Assignment 1: Mixnets

Privacy Enhancing Technologies (201500042)

(Total number of achievable points: 18)

Issue date: 28 April 2016; **Due date: 11 May 2016, 23:59 CET** *(hand in via BB)*

1 Introduction

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 can submit an archive (ZIP, GZip, etc.) containing both the report and the source code, but please do not submit a RAR file.

In this assignment, you will study how mix networks (abbreviated as mixnets) are used, what their weaknesses are and how these can be mitigated. You will do this by making an implementation of a mixnet client. Additionally, you will perform an attack on our mix network and analyze the mixnet's behavior.

The goal of this assignment is to give you a hands-on experience with mixnets and give you some insights in what privacy guarantees mixnets can give.

2 Mixnets

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

In the last part of the assignment, you will investigate an attack on mixnets. Read the paper of Serjantov, Dingledine, and Syverson [SDS03] for a detailed explanation on $n-1$ attacks.

If you are familiar with Tor [DMS04], you might notice the similarities between mixnets and Tor. Tor is based on the same concepts mixnet relies on. However, there are some fundamental differences between these two. In this assignments, you will only look at mixnets.

3 Assignment Environment

You have to use your own, personal mixnet environment at `pets.ewi.utwente.nl`. Please send an email to t.r.vandekamp@utwente.nl with your own name and student number and your partner's name and student number if you did not receive an email with your own port numbers yet.

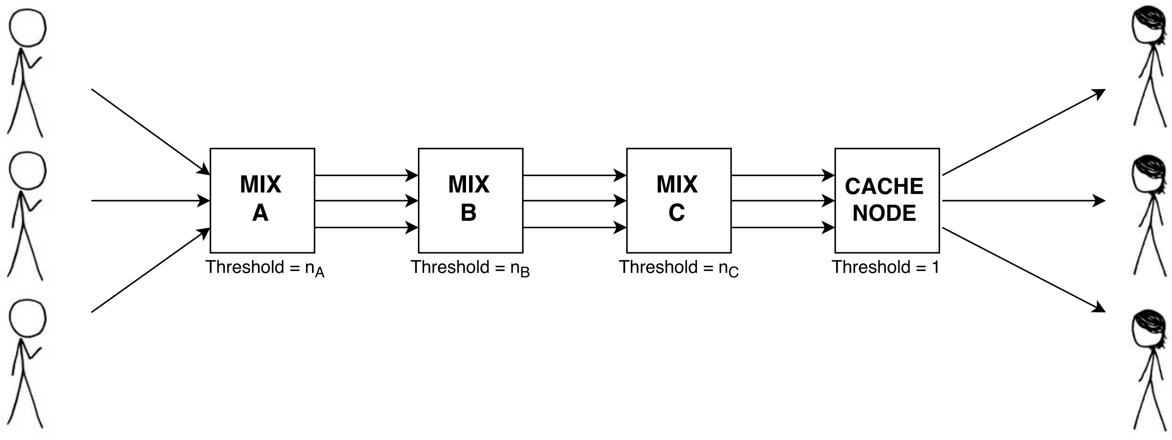

Figure 1: Visualization of a mix network.

3.1 Available Mixnets

For this assignment, we use three different mixnets.

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

Mixnet 2 This mixnet consists of three n-message flush mixes, all using a different threshold. You will be the only active participant in this mixnet.
You will use this mixnet for assignment 2.

Mixnet 3 This mixnet consists of three n-message flush mixes, all with a different threshold. Other participants are using this network, and in assignment 3 you will try to break their anonymity.

In some of the assignments 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. To model this situation, we extend the protocol by requiring all participant to send their messages though a special node in the network, called the *cache node*. This cache node will directly forward the received messages to the final recipient and additionally log that it has done so. The first 8 bytes the cache node receives will be considered as the final recipient, the rest of the data as the actual message. Even if there is no valid recipient, the cache node will still store the message in its log.

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

3.2 Protocol

3.2.1 Connecting to the Mixnet

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

Type	Key Length	Mode	Padding
RSA	1024 bits	PKCS1-OAEP	-
AES	128 bits	CBC	PKCS#7

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

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

Figure 2: Format of a single encrypted protocol message.

