CS3211 Course Projects (two options):

P1: Formal Verification of TLS 1.3 (Key Exchange) Protocol

This course project requires 4-6 students in a group. This project requires some network/security background of TLS 1.3 protocol. Apart from modelling the concurrency system with CSP# in PAT, the students are required to study and investigate TLS 1.3 protocol and, public and private key, etc. We encourage the students with more cybersecurity background or strong enthusiasm for the topic to go for this project.

Background

TLS (Transport Layer Security) protocol is designed for the privacy of communicated messages on the Internet, through cryptographic techniques. Once the protocol is designed, researchers never stop investigating its adversarial approaches [1, 2], leading to its evolution. TLS 1.3 is the latest version (hopefully) addressing the most known vulnerabilities. TLS protocol is designed to build an encrypted communication channel between the client and the server on the Internet. Overall, the client and server shall use asymmetric encryption technique (e.g., RSA and DH key exchange protocol) to build a *shared secret*. With the shared secret, both peers use a symmetric encryption technique (e.g., AES) to build the encrypted communication channel.
In this project, the students shall model a key exchange protocol called Diffie-Hellman Ephemeral key exchange protocol [3].

DHE key exchange

Fig. 1 shows a simplified version of DHE key exchange.

Step 1. The client starts the TLS connection with a ClientHello message, indicating possible options of cipher suites. In this course project, we simplify that there is only one way of key exchange as DHE. Moreover, DHE has two parameters, a primitive *p* and a base *g,* where *g* is a primitive root modulo of *p*. We will introduce how this mathematical relation will ensure the success of key exchange.

Step 2. Then, the server generates a random key KS and use it to derive a message M_s with the formula showed in Fig.1. Then, it sends a Server Hello message back to the client, telling the client that (1) it agrees to use DHE as a key exchange method, and (2) it can derive a message M_s from the given parameters. In addition, the server will send its public key for encrypting all the follow-up communication.

Step 3. After the client receives the message, it conducts the same procedure, i.e., generate a random key *KC* and a message M_c. Moreover, it generates a premaster key PKC = M_s^{KC} mod p. Then, it encrypt M_c with the public key PS back to the server.

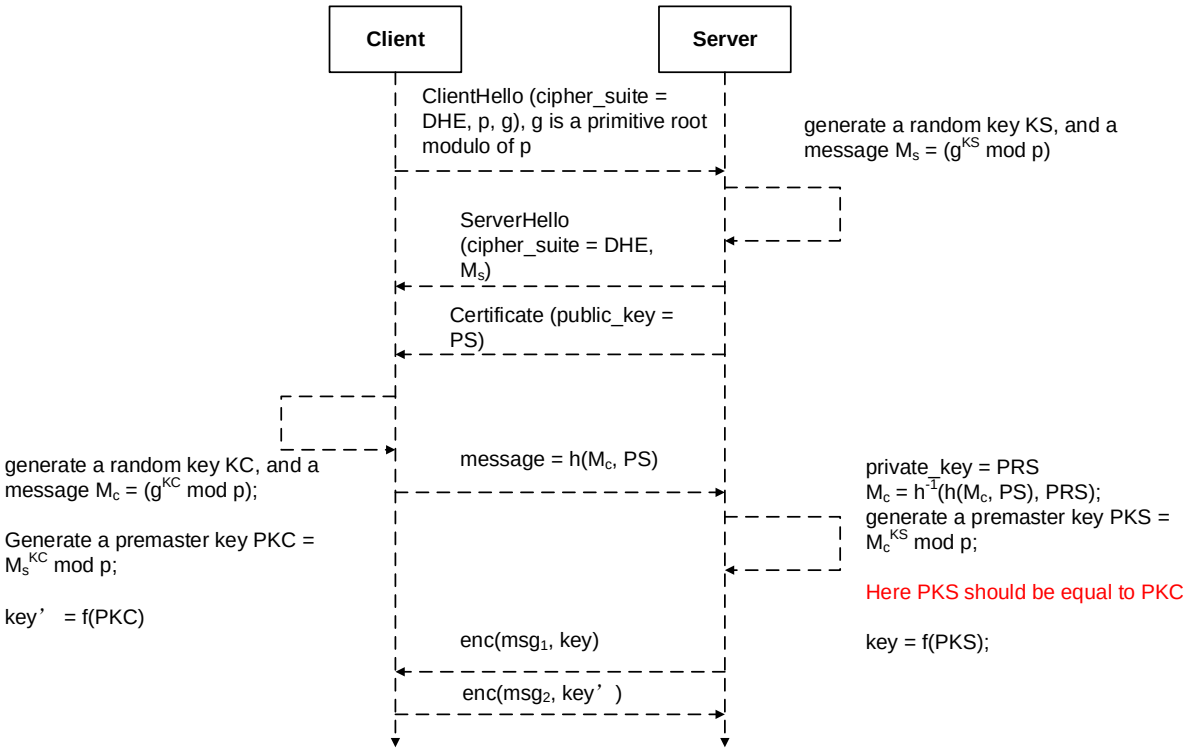

Fig.1. TLS Handshake with DHE Key Exchange

Step 4. After the server receives the encrypted message, it decrypts it into M_c with its private key PRS. Then, it generate a premaster key PKS = M_c^{KS} mod p.

