Assignment 1: Anonymous Communication

Privacy Enhancing Technologies (201500042)

Issue date: 1 May 2017; **Due date: 14 May 2017, 23:59 CET.**

Total number of achievable points: 23 points.

You will be working in groups of size 2 on this assignment. Please find your own partner. Hand in your solution via Blackboard. Please submit a PDF document including your and your partner's name and student number (or student numbers if you are enrolled in multiple universities). Additionally, submit your source code in plain text.

1 Introduction

In this assignment, you will study both *mix networks* (abbreviated as *mixnets*) and *dining cryptographers networks* (abbreviated as *dc-nets*). There are two questions on mixnets and one on dc-nets.

1.1 Submission Guidelines

For this assignment you need to work in pairs. You have to hand in a document (PDF) with your answers and your source code (plain text) on Blackboard. Please mention your own name and (UT) student number as well as the name and (UT) student number of your partner. Do not submit your solution using multiple accounts. You may also submit an archive (ZIP, GZip, etc.) containing both the report and the source code, but please do not submit a RAR file.

2 Mixnets

In this question, you are required to create an implementation of a mixnet client. The goal of this question is to give you a hands-on experience with mixnets and give you some insights in what privacy guarantees mixnets can achieve.

Mix networks, *mixnets* for short, are routing protocols that have as goal to make it hard to trace the communication between different parties. The goal of a mixnet is to provide unlinkability and resistance to traffic analysis. This is achieved through the use of *mixes*, central nodes in the network, routing the messages. Make sure you study the lecture slides on this topic before you proceed. Additionally, the original paper by Chaum [Cha81] provides a more detailed description of mixnets.

2.1 Assignment Environment

You have to use your own, personal mixnet environment at pets.ewi.utwente.nl. Please send an email to petcourse-scs@utwente.nl with your own name and student number and your partner's name and student number if you have trouble with creating an account.

For this assignment, we use two different mixnets.

Mixnet 1 This mixnet consists of three t-message flush mixes, all using the same threshold t. There are multiple participants actively using this mixnet.
You will use this mixnet in question 1.

Mixnet 2 This mixnet consists of three message flush mixes, all using different thresholds, t_1, t_2, t_3. You will be the only active participant in this mixnet. However, other participants might have used the mixnet before.
You will use this mixnet in question 2.

To answer some of the questions you will need to do some network analysis on the mixnets. You are given access to all messages entering the mixnet and leaving the network, but you are not allowed to alter the messages, nor delay or drop any message.

A schematic overview of our setup of a mixnet can be found in figure 1. Note that for the mixnet 1, we have threshold $t_1 = t_2 = t_3$.

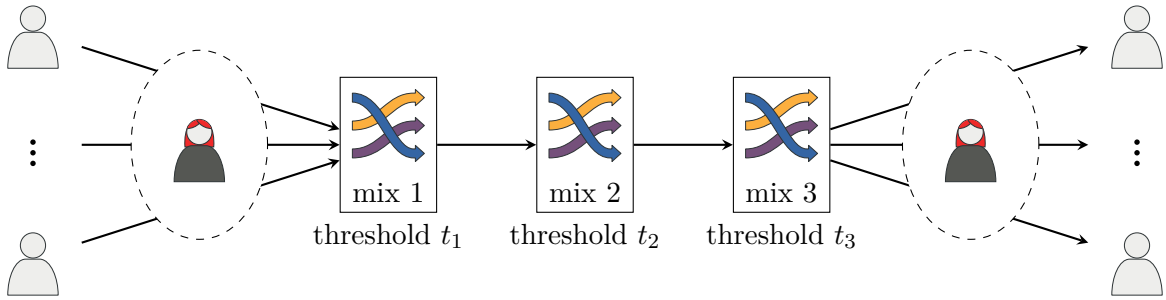

Figure 1: Mix network with an eavesdropper both before the entry mix 1 and after the exit mix 3.

2.2 Protocol

2.2.1 Connecting to the Mixnet

As part of the assignment, you can only connect to the entry mix of the mixnet, mix 1. It is not possible for you to directly connect to the other two mixes or the other participants.

There is a fixed order in the mixnet: every message sent first to mix 1, will at some point reach mix 2 and mix 3, and eventually reach the final recipient (*i.e.*, a participant in the network). It is not possible for you to change the order of the routing.

Type	Key Length	Mode	Padding
RSA	2048 bits	PKCS1-OAEP	
AES	128 bits	CBC	PKCS#7

Table 1: Overview of used encryption in our mixnet protocol.

RSA encryption	AES encryption
$\mathrm{RSA}_{\mathrm{pk}}(\mathrm{IV} \parallel \mathrm{key})$	$\mathrm{AES}_{\mathrm{key,IV}}(\mathrm{message} \parallel \mathrm{padding})$

Figure 2: Format of a single encrypted protocol message.

