Exercise 1: In the public-key certificate system, suppose that the certificate authority (CA) employs DSS signature. Assume that CA's private key and public key pair is denoted by (sk_{CA}, pk_{CA}). Bob requests a public-key certificate for his key pair (sk_B, pk_B), which is a RSA key pair.

(a) Explain what Bob should submit to the CA to get the certificate for his public key pk_B.

Using the following information, the certificate authority will generate a digital signature to bind Bob's secret key with his public key:

- pk_B
- Bob's identity
- Expiration date
- Scope of key (encryption/signing)

(b) How does CA generate the certificate of Bob's public key? (You only need to specify the format of the certificate).

This procedure is performed by creating a digital signature using the certificate's own secret key. The format is as follows:

$$Certificate_{pk_B} = Signature_{CA}(pk_B, Bob's\ identity, Expiration\ Date, Scope\ (enc/sig))$$

(c) When Alice wishes to send some sensitive information to Bob using Bob's public key, what does she need to do before she performs the RSA encryption using Bob's public key?

Alice must verify Bob's identity first – in order to do so, she must verify the public key in her possession is Bob's, and not an adversary's public key. To verify Bob's identity, she must contact the CA, which will return a certificate generated using their own digital signature to bind Bob's identity and public key, as shown in part b). Alice is then able to decrypt the digital signature using the CA's public key to verify Bob's identity and public key. Once verified, Alice can encrypt her message using Bob's public key before sending it to him.

(d) Why is a certificate authority necessary for a public-key system?

A certificate authority converts a peer-to-peer trust assumption to a multiple-to-one trust assumption in the public key infrastructure. By assuming that the CA's public key is unique, every user must now trust that CA. Without a trusted CA, everyone is able to issue their own keys and this decreases the level of trust with respect to authentication. A trusted CA is the only entity that can issue trusted digital certificates, allowing trusted authentication between peers. [1]

Exercise 4: Security analysis on IKE Auth:

(a) Try to find a man-in-the-middle attack on the "IKE AUTH" exchange with the modification that the data fields over which the authentication apyloads are generated such that $\boldsymbol{AUTH_i = Sig_{ski_i}(N_r)}$ and $\boldsymbol{AUTH_r = Sig_{ski_r}(N_i)}$, assume certificates are exchanged.

The IKE AUTH exchange relies on the Initiator and Responder binding their private key with their ID. This exchange achieves the binding by first generating the confirmation keys SK_{pi} and SK_{pr}, and then binding these keys to the ID of the initiator and responder, respectively. Each respective binding is signed by either the initiator or responder's private key, ensuring that they can validate each other's identity and establish mutual connection.

$$AUTH_i = Sig_{sk_i}(INIT, N_r, PRF(SK_{pi}, ID_i))$$
$$AUTH_r = Sig_{sk_r}(INIT, N_i, PRF(SK_{pr}, ID_r))$$

In the case of this question, the initiator and responder only sign the nonce of the other party as follows:

$$AUTH_i = Sig_{sk_i}(N_r)$$
$$AUTH_r = Sig_{sk_r}(N_i)$$

As a result, the initiator and responder's private key is not bound to their IDs and therefore no mutual authentication is established. Additionally, since the certificates are already exchanged, there is no additional identity verification and authentication at the AUTH stage. The security of IKE becomes solely reliant on $KE_i = g^i$ and $KE_r = g^r$, which are exchanged in the IKE INIT phase to establish the pre-secret master key and generate the various keys in the subsequent exchanges. Therefore, a simple man-in-the-middle attack on any Diffie Hellman key exchange applies in this situation, where the attacker may intercept communications during the key establishment procedure. The following method in Figure 1 describes MITM attacks on any DH key exchange:

