Assignment 3

Due Friday, November 30th at 11:59:59pm

Submission Instructions

Assignments are to be completed individually. Place each answer in a clearly named file (*e.g.,* `q1.txt`). Answers must be in `txt`, `pdf` or `doc/docx`. Place all files, including answers and any code attachments in a `.zip` file and submit via OWL by the due date. Email submissions will not be accepted. As per the course late policy, **assignments will not be accepted more than 48 hours past the due date.**

1. [4 marks] Digital Signatures & Certificates

In this question you'll explore RSA signatures. Consider the "textbook" (i.e., unpadded) RSA signature scheme.

- **Generate:** Input: Security parameter b
  - Generate two large b-bit primes p, q
  - Compute $n = pq$ and $\phi = (p-1)(q-1)$
  - Pick integers e, d such that $ed \equiv 1 \bmod \phi$
  - Return private signing key (d, n), and public signature verification key (e, n)
- **Sign:** Input: (m, d, n). Message m is an integer $1 < m < n$.
  - Compute signature $s = m^d \bmod n$
  - Return s
- **Verify:** Input (s, m, e, n)
  - Compute $m' = s^e \bmod n$
  - Return $True$ if $m' = m$. Return $False$ otherwise.

(a) [1 marks] **Unpadded RSA**. Given two valid unpadded RSA signatures s_1, s_2 and public verification key (e, n) we discussed in the lecture how an attacker could create an existential forgery. Using either pseudocode or equations, show how an attacker could create an existential forgery using *only* the public key (e, n). Hint: Focus on the **Verify()** function. How could you make it return $True$ without knowing the private signing key d?

(b) [1 marks] **Digital certificates**. Download the following certificate:

https://whisperlab.org/information-security/assignments/rootca.pem

Use OpenSSL to parse the certificate and answer the following: Who is the common name of the issuer? When does the certificate expire?

(c) [2 marks] **Digital signature**. Using the steps outlined in the Week 9 lecture slides, manually verify the signature on the certificate, i.e., (1) compute $s^d \bmod n$ and (2) compute the hash of the certificate. Is this signature valid? Explain why or why not. Show your work. **Tip:** Python is a great choice for (1), and openssl is needed to compute (2).

2. [2 marks] Key Exchanges

In this question you'll explore Diffie-Hellman key agreement through a (mostly) realistic numeric example. In this scenario, your computer (the client) is attempting to create a shared secret with a website (the server). You will be using the standardized 2048-bit Diffie Hellman parameters (i.e., the prime p and generator g) specified in RFC 7919 (See `ffdhe2048` in Appendix A.1):

https://tools.ietf.org/html/rfc7919

The client's secret (i.e., a) will be your student number. The sever's secret (i.e., b) will be 4472. Write a program (Python is a great choice) to compute the following:

(a) the client's public key,

(b) the server's public key,

(c) the shared secret.

Submit these values and your code. **Hint:** You can test your implementation for correctness by checking that $(g^a)^b \bmod p = (g^b)^a \bmod p$.

3. [2 marks] TLS Key Kraziness

A single TLS connection involves *a lot* of keys. In this question you will enumerate *all* of the keys that go into making a single TLS handshake possible. This includes all public, private, and secret keys exchanged or generated during a TLS handshake. The particular ciphersuite we will consider is:

`TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`

Your answer should consist of 3 columns: a description of the particular key (*e.g.,* the server's public ECDHE key), the name of the entities who know the key (*e.g.,* everyone), and a sentence or two describing the purpose of the key (*e.g.,* used for public key agreement).

4. [2 marks] Investigate TLS Configuration

In this question you will examine a real-world web server's TLS configuration. Select a website that uses HTTPS. To avoid everyone picking the same website, pick a website that begins with the first letter of your first name. For example, if your name was Fred, you might pick `https://www.fortisinc.com`. Be sure to state the exact URL you examined.

Use SSL Test (https://www.ssllabs.com/ssltest/analyze.html) to analyize the website's TLS configuration and answer the following questions:

(a) What grade did the website recieve?

(b) Which versions of SSL/TLS does the server offer?

(c) What ciphersuite does the server most prefer?

(d) Does the server offer any weak cipher suites?