2.2.2 Network Message Format

Each message sent over the network is prepended by a four byte length field, indicating the number of bytes that are coming afterwards. The length field is formatted as a big-endian unsigned integer. It should be obvious that if the length field does not match the actual content length, this will result in unexpected behavior.

network message	
length (4 bytes)	payload: onion encryption of message

In mixnet 1, debugging is enabled: On successful delivery the server replies with the byte `0x06`, if something went wrong (which may be a delayed error), the server replies with the byte `0x15`. When debugging is disabled, the server replies in most cases with the byte `0x06`.

2.2.3 Message Contents

A vital part of using mixnets involves encrypting the messages sent over the network. To send a message to a mix, we will be using *hybrid encryption*. This means that the plaintext will be encrypted using a symmetric cipher using a randomly selected key. The random key (and Initialization Vector (IV)) in its turn, will be encrypted using the public key of the recipient. In our protocol, we use AES with a 128 bit key in Cipher Block Chaining (CBC) mode and RSA with a 2048 bit key using Optimal Asymmetric Encryption Padding (PKCS1-OAEP). The plaintext messages have to be padded to the AES block length using the PKCS#7 standard. A summary of the used technologies and modes can be found in table 1.

To send a message to a participant, you need to instruct the exit mix where to send your message to. This is done by prefixing your message by the recipients name and separate the name and the message using a comma (,) as delimiter, *e.g.*, `Alice,Hi Alice!`.

In figure 2 the format of a single encrypted message in shown. For the onion encryption it is needed to first encrypt your message to the third mix, encrypt the resulting ciphertext to the second mix, and encrypt the result to the first mix. See also the layered structure below.

Initial

recipient,message

First encryption step

For random key_3, and IV_3, set
$E_1 = \text{RSA}_{\text{pk}_3}(\text{IV}_3 \,\|\, \text{key}_3) \,\|\, \text{AES}_{\text{key}_1,\text{IV}_1}(\text{recipient,message} \,\|\, \text{padding})$.

Second encryption step

For random key_2, and IV_2, set
$E_2 = \text{RSA}_{\text{pk}_2}(\text{IV}_2 \,\|\, \text{key}_2) \,\|\, \text{AES}_{\text{key}_2,\text{IV}_2}(E_1 \,\|\, \text{padding})$.

Third encryption step

For random key_1, and IV_1, set
$E_3 = \text{RSA}_{\text{pk}_1}(\text{IV}_1 \,\|\, \text{key}_1) \,\|\, \text{AES}_{\text{key}_1,\text{IV}_1}(E_2 \,\|\, \text{padding})$.
Now, send E_3 to mix 1.

The public keys of all the mixes are available for download in PEM format in your online environment.

If you decide to implement your solutions in Python, have a look at the *cryptography* package[1]. This package includes predefined functions for AES and RSA-PKCS1-OAEP encryption.

3 dc-nets

dc-nets (or *dining cryptographers networks*) were first described by [Cha88], using the following anecdote: Three cryptographers are having dinner and either one of them or the NSA pays for their dinner. While the cryptographers value each other's privacy, they *do* want to figure out whether one of them or the NSA paid the bill. The cryptographers decide to use a protocol that allows them to communicate anonymously. Using the protocol, the cryptographers only learn if one of them paid or the NSA did. If a cryptographer paid the bill, the protocol does not reveal *which* cryptographer had to reach for his wallet.

The dining cryptographers protocol can be generalized to communicate single-bit and even multi-bit messages if the protocol is repeated.

Make sure you have studied the lecture slides on this topic before you proceed. The paper by Chaum [Cha88] additionally gives a more detailed description of dc-nets.

[1] See https://cryptography.io/.

References

[Cha81] David L. Chaum. "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms." In: *Communications of the ACM* 24.2 (Feb. 1981), pp. 84–90. ISSN: 0001-0782. DOI: 10.1145/358549.358563.

[Cha88] David Chaum. "The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability." In: *Journal of Cryptology* 1.1 (1988), pp. 65–75. ISSN: 1432-1378. DOI: 10.1007/BF00206326.

4 Questions

1. **Interact with the Mixnet** *(8 points)*
   Be sure to use **mixnet 1** for this assignment.

   (a) *(5 points)* Create an application (in Python, Go, or Java) that connects to this network and (correctly) sends a message according to the protocol. Send a message to mixnet participant `PETs` containing your group number. Make sure your messages end up in the log files.

   (b) The next questions are about the design choices in the protocol and your implementation.

      i. *(1 point)* In the described mixnet protocol, you need to encrypt the messages using AES-CBC, while no message authentication code (MAC) is required. Why is this a bad idea if no anonymity is required?

      ii. *(1 point)* In your implementation you probably chose to pick a random IV for each encryption. If you would choose a constant IV, say `0x0000`, in your implementation, would this mean that the eavesdropper could link messages to the same sender/receiver? Explain why or why not.

