Course Project Descriptions

1 A List of Projects

There are a total of 4 projects. You may choose one of them.

Project 1. Correlation attack on a simplified A5 pseudorandom sequence generator

A5/1 is a stream cipher used in GSM. In this project, you are asked to launch a correlation attack on a simplified A5, which is given below, and recover the key by knowing certain bits of the key stream. The specification of simplified A5 is given as follows.

1) It employs three LSFRs which generate m-sequences with periods 2^7-1, 2^8-1, and 2^9-1, respectively.

2) Let $f_i(x)$ be the primitive polynomial for $LFSR_i, i = 0, 1, 2$ which generate their respective m-sequences $\mathbf{a} = \{a(t)\}$, $\mathbf{b} = \{b(t)\}$, and $\mathbf{c} = \{c(t)\}$ where

$$\begin{aligned} f_0(x) &= x^7 + x + 1 \\ f_1(x) &= x^8 + x^4 + x^3 + x^2 + 1 \\ f_2(x) &= x^9 + x^4 + 1. \end{aligned}$$

3) The tap positions on the LFSRs are: $d_0 = 3, d_1 = 4$ and $d_2 = 5$, respectively.

4) Majority function $f(x_0, x_1, x_2) = (y_0, y_1, y_2)$ is defined by Table 1. It can be described as $y_i = 1$ if x_i is in majority where $(x_0, x_1, x_2) = (a(t + 3), b(t + 4), c(t + 5))$.

The output sequence is given by $\mathbf{u} = \{u(t)\}$ which is computed as, at time t

$$u(t) = y_0(t) + y_1(t) + y_2(t) + x_0(t) + x_1(t) + x_2(t), t = 0, 1, \cdots$$

where

$$f(a(t+3), b(t+4), c(t+5)) = (y_0(t), y_1(t), y_2(t)), (x_0(t), x_1(t), x_2(t)) = (a(t+3), b(t+4), c(t+5))$$

In other words, this is obtained from A5 by removing the stop-and-go operation, which becomes simply a combinatorial generator.

Your tasks are as follows.

(a) Find the boolean representation for y_i for each $i = 0, 1, 2$, and show that the output is correlated with each input variable.

Table 1: Majority function in A5/1

(x_0, x_1, x_2)	$f(x_0, x_1, x_2)$ $= (y_0, y_1, y_2)$
000	111
111	
001	110
110	
011	011
100	
101	101
010	

(b) The adversary has obtained 520 bit key stream bits, i.e.,

$u(0), u(1), \cdots, u(519) =$

00111101000011010111011010111101010011011001000000110101010100001
01011001101110111110010001100010000011110011110000010100101110101
00111000011111101010100111101000100011101110110100001011100100101
11110001110100111010100000110001001110010010101111010000110001011
00110011110101101100011111011000101101100100111001110010000110101
00101100110110010010100100010001110100110000011000000110101110100
00001111101111011111101010100001101110101110000101100010010111100
01110110110000010000011011001110100010100110101100110100001111010

The key is the 24 bits of the initial states of three LESRs and the key initial phase is run 64 times without output. Then pseudorandom generation phase starts to output, which is denoted as $u(0), u(1), \cdots,$. Find the key using correlation attack.

(c) Assume that the attacker gets 64 bit ciphertext below:

$c_{1014}, c_{1025}, \cdots, c_{1087} =$

0010101100001011110110100010001010010110111001100100100001111111

Find the message $m_{1014}, m_{1025}, \cdots, m_{1087}$ (note that $c_t = m_t + u(t)$, $t = 0, 1, \cdots$).

(d) Decrypt the ciphertext given in (c) by finding a more efficient attack than the correlation attack (e.g., your attack uses less than 512 consecutive bits of the output). Justify your answer.

Project 2. Security on the TLS/SSL handshake protocol and application data protocol

The goal of this project is 1) to simulate the key agreement from TLS/SSL handshake in C++. In class, we learnt there are 3 different ways to do this: DH, DHE, RSA. Only DHE and RSA are required for this project; 2) security analysis on the application data protocol.

Description: The TLS/SSL sits on top of a TCP layer to provide confidentiality, integrity, and authentication. The application layer sits on top of TLS/SSL layer and uses it to connect to other applications, and to send and receive encrypted messages. Confidentiality is done by encryption and message authentication is achieved by MAC. To use encryption/authentication, both the client and server must first agree on a shared key and must done mutual entity authentication before the shared key is accepted. The shared key must not be sent using directly TCP since we don't want Eve to notice and to join the conversation. This shared key must instead be established during the handshaking phase that you are to implement.

Most of the TCP and SSL code are provided to you. You are only required to complete two of the following functions: "connect()" in SslClient and "accept()" in SslServer (see UML below). This part is provided by Meng Yang.

connect(): This function belongs to the SslClient. When this functions is called, a TCP connection should first be established, then handshaking should happen. If the connection is successful, this function should return 0, otherwise -1.

accept(): This function belongs to the SslServer. When this functions is called, the server should wait for a client's TCP connection, then create a new SSL class to return to application. If the connection is successful, this function should return the new SSL connection containing the shared key, otherwise NULL.