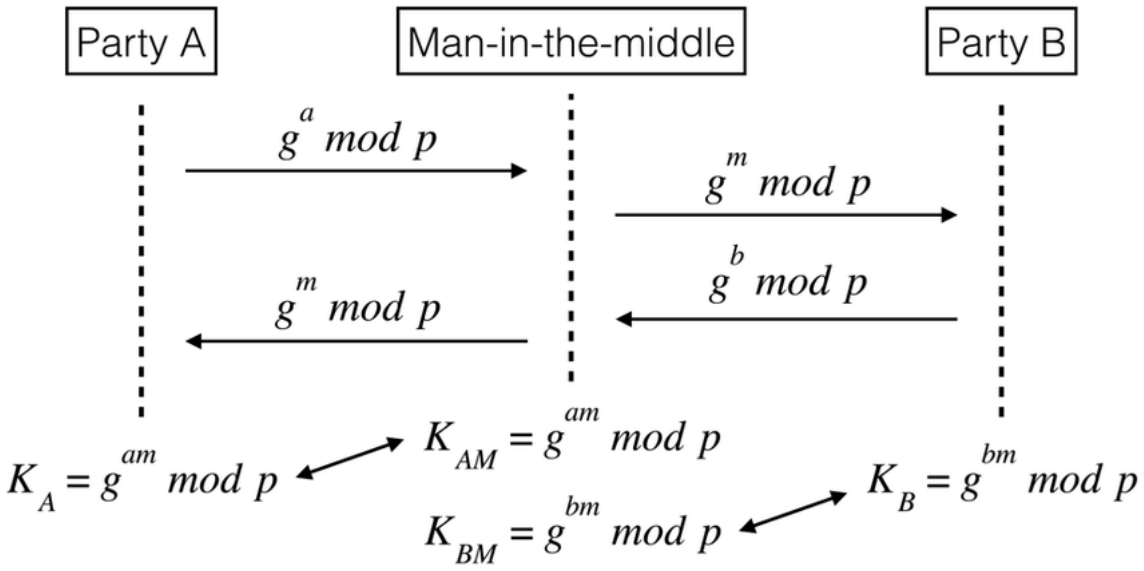

Figure 1 – MITM attack during DH exchange - https://www.researchgate.net/figure/The-man-in-the-middle-attacks-during-key-exchange-procedure_fig6_313738303

In Figure 1, party A is the initiator and party B is the responder. The attacker can attack the IKE AUTH exchange since the initiator and responder cannot verify the identity associated with the pre-secret master key.

(b) Try to explore possibilities to conduct a dictionary attack in IKEv2, when the pre-shared secret S_{pre} is a password with binary length 8 bits. (Hint: A failed execution may expose a value AUTH and the data it is protecting)

Attackers must wait initially for the victim to initiate an IKE exchange with a responder during an offline dictionary attack. The attacker will intercept the ClientHello message and prevent it from reaching the responder – the attacker will acts as the responder using IP spoofing and perform each of the necessary exchanges until message m5 is received from the victim as shown in Figure 2 below:

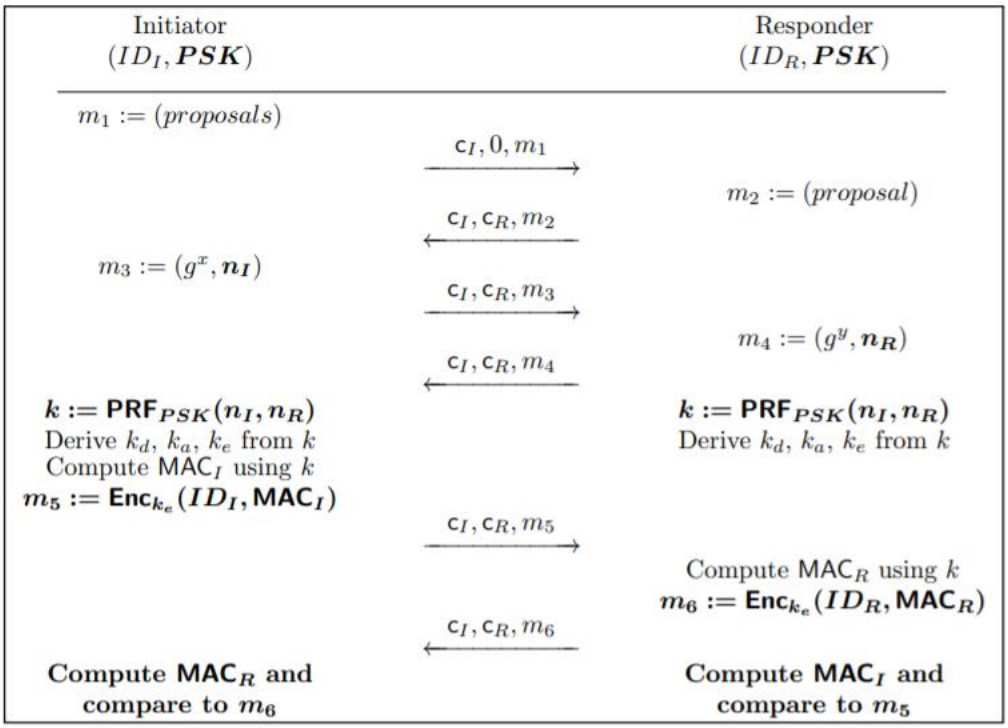

Figure 2 – IKEv2 offline dictionary attack exchanges - The Dangers of Key Reuse: Practical Attacks on IPsec IKE, pp. 13–14

Once m5 is received, the attacker has the ID and the MAC of the inititator, encrypted using k_e derived from $k = PRF_{PSK}(n_i, n_R)$, where PSK is the pre-shared key. ID_I is easily determined given it is usually assigned the IP address of the initiator. Given that the pre-shared key is 8 bits in length, an iterative offline dictionary attack need only to iterate a total of 256 permutations. For each PSK between 0 and 255, inclusive, k and k_e can be derived and m5 can be decrypted. The k_e value that decrypts m5 such that ID_I is the IP address, or another ID that the attacker knows, reveals the corresponding k and PSK values that the attacker needs. Once determined, the attacker can derive the other keys easily. This attack is made possible given the plaintext of m5 has a known structure that begins with ID_I, a known value for the attacker.

Exercise 5: Security analysis on TLS:

(c) Assume the key establishment algorithm is RSA, and the client authentication is not conducted, that is, message CertificateVerify is not sent. Try to identify an attack which hijacks the session by sending an attacker-generated "pre-master secret" to the server, where the messages Finished can carry along without being detected by either the client or the server.

An attacker can use a tool such as Wireshark to monitor network traffic between the client and server. The following is a sample scenario that entails its usage:

I. Client sends plaintext message ClientHello to the server. The message is not encrypted – thus the attacker knows client random and has knowledge of all the cipher suites requested.
II. Server sends plaintext message ServerHello to the client. The message is not encrypted – thus the attacker knows server random and has knowledge of the cipher suites that the server has chosen. The server will also send an SSL certificate as part of ServerHello, which the attacker can then use to contact the CA that issued the certificate.
    a. There is no ServerKeyExchange from the server under RSA
    b. There is no CertificateRequest given that client verification is not performed in this scenario
III. Attacker can actively intercept messages exchanged between the client and server.
    a. Under RSA, the attacker establishes the pre-secret master key α and encrypts it using the server's public key pk_S as follows: $E = (pk_S, \alpha)$ – this is generated as part of the ClientKeyExchange. The attacker is now capable of generating the Finished message on its side.
IV. The server and attacker have established a pre-message secret – the attacker also has knowledge of the client random and server random from part II, along with the cipher suite that was chosen by the server.
    a. Therefore, the attacker can generate the master secret and any other keying material necessary such as the client write key which is used to send messages to the server.
V. The master key is necessary to generate the Finished message – it will be used to generate the MAC key and the encryption key. The attacker will collect the ClientHello from the victim client, the server hello, and the ClientKeyExchange into a single message. This message is used with the MAC key to generate a tag. The message and tag are then combined to generate a payload which is encrypted using the encryption key, creating the Finished message.
    a. The attacker will send the ClientKeyExchange and Finished messages to the server – the server will respond with its Finished message directly to the attacker and the client will not have any knowledge of this message. Consequently, the attacker will have successfully hijacked the session and the Finished messages are generated without the server and client being aware of the attacker's presence.

(d) Explain why the attack identified in a) will not gain access to the server, if the client must enter a password before any further application data will be exchanged.

Clients are required to enter their password before application data can be transmitted by the server in TLS. This step is independent of the key establishment and authentication steps. Given that the client is the only party aware of the password, the attacker is unable to gain access to the data on the server without knowledge of that password. TLS was designed to establish a secure tunnel through which sensitive user data can be transmitted. Otherwise, sending the user's application data without requiring the user's password enables attackers to hijack a session with relative ease.

