COMM048 Information and Network Security 2023–2024

Coursework (100%)

Professor Liqun Chen and Dr Jack Tian
Surrey Centre for Cyber Security
University of Surrey

Information

Deadline

Coursework solutions must be uploaded to SurreyLearn by

Wednesday, 24th April 2024 (16:00 h)

The date for the return of feedback is

Wednesday, 15 May 2024

Layout

A cover page, which is the first page of the coursework sheet with your name and URN, and Solutions, including a python script as part of an exercise, must be submitted in a single PDF document to the Coursework Submission Folder of the module. You can submit multiple times and only the last submission will be valid.

Miscellaneous

Exercises must be solved individually, without consultations with other students. Students are advised to read the exercise questions and supplementary materials carefully. This table must be filled by the student before submission:

Student's Name	
URN	—

The following table will be filled during the marking process. Each exercise brings up to the number of points listed in the table that can be obtained by solving the corresponding tasks. Coursework results will be returned within three weeks of the submission deadline and feedback will be provided in the form of solutions to compare with, which will be published on SurreyLearn.

Exercises	1	2	3	4	5	6	Total
Maximum Points	15	25	20	10	15	15	100
Received Points							

Software and Tools

You may want to use the following software and tools to complete this coursework.

Linux (Ubuntu)

It will be easier to use Ubuntu for most of the exercises. This operating system is installed on the lab machines. If you prefer, you can use your own computer.

To get the manual page of a program, e.g. openssl, type

```
..$ man openssl
```

into a terminal window.

OpenNebula

OpenNebula is a cloud-based system that is used to provide and manage virtual machines on demand. Please follow the OpenNebula introduction document (which you should have used for COMM048 Labs) to set up and use a virtual machine in OpenNebula. To do so, start by logging onto a lab workstation using your University account name and password. Then run Firefox and navigate to

https://opennebula.eps.surrey.ac.uk.

Then login to OpenNebula using your University account name and password again.

Wireshark

Wireshark (`https://www.wireshark.org/`) is a well-known GUI tool to interactively dump and analyze network traffic (introduced in Lecture 1 and Lab 1). It lets you interactively browse packet data from a live network or a previously saved capture file. Wireshark uses the common libpcap file format, which is also the format used by tcpdump and various other tools. In order to start Wireshark type

```
..$ sudo wireshark
```

in a terminal window. To identify network packets use Wireshark frame numbers. Frames reassemble network packets and display them in their context.

Scapy

Scapy (`https://scapy.readthedocs.io/en/latest/`) is a powerful interactive packet manipulation program (introduced in Lecture 1 and Lab 1). It can forge or decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies, and much more. It is written in python (https://www.python.org/) and also uses python syntax in its interactive interface. Run scapy using the following command:

```
..$ sudo scapy
```

There are two useful commands to remember, which are: `ls()` and `ls(protocol)`. The first command lists all of the available protocols, and the second command lists all of the details relevant to a specific protocol. For instance, if you type `ls(IP)`, you will see all of the details that can be edited when creating an IP packet.

Snort

Snort (https://www.snort.org) is a free and open-source network intrusion prevention system and network intrusion detection system (introduced in Lecture 5 and Lab 5). For a complete overview, refer to the snort user manual in the Lab 5 directory.

Contents

1 Exercise 1: SYN Flooding Attack by Using SCAPY (15 Points)

1.1 A SYN packet in Scapy (7 Points)

Make use of two virtual machines one of which plays the role of the target server and the other one the role of the attacker. Use scapy to craft a TCP SYN packet and let the attacker send it to the target server. Describe any component of your scapy packet and the command used to send it.

Hints:

1. First explain your environment.
2. Make sure that you are sending the TCP SYN packet to an open port.
   (a) If you use virtual machines on your computer supporting "nmap" function, type

   ```
   ..$ nmap -open <IP address of victim>
   ```

   in a terminal window on the attacker machine. You can see how many open ports are available and choose one to use.
   (b) If you use the COMM048 virtual machines via OpenNebula created for COMM048 Labs, type

   ```
   ..$ nc <IP address of victim> 22
   ```

   in a terminal window on the attacker machine. The port 22 is used for ssh. You can use this one as your target.
3. You can check the effect of your packet on the victim by typing

   ```
   ..$ netstat -nt
   ```

   in a terminal window on the victim machine. This will give you a table showing the status of any connections to the machine (see below).
4. Source ports for your tests should be in the range set aside for public use (49512 —65535).