   (c) *(1 point)* Find out what the (shared) threshold is for the mixes in this network. Explain clearly how you determined the threshold. If you determined this programmatically, make sure you include the source code in your answer.

2. **Unused Mixnet** *(9 points)*
   Be sure to use **mixnet 2** for this assignment.

   (a) *(6 points)* Find out what the threshold is for each of the mixes in this network (t_1, t_2, and t_3). Explain clearly how you determined the thresholds. If you determined this programmatically, make sure you include the source code in your answer.

   No points will be awarded if you found the thresholds by pure luck or when you brute-forced all possible thresholds. You need to provide a logical reason on how you determined the thresholds.

   (b) *(3 points)* Suppose mix 2 in figure 1 would be a timed mix. How would you now determine after how many seconds mix 2 flushes and the thresholds of mix 1 and 3? Argue or prove that your method will always yield the correct thresholds.

   You may assume that there the messages are processed instantly by the mixes and no delay incurred by the network. Furthermore, you may assume that mix 2 flushes within the upper bound of at most $s_{2,\max}$ seconds and that the thresholds t_1 and t_3 are relatively prime. Moreover, you may assume that all mixes are initially empty and you are the only one using the mixnet.

3. **Dining Cryptographers Networks—Colluding Participants** *(6 points)*
   Consider the dc-network shown in figure 3 where not every pair of participants select a bit, but only a subset of all possible pairs. Note that this is different from a traditional dc-net with more than three participants.

   In this network, participants A and E are colluding and share their secret bits that they have in common with another participant, *i.e.*, the bits x_1, x_4, x_5, x_6, and x_7.

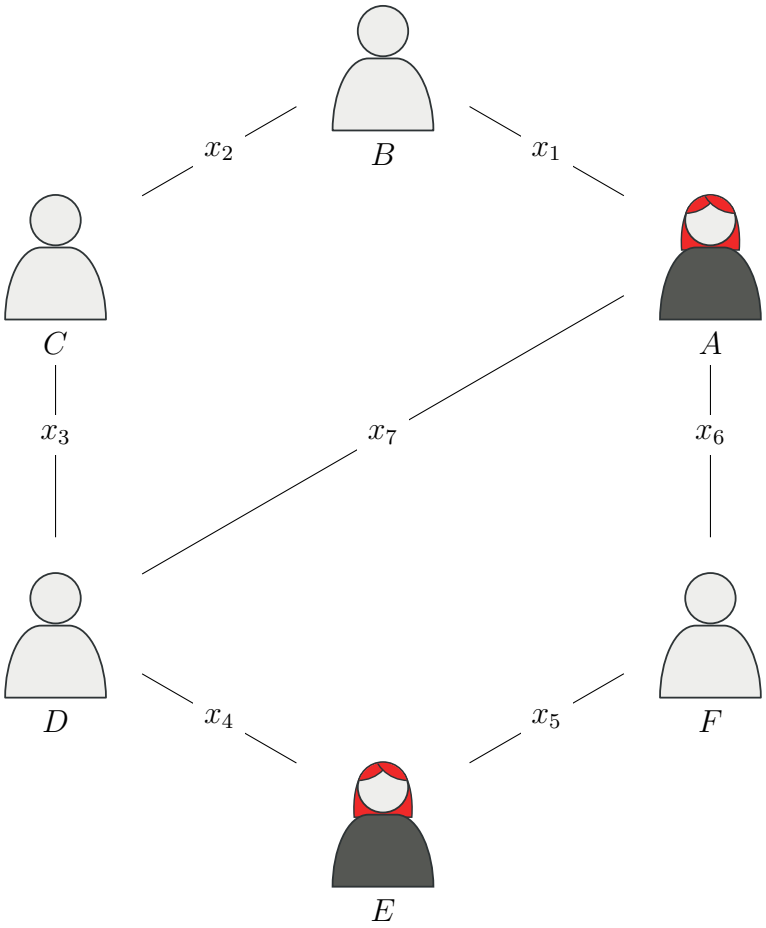

Figure 3: The dc-net for question 3.

(a) *(1 point)* Consider the announcement $(1, 1, 0, 1, 0, 1)$, *i.e.*, participant A announces 1, B announces 1, and C announces 0, *etc.*. What is the bit that is communicated? Remember to motivate your answer.

(b) Assume the network is used by a single participant to communicate the bit 0.

   i. *(1 point)* Can the colluding participants A and E discover if participant F was the sender? Remember to motivate your answer.

   ii. *(1 point)* Can the colluding participants A and E discover if participant D was the sender? If so, how? If not, why?

(c) Assume the network is used by a single participant to communicate the bit 1.

   i. *(1 point)* Can the colluding participants A and E discover if participant F was the sender? Remember to motivate your answer.

   ii. *(2 points)* Can the colluding participants A and E discover if participant D was the sender? *Hint:* Consider all possible values for the secret bits.