Once the TCP connection is established, both SSL client and server must communicate using records (SSL::Record) and the send() and recv() functions for records inside the SSL class. Later when the handshaking phase is done, the communication of the application layer will use send() and recv() functions for strings. The two latter functions will apply encryption/decryption on the message using AES in CBC mode with the shared key.

UML: See Figure 1.

Provided (download from LEARN):

- makesfiles and CryptoPP library
- source code for TCP layer (src/tcp) and part of SSL (src/ssl)
- logger class: contains functions to sniff the traffic. When SslServer and SslClient starts, they will create a new instance of Logger. What TCP sent and receive will be contained inside the log files created by the loggers.
- test cases for SSL (tst/ssl): there are 3 main files inside the folder: main_server runs the server, main_client1 runs a client connecting using DHE, main_client2 runs a client connecting using RSA. (See Figure 1 for the stacks.)

Testing: Please use the provided test files to test your functions. The test cases reside inside the "tst/ssl" folder.

Extra: Please use a nice coding style and ensure there are no memory leaks.

Note. You may also use Python to establish communication at socket layer without using what are provided you here.

Questions to be answered: In your report, you should answer the following questions.

(a) For each case of the following two cases for generating the pre-master key: one is DHE and the other is RSA, explain how entity authentication is conducted. Comment the advantages and disadvantages in these two different key sharing processes.

(b) Explain how IVs are chosen in both CBC MAC and GCM in SSL/TLS application data protocol, attacks related to those settings in a real world application, and their possible countermeasures.

(c) Identify three different attacks on TLS/SSL in an application that a user conducts his banking activities, and show the consequences, and countermeasures. You should include one attack which runs on HTTPS, i.e., HTTP over TLS.

(d) Suppose Alice sends packets to Bob using TCP over IPsec. If the TCP acknowledgement from Bob is lost, then the TCP sender at Alice's side will assume the corresponding data packet was lost, and thus retransmit the packet. Will the retransmitted TCP packet be regarded as a replay packet by IPsec at Bob's side and be discarded? Explain your answer.

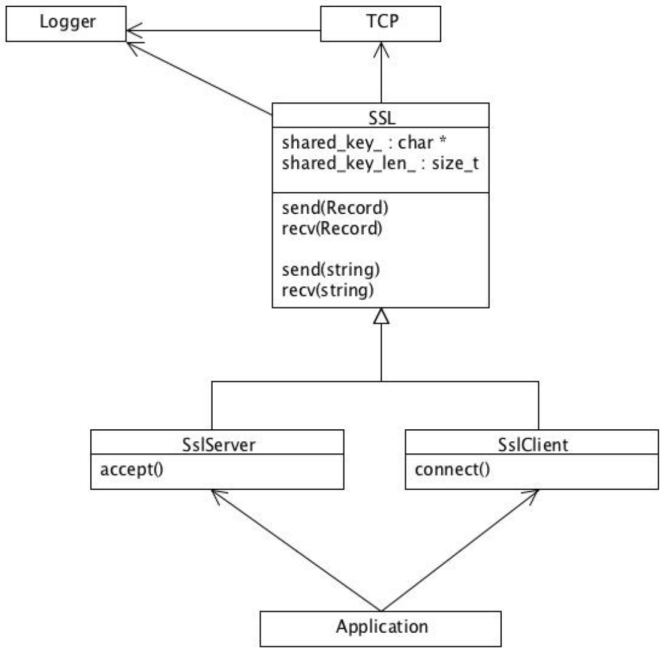

Figure 1: Stacks of provided code for SSL hand shaking

(e) In order to make TLS/SSL handshaking protocol more efficient, one of the approaches is to simplify TLS/SSL by restricting the key establishment to DHE. Provide a simplified handshaking protocol where certificate requests are always sent, and give analysis of pros and cons for this approach. Find an example which exists in practical systems and comment their approach. (In TLS 1.3, the other two key agreement schemes are removed.)

Project 3. Merkle tree based authentication and its applications

In a Peer-to-Peer (P2P) system, each node stores same data cross the network. A Merkle tree can be used for a user to detect whether a retrieved data from a particular node has been modified or corrupted. This can be done by the Merkle tree authentication. The scheme, denoted as Γ consists of three algorithms: $\Gamma(Gen, Retrieve, Verify)$, which are explained below.

We assume that there are n nodes which store an identical data d, and attackers are passive.

Gen Gen is to generate a binary Merkle tree. Let h be a collision resistant hash function and $m = \lceil \log n \rceil$. The notation $x||y$ means the concatenation of x and y.

1. Each node generates the hash of d with its identifier ID_i and a random number s_i. Specifically, for node i: it computes $a_i = h(x_i), x_i = ID_i||d||s_i$, and it broadcasts a_i to the entire network.