Note here*: given the primitive p and its primitive root modulo g, whatever the value of the random key *KS* and *KC*, the PKS in the server should be of the same value to the PKC in the client. By this means, the client and the server reach a shared secret even when an attacker has observed all the transmitted value, e.g., *p, g, M_s*, etc.

Step 5. Finally, using the same transferring function *f*, the client and the server can have the same session key (i.e., key in the server and key' in the client, while key is equal to key') to apply the symmetric encryption on the communication.

The above procedure is call DH key exchange. However, in order to avoid the attacker to apply replay attack, i.e., record everything during the communication and spend more time on guessing the random key *KS* and *KC*, the client and server will regenerate their random keys after a short period, which makes the key exchange a DHE (Diffie-Hellman Ephemeral).

Verification properties

We are going to model both protocols and verify the following security properties.

1. Client and server should establish the same session key after key exchange.

2. If the client/server believes it has established a session key with an authenticated peer, then the attacker does not know the session key when it is being used.

3. Even if the private key of the server (of the time being) is stolen, the attacker cannot compromise the communication in the next phase, i.e., perfect forward secrecy (PFS).

Adversarial model

We consider an extension of the Dolev-Yao (DY) attacker [4] as our threat model. The DY attacker has complete control of the network, and can intercept, send, replay, and delete any message. To construct a new message, the attacker can combine any information previously learnt, e.g., decrypting messages for which it knows the key, or creating its own encrypted messages. We assume perfect cryptography, which implies that the attacker cannot encrypt, decrypt or sign messages without knowledge of the appropriate keys. We additionally allow the attacker to do the following:
1. compromise the private key of protocol participants,
2. compromise the random keys.

Tasks
- Please construct a CSP concurrent model regarding the protocol description and verify the properties.
- After your verify the given protocol with your PAT program, please consider the following questions:
1. If the attacker would like to compromise the protocol, what other capability does he or she need?
2. Given existing protocol, what is the most damage can the attacker cause?
Please create new assertions and verify them.

Note that, it is a simplified version of the protocol. Interested students can implement and verify a model in a more sophisticated design, which can be referred in [5] for more details.

Reference

1. Sheffer, Y., R. Holz, and P. Saint-Andre, *Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS).* RFC 7457, 2015.
2. Moller, B., T. Duong, and K. Kotowicz, *This POODLE Bites: Exploiting The SSL 3.0 Fallback.* 2014.
3. Diffie, W. and M. Hellman, *New directions in cryptography.* IEEE transactions on Information Theory, 1976. **22**(6): p. 644-654.
4. Dolev, D. and A. Yao, *On the security of public key protocols.* IEEE Transactions on information theory, 1983. **29**(2): p. 198-208.
5. Cremers, C., et al. *A comprehensive symbolic analysis of TLS 1.3*. in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017.

If you are interested in PI, then email your group students details and initial questions to TA: Lin Yun llmhyy@gmail.com during the week of 25 January 2021 (tutorials for this week is through email discussions on the selection of the projects)

P2: Applying Probabilistic Model Checking in Soccer Analytics

Probabilistic and Concurrent Soccer Game Model description:
Traditionally, Probabilistic Model Checking (PMC) was mainly used to analyze the correctness and performance of the computer system and protocols, however, it can be applied in other areas too. For example, during the lecture, we have demonstrated an interesting case where PMC is applied to sports analytics to reason the relations between player strategy and his winning probabilities. That example models a tennis singles tiebreaker game. We ask you to apply the PMC technique to **model a soccer game with multiple parallel processes** and reason the relations between player strategy and the team's winning chance.

To help you, we would like to offer the following tips:

- You can start with a much simplified street soccer game. There are two teams, each consists of two players. You can divide the entire soccer field into 4x3=12 zones. Players can stay, run, dribble, pass or shoot. If he runs or dribbles, he can only move to the adjacent zone (left/right/up/down). Take note dribble, pass and shoot can fail, especially for long passes and shoot from distance. You should reasonably estimate the success rate of each possible action.
- Use **proper abstraction** to model the states, the state transitions and the player's choices and the probabilities distributions.
- **Part of the challenge is to estimate the probabilities** in your model. Please be innovative and use your knowledge in statistics or machine learning.
- **Another challenge is to reduce the state space.** You can use proper abstraction, eg. model the team as whole instead of modeling individual players.
- Once you are able to model a simplified game, you should extend it to be more **realistic** (eg. consider a real game with 11 players each team). Ideally, the winning strategies or probabilities discovered by your model should be supported by real match results.

If you are interested in P2, then email your group students details and initial questions to TA: Jiang Kan (Michael) jiangkan.sg@gmail.com during the week of 25 January 2021 (tutorials for this week is through email discussions on the selection of the projects)

Project requirement:

- Use CSP# (PAT) to model and verify system properties.
- Use CSP# (PAT) concurrent construct to model the concurrent behaviors.
- The main deliverable is a project report, which is about 20 pages (single spaced 11/12pt) and contains the following:
    - Describe the application you have chosen to model.
    - Properties to be verified, e.g. reachability probabilities.
    - Formal model in CSP# language.
    - Result and discussion.
    - Feedback on how to improve the PAT.
- Each team should have one to four persons.
- There is a mid-term project presentation scheduled on **9 Mar**, and a final project presentation on **13 Apr**.