COMP311-1: Logic Circuit Design

Fall 2019, Prof. Taigon Song

Project 3. Due: Dec. 19, 9:59am [Total: 210 + 40 points]

[Your Student ID]

Your Name

No.	Checksheet item	Done?	Points
1.	ascii (part 1)	[bit stream in decimal]	10
2.	tgBASE (part 1)	[bit stream in decimal]	10
3.	oneBig (part 1)	[bit stream in decimal]	10
4.	encrypt (part 1)	[bit stream in decimal]	10
5.	DoneBig (part 2)	[bit stream in decimal]	10
6.	DtgBase (part 2)	[bit stream in decimal]	10
7.	Dascii (part 2)	[bit stream in decimal]	10
8.	decrypted message (part 2)		10
9.	[Modelsim] ascii (part 1)	[Y/N]	5
10.	[Modelsim] tgBase (part 1)	[Y/N]	5
11.	[Modelsim] oneBig (part 1)	[Y/N]	5
12.	[Modelsim] encrypt (part 1)	[Y/N]	5
13.	[Modelsim] DoneBig (part 2)	[Y/N]	5
14.	[Modelsim] DtgBase (part 2)	[Y/N]	5
15.	[Modelsim] Dascii (part 2)	[Y/N]	5
16.	[Modelsim] output file (part 2)	[Y/N]	5
17.	p, q (part 3)	[value in decimal]	10
18.	d (part 3)	[value in decimal]	10
19.	decrypted message (part 3)		10
		[write down the message]	
21.	[Modelsim] p, q (part 3)	[Y/N]	20
22.	[Modelsim] d (part 3)	[Y/N]	20
23.	[Modelsim] message print	[Y/N]	20
24	[Bonus]		40

1. Goal:

Students will design an encryption/decryption system of RSA in this project. Given an input stream, you should design a system that encrypts/decrypts the message in RSA format. In short, (1) you should confirm what the message is, (2) encrypt the message in to the given RSA format, and (3) decrypt it back to the original message stream in given scenarios. In this project, you will need to encrypt/decrypt three different messages. The following will be the details to perform your desired action.

2. Specifications:

Design a system considering the following specifications.

- Your input stream is 1022 bits (binary).
- A scenario of the input stream is in the format of ASCII.
- Assume your number system is 'unsigned decimal'.
- Input stream is translated into a 6-bit (tgBASE64) format to reduce the data size.
- Two tgBASE64 characters will be merged into one packet for RSA encryption.
- An RSA packet cannot be bigger than 10k (decimal).
- The encrypted message will traverse the opposite to be translated back to the original bit stream.
- You should extract an output stream that decodes the RSA message.

3. System design – part 1 and 2

3.1. Splitting the input stream

Your top-level block diagram for Project 3 should be very similar to Figure 1. Adjust your system based on your needs. Part 1 has a file of an input stream of a certain ASCII code. For your convenience, this project asks your system to first split the input bit stream into 7 bit streams. When splitting the number, earlier bits should be delivered prior to the latter bits. For example, if your system should send some decimal numbers of 1324354679, then you should send your number in the following order:
(prior) 1 – 3 – 2 – 4 – 3 – 5 – 4 – 6 – 7 – 9 (latter)

3.2. Converting ASCII in to tgBASE64

ASCII is a text format where a binary bit stream is converted into certain text characters. Below is a link to a reference you can follow regarding the details of ASCII format.

https://en.wikipedia.org/wiki/ASCII

ASCII is a 7-bit format supporting 128 characters. However, this project utilizes only 64 characters, so it decides to convert ASCII into a 6-bit custom data format: tgBASE64. Inside the Project 3 package, you will see a 'ASCII-tgBASE64_conversion_table.png' file. Use the table as a reference to convert 7-bit ASCII characters into tgBASE64 format.

