * g^{r\ mod\ (p-1)}) \bmod p \\ &= (g^k \bmod p) * (g^{r\ mod\ (p-1)} \bmod p) \\ &= m * m^* \end{aligned}$$

4. Repeat steps 1∼3 for n times where n is a huge number. For each iteration from step 1 to 3, the probability ρ that Bob is convinced that Alice actually knows k is 0.5. Assuming that Alice is trying to fake Bob, the probability that Alice could successfully cheat on all iterations is $(0.5)^n$ if the protocol is repeated n times. The probability converges to 0 when n approaches to ∞. Therefore, Bob would be able to believe that Alice is actually possessing k after a sufficient number of repetition.

2. Information disclosure

It is impossible for Bob to obtain k from r or $(k+r) \bmod p-1$. Since r is a randomly chosen number and changed in every iteration, so it does not disclose any information about k as there is no relation. Moreover, it is impossible to deduce k given $(k+r) \bmod p-1$ without knowing the random number r that is changed in every iteration. Therefore, Bob can never obtain k from this protocol.