2. Now node i receives all $a_i : 0 \leq i < n$. It will form a tree, where the n leaves are ordered from left to right, and hash value $a_i = h(x_i)$ is attached to vertex leaf i.

3. Each parent vertex is attached with a hash value where an input to h is the concatenation of the hash values from two children vertexes. In other words, let a parent vertex have index (i, j), denote an attached value as $a_{i,j}$, and the two children of $a_{i,j}$ as v_{left} and v_{right}. Then
$$a_{i,j} = h(v_{left}||v_{right}). \tag{1}$$
Finally, it gets the root at the top, denoted as $Merkle_{root}$.

4. The Gen outputs $Merkle_{root}$, and publishes it in the system.

Retrieve: the retrieving process. If a user requests to retrieve data d from node i, then node i will provide the data d together with the authentication path from the leaf i to the root: $\sigma = (a_t = b_t, b_{t+1}, \cdots, b_{m-1})$ where t is the sibling of node i, i.e., $t = i - 1$ or $i + 1$.

Verify: Verification process. The user computes the root from σ, denoted as r' and $Verify(\sigma) = 1$ if and only if $r' = Merkle_{root}$. In this case, the user is sure that data d is not modified. Otherwise $Verify(\sigma) = 0$, the user will reject d and search d from some other node.

This project is to implement the above scheme for $n = 32$. Cryptographic primitives and the parameters are given below.

A hash function

The hash function you will use in the project is SHA3-224, for which c code and python code are provided. You can use any available code or library. Here is an online hash calculator that you can play around `https://www.pelock.com/products/hash-calculator`. An example computation of SHA3-224 is as follow, the input is a hexadecimal string.

```
sha3_224("38363138363536383336")=
0000000089cdf55ab2f8463c7438d555f8fd2c91c87ede3235c9bd18
```

Encoding convention needed for this project is given below.

Bit string concatenation:

1. $x_i = ID_i||d||s_i$. In this project where $ID_i = i + 900, i = 0, \cdots, n-1$, data d is a text file more than 1000 bits (you may choose it by yourself), and s_i is a 224-bit random number, generated by a PRNG, specified later.
2. Input to SHA3-224: a bit string whose length is a multiple of 8.
3. Convert the output of SHA3-224 to an integer if needed:

$$\begin{aligned} h(x) &= b_0, b_1, \cdots, b_{222}, b_{223} (\text{a bit string}) \\ &\leftrightarrow b_0 2^{223} + b_1 2^{222} + \cdots + b_{222} 2 + b_{223} \text{ (an integer)} \end{aligned}$$

Pseudo-random number generator (PRNG)

PRNG is implemented by SHA3-224 with an input

$$0xff0101010101010101010101010101*$$

where $*$ is 16 bytes of any truly random bits, such as the strength of your touching your key board or the clock time when you first attempt the project, or power level when you switch on your devices. The output SHA3-224 is a 224-bit random number, namely, s_i for node i. You need to generate $n = 32$ random numbers $s_i : 0 \le i < 32$.

Questions to be answered: In your report, you should answer the following questions.

(a) Implement $\Gamma(Gen, Retrieve, Verify)$ for $n = 32$. In other words, generate the Merkle tree with 32 nodes and list all hash values for leaves, parent nodes, and the root. Show Retrieve for querying node 7 and show how Vefify works for this query. Provide security analysis for this system and discuss some possible attacks. Determine the security level of PRNG defined above with justification. Note that SHA3-224 has 112 bit security.

(b) Design a system to achieve this functionality using DSA with 112 bit security. Compare its efficiency and security with the Merkle tree based scheme.

(c) Explore at least two additional applications for the Merkle tree based scheme, and discuss their pros and cons.

(d) If the system requests that the nodes in P2P system should not be able to read the contents of d, how the scheme should be modified to achieve this goal. In this case, we assume that P2P nodes are honest-but-curious, i.e., they faithfully execute the scheme, but attempts to read the content of data.

2 What to do next

You may work along or team with one person. Once you have decided on this, please email Radi Abubaker at rabubaker@uwaterloo.ca (even if you are working alone) and he provide you with a group number. Any project is not allowed more than ten groups (individual or two-person group) to work on that. How to sign up your project? First come and first serve.

3 Due Dates

(1) The presentation will be held in the class on **April 6**. Each group (individual or two-person group) has 5-minutes to present your results.

(2) Both slides and report is due at

11:59pm ET, March 30

You need to submit 1) Project Report, 2) Presentation Slides, 3) Code. You may upload all three into the dropbox on LEARN.

4 Grading

(1) (5 marks) You will do a 5-minute presentation including question period (5-7 slides will be enough).

(2) (25 marks) You must submit a report about 10 pages without counting your code. Your code is also needed to upload to LEARN. We will run your code to check your results. The 25 marks will be assigned based on how well you answer the questions, how good of your implementation or solutions in terms of complexity for solving that, and how much extension you made beyond the questions that I have asked.

If there are any questions or something is confusing, please ask me. I reserve the right to provide updates and clarifications as needed.