Capture your packet with Wireshark and provide a snapshot showing the packet details together with the `netstat` output from the victim.

1.2 A python script (8 Points)

Write a python script that repeatedly sends TCP SYN packets to the victim. Run your program and show the result by using

```
..$ netstat -nt
```

on the victim machine. Include part of this output in your write-up, 15 lines, or so, should be enough to show that your program is working. Make sure that the src IP addresses should be different from each other in multiple sendings. For example

```
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0  192.168.1.164:22        192.168.1.124:62279     SYN_RECV
tcp        0      0  192.168.1.164:22        192.168.1.64:57815      SYN_RECV
tcp        0      0  192.168.1.164:22        192.168.1.27:52417      SYN_RECV
tcp        0      0  192.168.1.164:22        192.168.1.169:57660     SYN_RECV
tcp        0      0  192.168.1.164:22        192.168.1.141:58526     SYN_RECV
tcp        0      0  192.168.1.164:22        192.168.1.172:55191     SYN_RECV
```

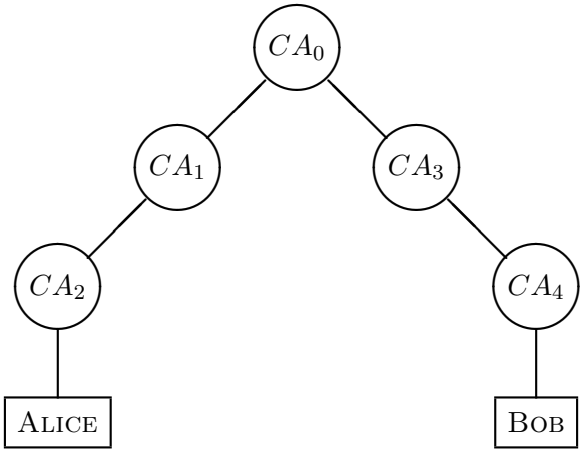


Figure 1: A certificate hierarchy

```
tcp      0      0  192.168.1.164:22        192.168.1.50:53440      SYN_RECV
tcp      0      0  192.168.1.164:22        192.168.1.218:53170     SYN_RECV
tcp      0      0  192.168.1.164:22        192.168.1.194:62334     SYN_RECV
tcp      0      0  192.168.1.164:22        192.168.1.241:54757     SYN_RECV
tcp      0      0  192.168.1.164:22        192.168.1.196:50560     SYN_RECV
tcp      0      0  192.168.1.164:22        192.168.1.4:63491       SYN_RECV
tcp      0      0  192.168.1.164:22        192.168.1.64:53050      SYN_RECV
tcp      0      0  192.168.1.164:22        192.168.1.11:60532      SYN_RECV
tcp      0      0  192.168.1.164:22        192.168.1.24:64138      SYN_RECV
```

In your python script, please write down how you stop the program when it is running.

2 Exercise 2: Digital Certification and Email Signing (25 Points)

2.1 The concept of digital certificates (3 points)

What is a digital certificate? Who does generate a digital certificate? Which cryptographic algorithms are usually used to create a digital certificate?

2.2 An example of certificate hierarchy (5 points)

Consider a certification system with multiple certificate authorities, including a root CA written as CA_0 and a set of intermediate CAs denoted by CA_1, CA_2, ..., and a set of clients, such as Alice, Bob, Charlie, ... The CAs corresponding to Alice and Bob are arranged in a hierarchy as shown in Figure 1.

Please demonstrate the various certificates necessary to navigate the hierarchy. For example, Bob receives a digital certificate from Alice, describe the process by which Bob validates Alice's public key.

(Hint: please re-draw this figure and add necessary information to your figure to indicate who certifies whose public key.)

2.3 Email signing (17 Points)

In this exercise, you should use OpenSSL (see Lab 2) to generate a signing certificate for **your university email address** or **your personal email address**. You should send a signed email to yourself and show that the certificate is accepted and seen as valid and include a screen capture in your write-up (for example, see Figure 2). The certificate has to be valid for **365 days** and contain a **2048-bit** RSA key.