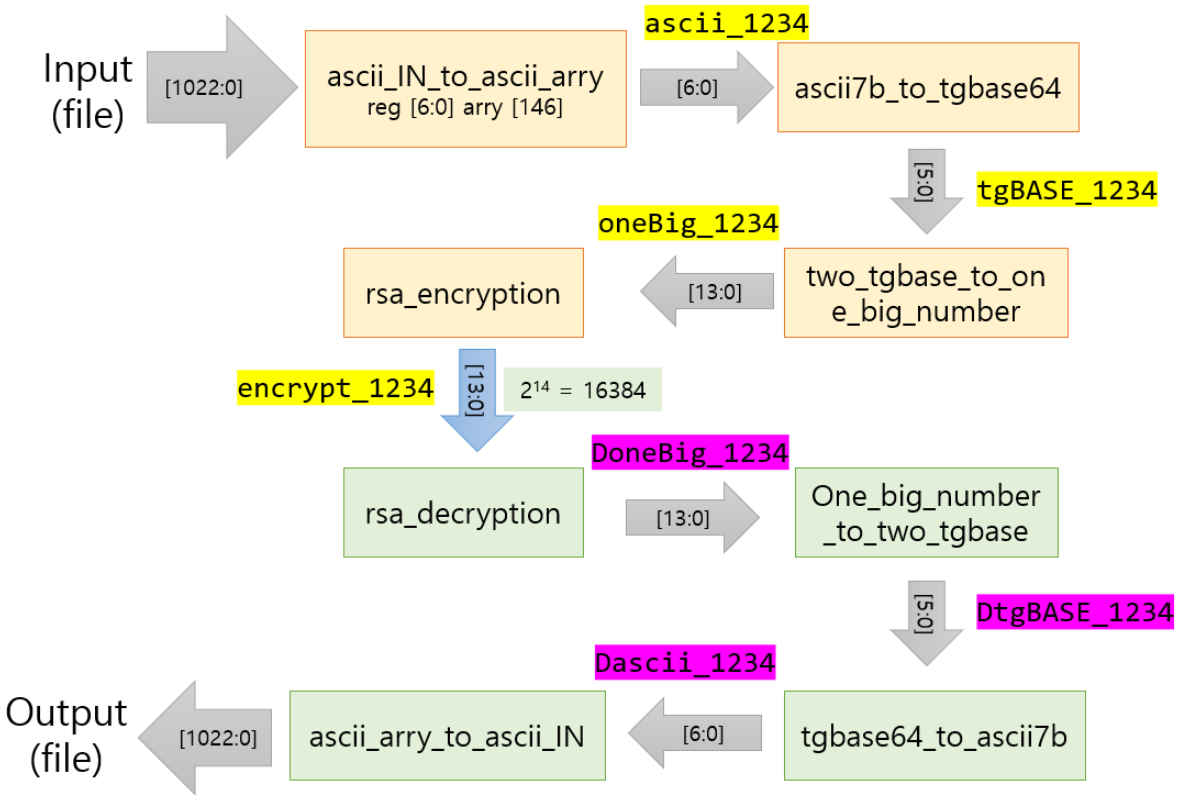

Figure 1 Top-level system diagram of Project 3
(_1234 should be the last 4 digits of your student ID).

3.3. Merging two tgBASE64 numbers into one number.

To better enhance security in the rsa system we design, this project decides to send a 14-bit binary number that is to be transmitted. Therefore, we use two tgBASE64 numbers to become one 'Big Number'. For example, if number A = 32, B = 51, Big Number = 3251. Keep in mind that your system should place the number that has been received earlier to be placed in the higher digit. In other words, if the numbers came in the orders A-B, the Big Number should be 3251, if B came earlier than A, Big Number should be 5132.

3.4. RSA encryption

The core of this project asks you to understand the RSA encryption mechanism. Below are links to obtain some background of the RSA encryption.

