CHALMERS UNIVERSITY OF TECHNOLOGY
Dept. of Computer Science

Home Assignment 3, Cryptography course

This assignment consists of two main parts:

(A) Implementing a decentralised voting protocol using Secure Multi-Party Computation (SMPC) techniques.

(B) Finding and Fixing vulnerabilities in protocols.

Reporting

Your report should be written in English in a plain ASCII file. In part (B), you must describe the attacks using the common notation for protocols given in this assignment, in the lectures and the course book.

Part (A): Decentralised Voting Protocol

Every year the Chalmers' Cryptography Society has a Christmas activity. They either go to the Christmas market at Liseberg, or to a Christmas concert in Konserthuset. Which of the two is decided by a simple anonymous voting procedure.

Each society member m_i sends an e-mail with their vote to a trusted third party A, called auditor. The auditor, who is outside of the society, sums up the votes and sends a mass e-mail to the society members announcing the outcome choice.

This procedure went fine until last year, when A turned out to be friends with one of the society members. The auditor lied about outcome, sending the society to Liseberg (the choice of A's friend) even though the majority voted for Konserthuset.

This year, the Chalmers' Cryptography Society decides to apply their knowledge of cryptographic protocols. The members propose to use Secure Multi-Party Computation to establish the result of the votes. They find three auditors, A_1, A_2 and A_3. Each society member then divides his or her vote between these auditors using Shamir's Secret Sharing Scheme.

Question 0: *Write down your birthday as $Y_1Y_2Y_3Y_4M_1M_2D_1D_2$ (we use it in the following questions as a "random value").*

Question 1: Let m_1, m_2, m_3 and m_4 be the members of the society casting their vote. The votes are encoded as 1 for the Christmas market, and -1 for the Christmas concert. *For each $i \in \{1, 2, 3, 4\}$ member m_i choose a vote $v_i \in \{-1, 1\}$ and construct a 2-degree polynomial based on the digits in your birth-date in the following way:*

$$m_1(x) = v_1 - Y_1x + M_1x^2$$
$$m_2(x) = v_2 - Y_2x + M_2x^2$$
$$m_3(x) = v_3 - Y_3x + D_1x^2$$
$$m_4(x) = v_4 - Y_4x + D_2x^2$$

Write down your polynomials $m_i(x)$ explicitly (do not forget the minus signs!) and the e-mails (values) sent by each member (e.g. m_i sends to A_j the share $m_i(j)$).

Question 2: At this point, each auditor A_j, $j \in \{1, 2, 3\}$, separately computes the sum of the votes and emails this sum back to each society member. *Explain how the a_j values are obtained and compute what each auditor returns to the society members.*

Question 3: The society members can use the values a_j sent by the auditors to compute the sum of their votes. *Write the computations performed by each member to obtain the final result.* If the sum is positive it means that the Chalmers' Cryptography Society will go to Liseberg, if it is negative they all go to the concert. If it outcome is zero (tie), they all stay at home and read the nutcracker.

The incentive for auditors to report a wrong value for the a_j is low: unless all A_1, A_2 and A_3 cooperate in the lie they have no control on what the outcome of the vote will be (you can try this to convince yourself of that). Unfortunately, it turns out that it is very easy for the society members themselves to cheat.

Question 4: *Assuming that all auditors and all members other than m_1 are honest, show how member m_1 can make sure that the final sum is positive by not following the protocol in the construction of the polynomial m_1.*

Bonus Question 5 (optional): *Suggest a way to prevent m_1 from cheating.*

Part B: Cryptographic Protocols

In this part you will break some simple, faulty cryptographic protocols, by demonstrating attacks against them. Note that what you will have to find are attacks against the protocol, not against the encryption primitives. So we assume throughout that an adversary can not break encryption. On the other hand, he can eavesdrop and send messages, intrude in the middle, manipulate address fields so that receivers believe that his messages come from other parties, etc.

Task 1

The setting is as follows. An organization employs RSA public key cryptography, so each user has a key pair (e, d), consisting of a *public encryption* key e and a *private decryption* key d. We use the notation that keys e_A and d_A denote encryption and decryption keys for user A etc. The organization has a working public key infrastructure, so all employees (or, rather, their software) can obtain authentic keys for all users. This means that we will not need to consider attacks based on the adversary trying to fool users to use wrong keys.

We start with a naive protocol, which is intended to provide acknowledgement of the receipt of an encrypted message M by the intended receiver.