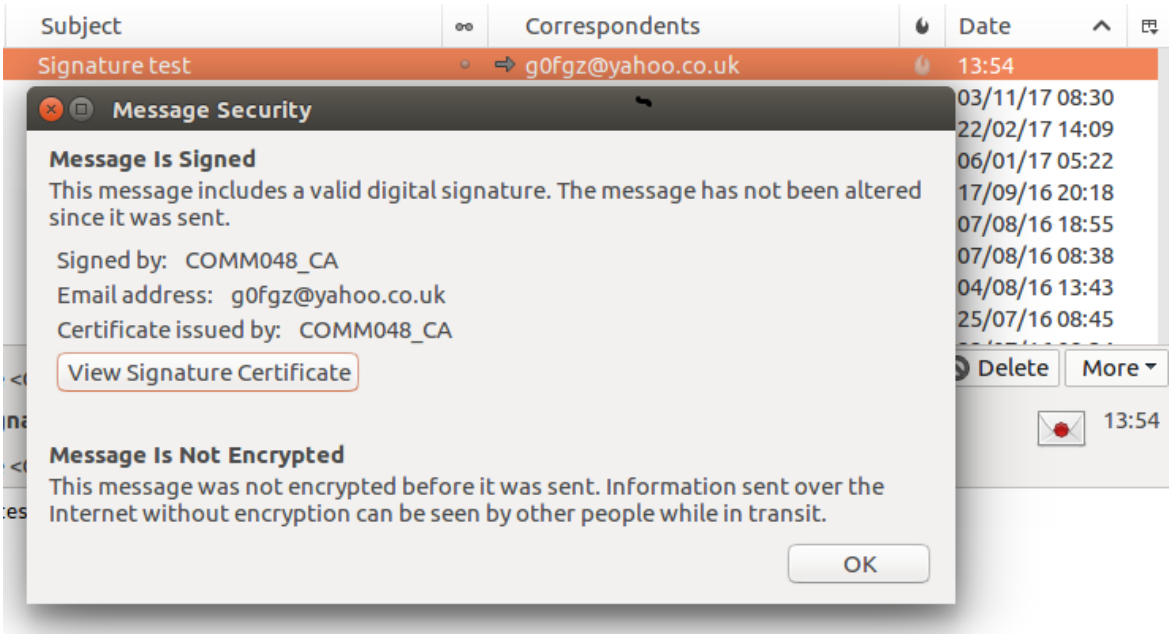


Figure 2: Confirming the signature's validity

You can use any email client on your computer or you can use Thunderbird on Ubuntu in a lab computer. However, please be aware that students are now required to use two-factor authentication and this is incompatible with using Thunderbird for the university email. If you do not find an email client that works with your Surrey email to do this exercise, you can use Thunderbird with a personal email account, for example using Gmail.

Hints

1. You will first need to become a root Certificate Authority as you did in Lab 2 and then create your email signing certificate.
2. To conform with RFC5280 the email address should not be included in the Distinguished Name (leave this blank), but included in the certificate as a `subjectAltName`. To do this set the `subjectAltName` in the `[ user cert ]` extensions section of the OpenSSL configuration file.
3. You will need to import your CA certificate together with your email signing certificate into your mail server.

Your write-up for this exercise should contain at least the following information:

- Please explain your email environment. (2 point)

- A complete list of OpenSSL commands (including a description of each of the arguments) that were used to generate the private key and email certificate. You do not need to include the commands that you use to become a Certificate Authority. (5 points)

- The certificate description output using OpenSSL (cf. Listing 1 for an example). (3 Points)

- A description of the steps taken to import the private key and certificate into your email client and confirm its validity. Please include a screenshot for each step. (5 points)

- Your screen capture shows that your signed digital signature is accepted. (2 Points)