(e) Try to explain why key establishment algorithms RSA and DH cannot provide perfect forward secrecy.

Perfect forward secret can only be achieved if disclosure of long-term secret keying material does not compromise the secrecy of the exchanged keys from earlier sessions.

While using RSA for key transport under TLS, the pre-secret master key α is chosen by the client and encrypted using the server's public key pk_S such that only the server is able to decrypt the pre-secret master key. Though the pre-secret master key is newly chosen each session, the encryption is done using the same long-term key pk_S. Therefore, if the long-term key is compromised, attackers can decrypt any of the session keys that were exchanged in the current session, as well as in any previous session provided the encrypted key exchanges are stored. Consequently, RSA is unable to provide perfect forward secrecy.

Using DH, the client chooses a new key c for each session. The client computes $Y_c = g^c$ and sends Y_c to the server. The server does not choose a new key however, and instead the long-term key s is used to compute the pre-master secret key α at the client and server as $\alpha = g^{sc}$. Consequently, α is directly related to the long-term key s in each session and if this key is compromised, the attackers can decrypt any past or present session key as long as the ClientKeyExchange Y_cs are stored in advance. As a result, DH is also unable to provide perfect forward secrecy.

Exercise 8: A forgery attack on GHASH. GHASH is used in GCM in TLS and GCMP in WiFi, as well as EIA1 in 4G-LTE. In theory, it has been proved it is secure under the assumption that nonce cannot be reused. As you have seen, in the real world, in both 4G-LTE and WiFi, the nonce can be forced to repeat. Hence, an attacker is able to forge the authentication generated by GHASH. In the following, we will assume that a GHASH polynomial is evaluated in finite field GF(2^4), defined by $t(x) = x^4 + x + 1$, a primitive polynomial, and α is a root of t(x) in GF(2^4). We give the following two pairs of plaintext and ciphertext.

plaintext	ciphertext
M = 001100101111	C = 101000111001
M' = 100000110000	C' = 001011100101

where the right most bit is LSB and each ciphertext is generated by a random cipher.

(a) **** Let H = 0101 in GCMP, compute GHASH(C, H) and GHASH(C', H). Find a ciphertext which has a valid hash value.**

$$\boldsymbol{C_1 = 1010, C_2 = 0011, C_3 = 1001}$$

$\boldsymbol{Y_1} = C_1 * H = 1010 * 0101 = \alpha^9 * \alpha^8 = \alpha^{17} = \alpha^2 = \mathbf{0100}$
$\boldsymbol{Y_2} = (C_2 + Y_1) * H = (0011 + 0100) * 0101 = (\alpha + 1 + \alpha^2) * \alpha^8 = 0111 * \alpha^8 = \alpha^{10} * \alpha^8$
$= \alpha^{18} = \alpha^3 = \mathbf{1000}$
$\boldsymbol{Y_3} = (C_3 + Y_2) * H = (1001 + 1000) * 0101 = (\alpha^3 + 1 + \alpha^3) * \alpha^8 = 1 * \alpha^8 = \mathbf{0101}$

$$\boldsymbol{GHASH(C, H) = Y_3 = 0101}$$

$$\boldsymbol{C'_1 = 0010, C'_2 = 1110, C'_3 = 0101}$$

$\boldsymbol{Y'_1} = C'_1 * H = 0010 * 0101 = \alpha * \alpha^8 = \alpha^9 = \mathbf{1010}$
$\boldsymbol{Y'_2} = (C'_2 + Y'_1) * H = (1110 + 1010) * 0101 = (\alpha^3 + \alpha^2 + \alpha + \alpha^3 + \alpha) * \alpha^8 = \alpha^2 * \alpha^8$
$= \alpha^{10} = \mathbf{0111}$
$\boldsymbol{Y'_3} = (C'_3 + Y'_2) * H = (0101 + 0111) * 0101 = (\alpha^2 + 1 + \alpha^2 + \alpha + 1) * \alpha^8 = \alpha * \alpha^8$
$= \alpha^9 = \mathbf{1010}$

$$\boldsymbol{GHASH(C', H) = Y'_3 = 1010}$$

Given the linearity of GHASH, the following relation holds:

$$\boldsymbol{GHASH(C, H) \oplus GHASH(C', H) = GHASH(C \oplus C', H)}$$

A valid ciphertext with a valid hash value can be generated as follows:

$$\boldsymbol{C'' = C \oplus C' = 101000111001 \oplus 001011100101 = 100011011100}$$

