Homework #3 Student ID: **2018280070,** Name: **Peter Garamvoelgyi**

1. According to the description of RSA in the textbook, prove completely that…

a. First, let us prove that

$$(m^e \bmod n)^d \bmod n = m^{ed} \bmod n$$

We know that multiplication in modular arithmetic has the following property[1]:

$$(a \bmod n) \cdot (b \bmod n) = (ab \bmod n)$$

Using this, we can conclude that

$$\begin{aligned}(m^e \bmod n)^d &= (m^e \bmod n) \cdot (m^e \bmod n) \cdot \ldots \cdot (m^e \bmod n) \qquad (d \text{ times})\\ &= (m^e \cdot m^e \cdot \ldots \cdot m^e) \bmod n \qquad (d \text{ times})\\ &= m^{ed} \bmod n\end{aligned}$$

Thus

$$(m^e \bmod n)^d \bmod n = (m^{ed} \bmod n) \bmod n = m^{ed} \bmod n$$

That concludes the proof.

b. Second, let us prove that

$$x^y \bmod n = x^{y \bmod (p-1)(q-1)} \bmod n \qquad (p, q \text{ are primes})$$

$(p-1)(q-1)$ corresponds to Euler's Totient Function

$$\phi(pq) = (p-1)(q-1)$$

Let us assume that

$$y \equiv l \left(\bmod \phi(pq)\right) \qquad \text{i.e.} \qquad y = k \cdot \phi(pq) + l \qquad k, l \in \mathbb{N}$$

Using common identities of exponentiation and modular arithmetic, we can conclude that

$$\begin{aligned}x^y \bmod n &= x^{k \cdot \phi(pq)+l} \bmod n\\ &= \left(x^{\phi(pq)}\right)^k \cdot x^l \bmod n\\ &= \left(\left(x^{\phi(pq)}\right)^k \bmod n\right)(x^l \bmod n)\end{aligned}$$

Assuming that $gcd(x, n) = 1$ (i.e. x and n are relative primes), *Euler's theorem* states that

$$x^{\phi(pq)} \equiv 1 \ (\bmod n)$$

[1] book page 54

Substituting this, the equation above becomes

$$\begin{aligned} x^y \bmod n &= (1^k \bmod n)(x^l \bmod n) \\ &= x^l \bmod n \\ &= x^{y \bmod \phi(pq)} \bmod n \end{aligned}$$

That concludes the proof.

c. Finally, let us prove that

$$(m^e \bmod n)^d \bmod n = m$$

Given that

$$\begin{aligned} &n = pq \quad \text{primes} \\ &\gcd(\phi(n), e) = 1, \quad 1 < e < \phi(n) \\ &d \equiv e^{-1} \left(\bmod \phi(n)\right) \\ &m < n \end{aligned}$$

Where m, p, q, e are chosen and n, d are calculated.

Using our result from a. we know that

$$(m^e \bmod n)^d \bmod n = m^{ed} \bmod n$$

Using our result from b. we can reformulate

$$m^{ed} \bmod n = m^{ed \bmod \phi(n)} \bmod n$$

We also know that

$$ed \equiv 1 \left(\bmod \phi(n)\right)$$

Thus

$$m^{ed \bmod \phi(n)} \bmod n = m^1 \bmod n = m$$

2. In PGP, why is a message signed first and then encrypted together with the signature, instead of the other way around (encrypt the message and then sign)? Other than PGP, are there cases where message (including network packet) can be encrypted and then signed?

Generally speaking, we should follow sign-then-encrypt.

- The problem with sign-then-encrypt: The receiver cannot authenticate the received ciphertext. I.e. the receiver knows who wrote the message but does not know who encrypted it.
- The problem with encrypt-then-sign: The signature does not prove that the signer was aware of the plaintext before encryption.

In particular, sign-then-encrypt is vulnerable to *surreptitious forwarding*[2]: A message signed and encrypted by Alice to Bob can be re-encrypted by Bob and forwarded to Claire. Bob's actions will be transparent and Claire will believe that Alice wrote to her directly. This violates the principle of *non-repudiation*.

A possible attack scenario on encrypt-then-sign is as follows[3]: When the message sent is a password, the attacker could intercept the message and replace the signature with its own one, thus gaining unauthorized access to the protected resources.

PGP offers sign-then-encrypt because it still has stronger security guarantees than encrypt-then-sign. The problem of surreptitious forwarding can be handled by prepending the recipient to the plaintext: this way it is provable who the message was originally intended to.

Common protocols use the following schemes[4]:

- SSL: sign-then-encrypt
- SSH: encrypt-and-sign
- IPSec: encrypt-then-sign

[2] Source: http://world.std.com/~dtd/sign_encrypt/sign_encrypt7.html

[3] Source: https://crypto.stackexchange.com/a/5466

[4] Source: https://iacr.org/archive/crypto2001/21390309.pdf

3. In PGP, it compresses the message before encrypting it. In SSL, it does compression before encryption as well. Do we have to do compression first and then encryption, if we want to do both? Why or why not?

Compression after encryption does not cause any security issues but it can potentially reduce the compression rate dramatically.

In compression, we rely on low entropy: By exploiting regularities in the data (repeating characters or words in texts, similar colors in pictures, etc.) we can reduce the amount of information needed to represent it.

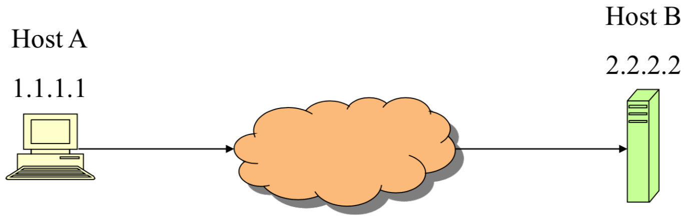
In encryption, we need to make sure our algorithm's output is of high entropy. Any regularities in the ciphertext could leak some information and be used to reproduce the plaintext.

Given this, the compression rate on low-entropy plaintext is usually much better than on high-entropy ciphertext.

4. (Bonus) In IPSec, SPI is used to index SA. Can we remove SPI from IPSec header and still get it work? What's the impact to communication and computation load (overhead) respectively?

SPI (Security Parameters Index) "*identifies which algorithms and keys are to be used for IPSec processing*". This configuration is called SA (Security Association). Active SAs are stored in the SA database (SADB) that is indexed using SPI, that simply serves as a unique 32 bit key.

On the other hand, the Security Policy Database (SPD) is basically a collection of policies describing how to handle certain traffic.



A's SPD

From	To	Protocol	Port	Policy
1.1.1.1	2.2.2.2	TCP	80	Transport ESP with 3DES

A's Outbound SADB

From	To	Protocol	SPI	SA record
1.1.1.1	2.2.2.2	ESP	10	3DES key

For the example above – given the assumption that we will use the same SA for packets in the same flow – we could directly index SADB using the standard 5-tuple of source and destination addresses and ports plus protocol.

In reality, however, one SA could be used for multiple connections and one connection could also use multiple SAs:

> *"For example, all traffic between two hosts may be carried via a single SA, and afforded a uniform set of security services. Alternatively, traffic between a pair of hosts might be spread over multiple SAs, depending on the applications being used (as defined by the Next Protocol and Port fields), with different security services offered by different SAs."*[5]

[5] https://tools.ietf.org/html/rfc2401