Exploration: Public and Private Key Cryptography

Introduction

In our final exploration we will take a look at encryption. We will examine the mechanisms for both public and private key encryption, and some of the most popular algorithms that make it all possible.

Encryption is the process of encoding data between a sender and a receiver, so that if the data is intercepted it cannot be decoded or used in any way. There are some very interesting algorithms that we will take a look at. These encryption algorithms require some number theory, which is not a prerequisite for this course but learning how they work requires only a bit of basic algebra to understand.

There are many encryption algorithms ranging from very simple substitution in ascii codes (extremely easy to crack), to algorithms that use very complex computations. Most of those use the modulo function with large primes. The two that we will look at are:

- *Private key* encryption (frequently called symmetric encryption)
- *Public key* encryption (frequently called asymmetric encryption)

Private Key Encryption (symmetric encryption)

With private key encryption (symmetric encryption), only the sender and receiver have the key. While the encryption/decryption algorithm might be public, the selection of the algorithm will normally not be publicized.

- (sender) For message M, with key K, the encrypted message E is
  - E = encrypt (K, M)
- (receiver) For encrypted message E, the original message is produced by the inverse of encrypt
  - M = decrypt (K, E)

The sender starts with message M and the private key K and applies the encryption algorithm to create encrypted message E. The Sender sends E to the receiver. The Receiver applies the decryption algorithm (inverse) to E with the private key K and the original message M is the output.

One flaw with all private key algorithms is that the private key and knowledge of the algorithm must be communicated to both parties, and of course that message could itself be intercepted.

It is easy to change the key, but difficult to ensure confidentiality of the key.

In spite of the flaws, private key encryption (symmetric encryption) is commonly used to enable secure third-party services in apps and websites. For example, to setup a payment gateway on a website using a credit card processing service.

Public Key Encryption (asymmetric encryption)

Public key encryption (symmetric encryption) is similar, but there are two keys involved: one public and one private. These two go together and are often referred to as key-pairs.

- (sender) For message M, with the destination user's public key (Kpublic), the encrypted message E is
  - E = encrypt (Kpublic, M)
- (receiver) For a message E (encrypted with the destination user's public key) the original message can be produced only by the destination user's private key (Kprivate)
  - M = decrypt (Kprivate, E)

A sender will take the receivers public key and encrypt the message using the algorithm. When the receiver gets the message that was encrypted with its public key, that message can ONLY be decrypted with its own private key. This requires quite a lot of background work (math) to get a key pair that will work in this manner. Public key encryption (symmetric encryption) makes it easy to change key and to ensure confidentiality, because the private key is never distributed.

RSA (RSA: Rivest, Shamir, Adleman algorithm)

The algorithm that we are going to demonstrate here is attributed to Rivest, Shamir, and Adleman and is appropriately referred to as RSA.

We are going to use some very small numbers to start with just for demonstration purposes:

- Kpublic = <3, 187>
  - Kprivate = <107, 187>
- Message = 25
- E = encrypt(Kpublic, Message)
  - = Message3 mod 187
  - = 253 mod 187 = 104
- M = decrypt(Kprivate, E)
  - = E107 mod 187
  - = 104107 mod 187 = 25 = Message

The public key has two components and the private key has two components. The second component in each of these will match. The message that we will be sending is just the number 25 to keep things simple. What we do in this algorithm is take the first part of the public key, raise the message to that power, then take the modulo of the second number of the second part of that key. If you carry this out, you get the encrypted message is the number 104.

It is important to understand that there is nothing that you can do with the public key and the encrypted message to get the original message back! There are also no computations that you can do with the public key to get the private key. When the encrypted message arrives at the receiver, the private key is used to decrypt the message. Recall that only one entity has the private key. The private key is 107, 187.

To decrypt the message we take the encrypted message E, raise it to the power of the first part of the private key (107), and mod that with the second part of the private key. This produces 25, which is our original message.

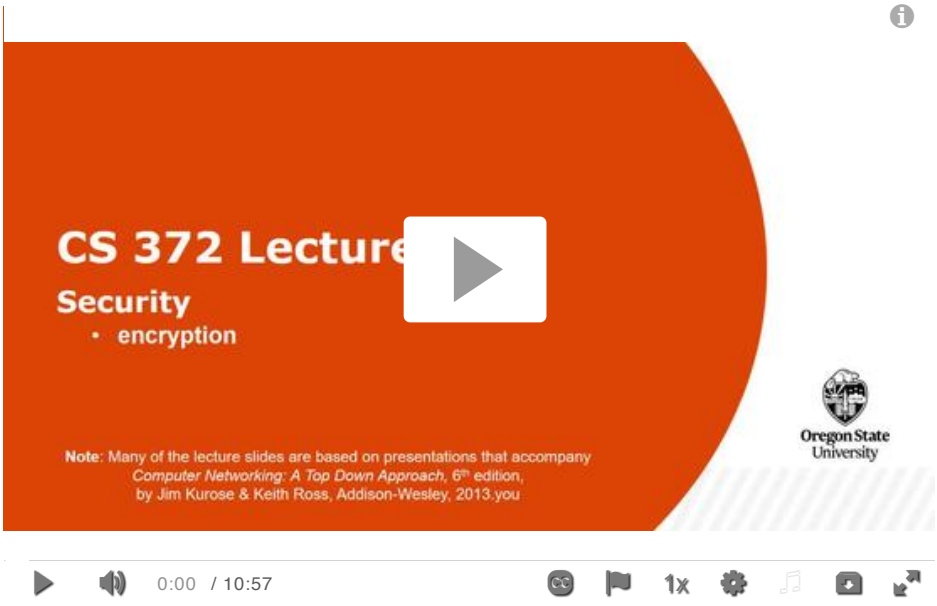
This concludes our brief introduction to Encryption. Be sure to watch the video lecture below for more details, including an in-depth discussion of the maths involved with choosing the RSA keys. Then head to the included Self-Check exercises to test your knowledge.

Video Lecture

Security Part 2

Note: Keep in mind that when the video mentions "**Private Key encryption**" it is often referring to "**symmetric**

encryption" (a somewhat more umbrella term, but used more frequently by security folk and industry).



(**PDF** (https://oregonstate.instructure.com/courses/1798856/files/83165295/download?wrap=1) (https://oregonstate.instructure.com/courses/1798856/files/83165295/download?wrap=1) |**PPT** (https://oregonstate.instructure.com/courses/1798856/files/83165084/download?wrap=1) (https://oregonstate.instructure.com/courses/1798856/files/83165084/download?wrap=1))

Self-Check Exercises

How might authentication work with public key encryption?

Turn

Card 1 of 1

Use the RSA algorithm discussed in lecture to develop a public key and a private key for public-key encryption. Let p = 5, q = 11, e = 7, m is the original message, c is the encrypted message.