Listing 1: Example of OpenSSL print of Certificate:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 2607447712 (0x9b6a7ea0)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=GB, ST=Surrey, O=COMM048 Ltd, CN=COMM048_CA
        Validity
            Not Before: Nov  4 14:02:51 2017 GMT
            Not After : Nov  4 14:02:51 2018 GMT
        Subject: C=GB, ST=Surrey, O=COMM048 Ltd, CN=COMM048_CA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:c7:f6:50:b3:bf:cc:ea:44:1a:47:85:e6:d7:1c:
                    49:a3:4b:1a:69:24:07:60:8b:45:17:46:7c:70:d6:
                    51:0a:71:6d:14:a8:dd:53:8a:5c:c5:40:1d:08:e0:
                    22:c9:a4:e0:c6:9a:8c:42:7b:a7:13:9b:6f:16:14:
                    f6:fb:fd:15:cf:d6:9e:bc:f3:44:88:64:c5:cd:7a:
                    0a:a0:21:e3:50:fa:f4:96:1c:02:fc:17:6e:25:0d:
                    a8:9a:87:e5:f5:d9:22:81:27:f3:c5:da:a4:56:31:
                    c9:3e:9a:df:fd:7a:e2:eb:9d:c8:1a:8b:5a:30:a0:
                    4a:f1:30:89:bb:13:63:43:52:bf:fc:7a:20:68:47:
                    ff:91:c1:e9:12:a0:e2:a0:cb:db:b8:0b:b6:70:88:
                    3d:63:16:2d:a0:ff:86:80:2b:ab:de:ac:72:a1:e9:
                    9c:0d:4d:1e:2f:85:4c:f1:f5:68:60:64:83:03:b7:
                    00:aa:15:95:e8:fd:5f:0a:7f:03:3f:c5:4a:4f:5f:
                    9f:f6:bd:d8:3a:02:8d:e3:62:a9:c6:b9:b1:04:61:
                    a3:20:04:d4:74:c2:a4:66:7b:29:d7:bd:5e:49:fe:
                    1d:8a:fc:6b:e4:c2:92:3c:0a:7a:12:bb:b4:8a:6d:
                    55:55:ca:f2:00:31:b3:cc:bc:5c:23:ef:3c:26:6e:
                    5a:53
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
```

```
            OpenSSL Generated Certificate
        X509v3 Subject Key Identifier:
            B4:E0:FC:64:77:5A:3E:1A:88:5A:28:78:B0:5F:CC:A5:DC:B5:CD:E2
        X509v3 Authority Key Identifier:
            keyid:DA:31:CF:2B:1B:3A:C4:09:1C:47:34:87:74:B6:92:5B:48:D8:60:47