Also, there is a fixed order in the mixnet: every message sent to mix A will at some point reach mix B, will then at some point reach mix C, will then at some point reach the cache node and thus eventually reach the participant the message was intended for. It is not possible for you to change this order.

3.2.2 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.

If you decide to implement your solutions in Python, you may use the code example below.

```python
import struct

# code to format the message...
# code to set up a socket...

socket.send(struct.pack('!I', len(message)))
socket.send(message)
```

3.2.3 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 1024 bit key using Optimal Asymmetric Encryption Padding (OAEP) (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. Make sure you understand how these modes and padding will influence the way messages have to be sent.

In figure 2 the format of a single encrypted message in shown. Note, however, that you will send your message to the receiver through three mixes and the cache node. Thus, you will first need to encrypt your message for the cache node, then, for the third mix, the second, and finally the first mix. In other words, the message you will eventually send contains four layers of encryption. See also the layered structure below.

Initial

message

First encryption step

For random $\text{key}_{\text{cache}}$, and IV_{cache}, set
$E_1 = \text{RSA}_{\text{pk}_{\text{cache}}}(\text{key}_{\text{cache}} \,|\, \text{IV}_{\text{cache}}) \,|\, \text{AES}_{\text{key}_{\text{cache}},\text{IV}_{\text{cache}}}(\text{message} \,|\, \text{padding})$.

Second encryption step

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

Third encryption step

For random key_{B}, and IV_{B}, set
$E_3 = \text{RSA}_{\text{pk}_{\text{B}}}(\text{key}_{\text{B}} \,|\, \text{IV}_{\text{B}}) \,|\, \text{AES}_{\text{key}_{\text{B}},\text{IV}_{\text{B}}}(E_2 \,|\, \text{padding})$.

Forth encryption step

For random key_{A}, and IV_{A}, set
$E_4 = \text{RSA}_{\text{pk}_{\text{A}}}(\text{key}_{\text{A}} \,|\, \text{IV}_{\text{A}}) \,|\, \text{AES}_{\text{key}_{\text{A}},\text{IV}_{\text{A}}}(E_3 \,|\, \text{padding})$.
Now, send E_4 to mix A.

The public keys of all the mixes and the cache node are available Privacy Enhanced Mail (PEM) format via the webserver on your environment.

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

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.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul Syverson. “Tor: The Second-Generation Onion Router.” In: 13th USENIX Security Symposium ’04. (San Diego, CA, USA, Aug. 9–13, 2004). 2004, pp. 303–320. URL: http://static.usenix.org/legacy/events/sec04/tech/dingledine.html.

[SDS03] Andrei Serjantov, Roger Dingledine, and Paul Syverson. “From a Trickle to a Flood: Active Attacks on Several Mix Types.” In: Information Hiding: 5th International Workshop, IH 2002. (Noordwijkerhout, The Netherlands, Oct. 7–9, 2002). Ed. by Fabien A. P. Petitcolas. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 36–52. ISBN: 978-3-540-36415-3. DOI: 10.1007/3-540-36415-3_3.

[1] See https://pypi.python.org/pypi/pycrypto.

4 Assignments

This project consists of three assignments. Make sure you read the description carefully before you start. If you used a program to solve an assignment, you are required to hand in the source code as part of your solution.

1. For this assignment you'll use mixnet 1.
   (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. Make sure your messages end up in the log file.
   (b) (0 points) Send a message to mixnet participant `TIM` containing your student number.
   (c) (1 point) Find out what the (shared) threshold is for the mixnodes 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. For this assignment you'll use mixnet 2.
   (a) (6 points) Find out what the threshold is for each of the mixnodes in this network (n_A, n_B, and n_C). Explain clearly how you determined the thresholds. If you determined this programmatically, make sure you include the source code in your answer.
   (b) (1 point) Is it always possible to determine the thresholds of each mixnode in a mixnet? If so, explain why. If not, provide a counterexample.
3. For this assignment you'll use mixnet 3.
   (a) (3 points) You will now perform an $n-1$ attack. In this mixnet, each participant is communicating exclusively with one other participant. Find out who is talking to participant `TIM`.
   (b) (2 points) How could this type of attack be prevented in this specific mixnet? Describe at least two preventions.