1. $A \rightarrow B : A, \{M\}e_B$
2. $B \rightarrow A : \{M\}e_A$

To send a message M to B, A encrypts M using B's public key. On receipt, B decrypts and, as acknowledgement, sends M back to A, encrypted for A. A now decrypts, compares with what she originally sent and concludes that B has received M.

However, this protocol is flawed; an adversary C within the organization that eavesdrops on the communication can easily use a message he picks up to set up a new run of the protocol himself and get M decrypted and encrypted for himself. You should first convince yourself that you understand this attack (see the solutions for the exercise session on protocols if necessary).

The problem is fixed by changing the protocol to

$$\begin{aligned} &1.\ A \rightarrow B : \{A, M\}e_B \\ &2.\ B \rightarrow A : \{M\}e_A \end{aligned}$$

You should (try to) convince yourself that this is secure if the adversary cannot break encryption and keys are authentic. However, in view of the many seemingly secure protocols that turn out to be broken, your level of confidence might not be very high.

You do not need to report anything of the above preparations. Now, for your first task. A newly employed, ambitious cryptographer proposes to strengthen the protocol further by adding one more layer of encryption:

$$\begin{aligned} &1.\ A \rightarrow B : \{A, \{M\}e_B\}e_B \\ &2.\ B \rightarrow A : \{M\}e_A \end{aligned}$$

Your task is to demonstrate that this "improvement" is disastrous; now the adversary C is again in a position to use a message he picks up and find M. More precisely:

Question 1a: Assume that C picks up message 2 in a protocol run, i.e. the message $\{M\}e_A$ from B to A. *Explain how C can recover M by suitably using the protocol himself.*

Question 1b: Assume now that C instead picks up message 1. *Again, your task is to explain how the adversary can recover M.*

Task 2

This task is independent from Task 1. We consider now a protocol, intended to achieve mutual authentication of two parties. We use the same notation as in the previous task and again assume that keys are authentic. We use n_A to denote a random nonce chosen by A, etc. The protocol is as follows:

$$\begin{aligned} &1.\ A \rightarrow B : \{A, n_A\}e_B \\ &1.\ B \rightarrow A : \{n_A, n_B\}e_A \\ &1.\ A \rightarrow B : \{n_B\}e_B \end{aligned}$$

On receipt of message 2, A concludes that B must be on line at the other end (since the receiver could decrypt the message and the nonce was just recently chosen by A himself). B draws similar conclusions after receiving the third message. We also note that A and B can now use n_A and n_B, which they both know after the run, as input to a suitable hash function in order to agree on a session key.

However, the protocol is broken.

Question 2: *Explain how an adversary C can attack this protocol and make B believe that he is actually A.*

Note! The attack is not very complicated, though perhaps a bit more so than in Task 1. Yet this protocol was published and discussed for many years before the attack was discovered. Thus it is possible that you will need a hint to find the flaw. However, to give you a chance to try to find the attack on your own, we give the hint encrypted using a ROT13 ($A \rightarrow N$, $B \rightarrow O$ etc) encryption. (ROT13 encryption = ROT13 decryption, online translators can be easily found.)

Hint. Gur nggnpx vf n sbez bs nqirefnel-va-gur-zvqqyr nggnpx naq nccyvrf bayl va gur sbyybjvat fvghngvba:

N vavgvngrf n frffvba bs gur cebgbpby jvgu nqirefnel P, jub, va nqqvgvba gb uvf frperg nqirefnevny angher, vf n yrtvgvzngr hfre va gur flfgrz. N'f nvz, gura, vf gb rfgnoyvfu zhghny nhguragvpngvba bs gur pbaarpgvba orgjrra uvzfrys naq P.

Ubjrire, P abj gnxrf gur bccbeghavgl naq vavgvngrf n cnenyyry eha bs gur cebgbpby jvgu O, jurer ur cergraqf gb or N. Va gur eha jvgu O, P arrqf gb qb pregnva bcrengvbaf gung ur pnaabg cresbez uvzfrys; ur znxrf fher gb trg nffvfgnapr jvgu gurfr qhevat uvf cebgbpby eha jvgu N. Gur raq erfhyg vf gung N naq P unir npuvrirq zhghny nhguragvpngvba, naq gung O naq P unir n pbaarpgvba jurer O oryvrirf gung ur vf gnyxvat gb N.

Bonus Question 3: *As a voluntary extra question, you may propose a modification of the protocol to prevent your attack.*