        X509v3 Subject Alternative Name:
            email:G0FGZ@yahoo.co.uk
Signature Algorithm: sha256WithRSAEncryption
     a7:6e:3a:d9:3b:ac:6a:80:d0:f6:b9:70:b0:ec:66:3c:38:83:
     dc:d1:01:83:3b:52:b3:d2:e9:7f:31:79:3b:25:b4:a5:42:3d:
     d4:33:93:bf:8a:77:d2:ce:65:d9:49:f6:83:d2:bc:0a:bb:c3:
     94:a8:ff:9e:23:04:ab:a1:60:a1:9d:21:9d:5f:e1:2c:c9:67:
     cb:bd:ac:2d:ea:89:b0:6c:4a:50:e4:80:95:4a:9e:2b:b1:49:
     40:e4:d3:ee:2f:61:d4:b3:5a:72:e7:cf:ad:33:8f:14:c7:83:
     02:09:38:0c:0c:34:22:c2:5b:9c:a5:4d:0a:b2:1d:c6:d9:c6:
     f3:9d:28:75:b9:a4:c4:3d:ed:87:e6:12:3e:2b:2b:7e:c1:d5:
     c1:17:1b:00:ee:15:21:1e:33:ec:fa:ed:49:a6:6f:d1:ec:0a:
     39:ec:48:b0:95:7f:b2:75:11:4f:87:60:59:ef:8d:d3:34:65:
     6d:5d:1c:33:0d:4e:d7:94:a3:25:6d:8c:3d:69:39:5e:2d:92:
     c0:62:b6:60:59:67:2c:05:19:bd:41:56:ff:4f:ab:82:a7:f2:
     15:49:64:dc:2d:fe:f0:47:89:9a:ff:13:cf:4b:fe:38:98:60:
     96:5a:31:ab:ae:95:14:66:ee:29:b4:72:b1:3c:69:04:b4:e6:
     96:90:40:51
```

3 Exercise 3: SSH and Kerberos (20 points)

3.1 SSH (6 points)

What is SSH? Explain how you can use SSH to remotely access a computer in the Surrey IFH lab. (Note that this is an alternative to using GlobalProtect)

3.2 Kerberos in SSH (6 points)

NIST (https://nvlpubs.nist.gov/nistpubs/ir/2015/NIST.IR.7966.pdf) specifies several methods for using SSH. One of them uses Kerberos. Explain how this method works. A Kerberos protocol can make use of public keys or symmetric keys. You are free to choose either of these two types of keys. You are expected to provide a sufficient details in your explanation. Please draw the message flow in the Kerberos authentication protocol and write the main information exchanged in each flow.

3.3 Kerberos in a cross-realm case (8 points)

If a client and a server are not in the same realm that includes a single Kerberos KDC. Explain a cross-realm authentication solution using a Kerberos protocol. You are expected to provide detailed information in your solution to show how each message is protected using a cryptographic algorithm. You can choose any type of cryptographic keys.

4 Exercise 4: IEEE 802.11 (10 points)

4.1 WiFi (4 points)

Discuss your experience of using WiFi, including observations on authentication, access control, data protection and system reliability.

4.2 WEP (6 points)

Why does the Wired Equivalent Privacy (WEP) mechanism specified in IEEE 802.11 fail? What are the major changes to WEP in the later IEEE 802.11i standard?

5 Exercise 5: VPN, IDS and Working with Snort (15 Points)

5.1 Virtual Private Network (VPN) (3 points)

What is a VPN and what is it used for?

Discuss benefits of VPNs compared to dedicated networks utilizing frame relay, leased lines, and traditional dial-up.

5.2 Intrusion Detection System (IDS) (3 points)

Briefly explain the following items.

1. Outline the components of an Intrusion Detection System (IDS).
2. Describe network-based and host-based IDSs and their differences.

5.3 Writing Snort rules (6 points)

Let Home_Net be set to one of the virtual machines which is used by yourself in COMM048 labs and External_Net set to !Home_Net.

1. Create a snort rule that alerts for SSH connection from any IP address different from Home_Net.
2. Create a snort rule that alerts for "worm" in content outgoing from your Home_Net.
3. Create a snort rule that alerts for pings from External_Net.

5.4 A computer worm (3 points)

Suppose you need to detect a computer worm that aims at causing a denial of service on some Internet hosts. Figure 3 shows the tcpdump output that resulted by running the worm script against a machine on a test network.

Write a Snort rule to detect this worm. The rule must include the following:

1. The head of the snort signature must alert on UDP packets to port 1434 on the "home network" from the "external network".
2. The message placed in the alert must specify "Internet Worm to be stopped".

3. The rule must search for an '07' byte at the start of the payload (coloured magenta in the figure). Note that in the figure the mustard coloured section is the IP header (20 bytes), followed by the UDP header (8 bytes) and then the start of the payload.

4. If the '07' match succeeds the rule must then search for the binary string "71 f2 03 01 04 9b 71 f2 01". This is the cyan coloured section in the figure.

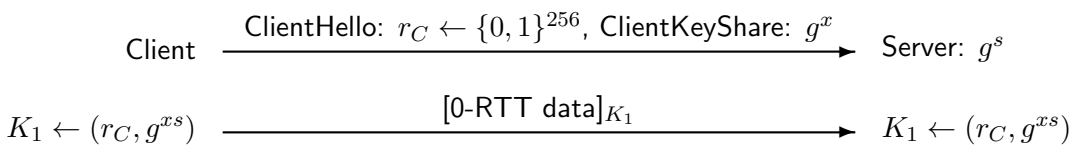
5. Finally, if the rule matches the previous checks, the rule must search for the text "sock" (coloured orange in the figure).

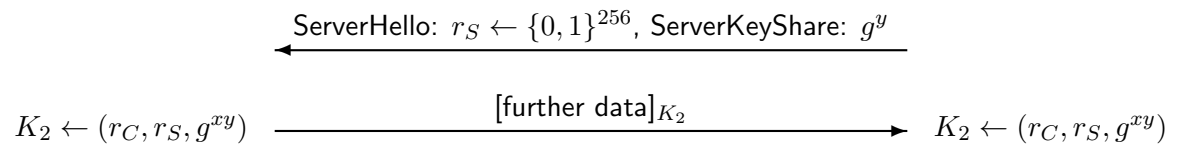
```
0x0000   4500 0194 56ea 0000 8011 5e24 c0a8 0164   E...V.....^$...d
0x0010   c0a8 0196 07b5 059a 0180 a94b 0701 0101   ...........K....
0x0020   0101 0101 0101 0101 0101 0101 0101 0101   ................
0x0030   0101 0101 0101 0101 0101 0101 0101 0101   ................
0x0040   0101 0101 0101 0101 0101 0101 0101 0101   ................
0x0050   0101 0101 0101 0101 0101 0101 0101 0101   ................
0x0060   0101 0101 0101 0101 0101 0101 0101 0101   ................
0x0070   0101 0101 0101 0101 0101 0101 01dc c9b0   ................
0x0080   42eb 0e01 0101 0101 0101 70ae 4201 70ae   B.........p.B.p.
0x0090   4290 9090 9090 9090 9068 dcc9 b042 b801   B........h...B..
0x00a0   0101 0131 c9b1 1850 e2fd 3501 0101 0550   ...1...P..5....P
0x00b0   89e5 5168 2e64 6c6c 6865 6c33 3268 6b65   ..Qh.dllhel32hke
0x00c0   726e 5168 6f75 6e74 6869 636b 4368 4765   rnQhounthickChGe
0x00d0   7454 66b9 6c6c 5168 3332 2e64 6877 7332   tTf.llQh32.dhws2
0x00e0   5f66 b965 7451 6873 6f63 6b66 b974 6f51   _f.etQhsockf.toQ
0x00f0   6873 656e 64be 1810 ae42 8d45 d450 ff16   hsend....B.E.P..
0x0100   508d 45e0 508d 45f0 50ff 1650 be10 10ae   P.E.P.E.P..P....
0x0110   428b 1e8b 033d 558b ec51 7405 be1c 10ae   B....=U..Qt.....
0x0120   42ff 16ff d031 c951 5150 71f2 0301 049b   B....1.QQP......
0x0130   71f2 0101 0101 518d 45cc 508b 45c0 50ff   ......Q.E.P.E.P.
0x0140   166a 116a 026a 02ff d050 8d45 c450 8b45   .j.j.j...P.E.P.E
0x0150   c050 ff16 89c6 09db 81f3 3c61 d9ff 8b45   .P........<a...E
0x0160   b48d 0c40 8d14 88c1 e204 01c2 c1e2 0829   ...@...........)
0x0170   c28d 0490 01d8 8945 b46a 108d 45b0 5031   .......E.j..E.P1
0x0180   c951 6681 f178 0151 8d45 0350 8b45 ac50   .Qf..x.Q.E.P.E.P
0x0190   ffd6 ebca                                 ....