(b) ** IN EIA1, let P = 1111, Q = 0001 and OTP = 0011 (i.e. without truncating), compute GHASH(M, P), and GHASH(M', P), the GHASH component in EIA1 for message M and M'.

$$\boldsymbol{M_1 = 0011, M_2 = 0010, M_3 = 1111}$$

$\boldsymbol{Y_1} = M_1 * P = 0011 * 1111 = \alpha^4 * \alpha^{12} = \alpha^{16} = \alpha = \mathbf{0010}$
$\boldsymbol{Y_2} = (M_2 + Y_1) * P = (0010 + 0010) * 1111 = \mathbf{0000}$
$\boldsymbol{Y_3} = (M_3 + Y_2) * P = (1111 + 0000) * 1111 = \alpha^{12} * \alpha^{12} = \alpha^{24} = \alpha^9 = \mathbf{1010}$

$$\boldsymbol{GHASH(M, P) = Y_3 \oplus LEN(M) \otimes Q \oplus OTP}$$

$\boldsymbol{LEN(M)} = 12\ bits = \mathbf{1100}$

$\boldsymbol{GHASH(M, P)} = 1010 \oplus 1100 \otimes 0001 \oplus 0011 = \mathbf{0101}$

$$\boldsymbol{M'_1 = 1000, M'_2 = 0011, M'_3 = 0000}$$

$\boldsymbol{Y'_1} = M'_1 * P = 1000 * 1111 = \alpha^3 * \alpha^{12} = \alpha^{15} = \mathbf{0001}$
$\boldsymbol{Y'_2} = (M'_2 + Y'_1) * P = (0011 + 0001) * 1111 = (\alpha + 1 + 1) * \alpha^{12} = \alpha * \alpha^{12} = \alpha^{13}$
$= \mathbf{1101}$
$\boldsymbol{Y'_3} = (M'_3 + Y'_2) * P = (0000 + 1101) * 1111 = \alpha^{13} * \alpha^{12} = \alpha^{25} = \alpha^{10} = \mathbf{0111}$

$$\boldsymbol{GHASH(M', P) = Y'_3 \oplus LEN(M') \otimes Q \oplus OTP}$$

$\boldsymbol{LEN(M')} = 12\ bits = \mathbf{1100}$

$\boldsymbol{GHASH(M', P)} = 0111 \oplus 1100 \otimes 0001 \oplus 0011 = \mathbf{1000}$

(c) IGNORE

(d) ** Show that after attacker intercepts the MAC-I(M) and MAC-I(M'), he can forge a valid MAC-I(M_{new}) where $\boldsymbol{M_{new} = 0010 \cdot (M + M') + M}$. (Hint. Show that $\boldsymbol{MAC - I(M_{new}) = \alpha^5[MAC - I(M) + MAC - I(M')] + MAC - I(M)}$)

Given the linearity of GHASH, the following relation holds:

$$\boldsymbol{GHASH(C, H) \oplus GHASH(C', H) = GHASH(C \oplus C', H)}$$

$$\boldsymbol{M_{new} = 0010 * (M + M') + M}$$

Because the MAC-I function is based on GHASH, MAC-I is also linear. As a result, it can be shown that **MAC-I(M_{new})** is linearly based on MAC-I(M) and MAC-I(M'):

$$MAC - I(M_{new}) = MAC - I(0110 * (M + M') + M)$$
$$= MAC - I\big(0110 * (M + M')\big) + MAC - I(M)$$
$$= 0110 * MAC - I(M + M') + MAC - I(M)$$

$$\boldsymbol{MAC - I(M_{new}) = \alpha^5[MAC - I(M) + MAC - I(M')] + MAC - I(M)}$$

Therefore, if an attacker intercepts both MAC-I(M) and MAC-I(M'), using their linear combination the attacker can forge $MAC - I(M_{new})$.