- https://ko.wikipedia.org/wiki/RSA_%EC%95%94%ED%98%B8

- https://en.wikipedia.org/wiki/RSA_(cryptosystem)

In the technological perspective, below are some links that would be helpful for you to understand the project.

- https://kevin0960.tistory.com/entry/RSA-%EC%95%94%ED%98%B8%EC%99%80-%EA%B7%B8-%ED%95%B4%EB%8F%85 (Korean)
- https://www.comparitech.com/blog/information-security/rsa-encryption/ (English)

For your convenience, you may study any source that you think that would benefit your understanding of the RSA encryption.

In part 1 and 2, the following are the essential key values.

p = 101, q = 103, n = 10403, e= 71, d = 431

By comprehending the methodology behind RSA, you should know what variables you should use. In addition, you should understand why the step of 3.3. is necessary for "efficient" RSA encryption encoding in the given public key.

The encoded number passes through a 14-bit channel as similar as 3.3. to 3.4. Regarding the RSA encoding itself, below is an example of how the encoding process is done:

If (n,e) = (2537, 13), original message = 1234 → encrypted message = 2391

If (n,e) = (2537, 13), original message = 1111 → encrypted message = 0724

3.5 RSA decoding.

The first part of this project provides you the decryption key. Understand the RSA mechanism and use the decryption key to decode the encrypted bitstream to its original value. Go through the exact opposite process so that you can extract your original bit stream back. Your final goal is to export an output file that contains your final bitstream, which should be exactly the same as your original bitstream. Part 1 gives you an exercise for encoding, and Part 2 gives you an exercise for decoding.

4. System design – part 3.

4.1. What is included in part 3.

In this section, we assume that you are a hacker that intercepted an open key (n , e) = (10573, 89) and an RSA encoded message. Your task is to (1) find the private key, and (2) export the output stream that is in binary.

4.2. Finding the private key – p, q.

Find the private key in whatever method you prefer in Verilog. As long as you can find the private key using Verilog and Modelsim, you will receive full credit. This section is an open section that provides full freedom to the students that wish to figure out the private key. To verify that you have completed this step in Verilog, you should screen capture a text or a waveform in Modelsim.

4.3. Finding the private key – d.

Similar as 4.2., your task is to find the private key in the Verilog perspective. Design a module that exports d. You will need to justify that your source code extracts the correct d by reporting the waveform or the screen text in Modelsim.

4.4. Deciphering the full message (for part 2 and part 3).

In this sub-section, you must decipher the full message and print it out in Modelsim. In order to perform this task, design whatever system necessary and recycle as many modules as possible to reduce your work load. Regarding 4.4., you have full freedom to design whatever you want.
Hint: if you want to print a number to text, the syntax you should use would be '%c'.

5. Grading policy.

In this project, all Verilog-related results should be printed out in Modelsim text. Only 4.2. and 4.3. results may be visualized in waveform if you desire. A sample of Modelsim text is shown below.

```
VSIM 27> run
# encrypt_1234:  1942
# encrypt_1234:  4949
# encrypt_1234:  5200
# encrypt_1234:  3452
# encrypt_1234:  5549
# encrypt_1234:  4101
```

6. Presentation

Once you submit your report, you will receive an email from Prof. Song of whether you are selected for presentation. If you are selected, please prepare a demonstration of your source code and a 5-page ppt presentation.

[Submission]

- Submit your report in ENGLISH.
- Both Online and Offline report.

- Use the first page of Proj 3 for submission. Fill in the boxes based on your completion ratio.
- Submit your report in .pdf and submit your source codes compressed in a .zip file.
- Synthesis report of two modules are necessary (no other synthesis reports are necessary):
  (1) RSA encoder, and 'd' finder. Please attach the synthesis report in your .zip file for these two modules (synth_1_synth_synthesis_report_0).
- Early submission: Students will get extra points when submitted before the following dates
  - 40 points: by 12/9 8:59am
  - 20 points: by 12/16 8:59am
  - 0 points: by 12/19 9:59am