tcpdump: listening on eth0 21:45:15.104113 192.168.1.100.1973
         > 192.168.1.150.1434: udp 376
```

Figure 3: tcpdump of the Internet Worm

6 Exercise 6: TLS 1.3 Handshake Protocol (15 Points)





In this Diffie-Hellman (DH) handshake protocol with 0-RTT, the server's public key g^s is used for a certain time period, e.g., a couple of days, where the value g is a generator of an algebraic group, and s is the server's secret key. Key K_1, created based on the client's nonce r_C and the DH output g^{xs}, is used immediately to protect the 0-RTT data, and key K_2 is used for protecting the subsequent communications.

6.1 Analysis of 0-RTT Key Exchange (7 points)

1. What is 0-RTT and what is it used for?
2. *Forward secrecy (FS).* Which key cannot guarantee FS? Please justify your answer. Hint: The attackers might compromise the server and learn the value s, but they are NOT allowed to learn the value x or y.
3. *Replay attacks.* How does a man-in-the-middle attacker replay the client's messages in this 0-RTT protocol?

6.2 PSK-based DH handshake protocol (8 points)

An alternative approach in the TLS 1.3 is that instead of using g^s, the client and server pre-share a symmetric key PSK, that is established from their previous communications. In the 0-RTT protocol, they use PSK to derive the 0-RTT key K_1.

1. *Key derivation.* In this case, how to compute K_1?
2. *Replay attacks.* Whether this PSK-based approach can provide replay protection guarantees? If yes, please describe how to use PSK to prevent replay attacks. Otherwise, please explain your argument.