Assignment 2

Due Friday, November 18th at 11:59pm

Assignments are to be completed individually. We generally expect you to make an honest effort searching around online before contacting course staff with technical questions (`stackexchange.com` and `crypto.stackexchange.com` are great resources btw).

Submission Instructions

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. As per the course late policy, assignments can be uploaded only up to 48 hours past the due date.

1. [40 marks] Digital Signatures

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) [10 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).

Solution. Attacker picks an arbitrary value $1 < a < n$ and computes $b = a^e \bmod n$. The pair (b, a) forms a valid message/signature combination: a verifier would compute $a^d \bmod n$ and discover it was equal to b!

(b) [20 marks] **Padded RSA**. Write a program called `sign` that accepts a filename, and a private signing key (n, d) as input and returns an RSA signature of the file using PKCS 1.5 padding, SHA-256 hashing and ASN.1 encoding. See Lecture 9. The public key is contained in the file assignment2-values.txt. You do not need to implement signature verification, however the public verification exponent e is provided to allow you to check the correctness of your program. Use any programming language you wish, but make sure your code is well documented.

(c) [10 marks] **Padded RSA Cont'd**. Digitally sign your code file. Place the signature in a file called `signature`. If your code is contained across more than one file, sign the file containing the main function/method.

Submit your answers to the first two parts, along with your `sign` code file(s) and signature file.

2. [35 marks] Key Exchanges

In this question you'll explore Diffie-Hellman key agreement. In this scenario, your computer (the client) is attempting to create a shared secret with a website (the server).

(a) [20 marks] Write a program called `keyexchange` that accepts a description of a finite cyclic group (p, g), and a server public key ys. These values are contained in the file assignment2-values.txt. The program returns two values: the client's public key yc and the shared secret s. The client's private key x will be your student number. Use any programming language you wish, but make sure your code is well documented.

Solution. Suppose your student number was: 250123456. Your public key would be:

170165686842667747391788537967480665808966821135904856277638057410392137696107253157509381885512390783537336120789269140449701137253268504106819180965999419958076966523303922830649363271328575596363672914761461225762084169473162854529765049076668773494261436783729791556429604884144853616655355019203662881741

and the shared secret would be:

102301622381215918979618247164918872397862117344712160281748253411476703588503939751648436882165736441461312910097916419659324846744146830533197048238864825664457583310444625496568153173269745062868843286546638887859473422175933223223129761501010549990477333539077665664808728298867508061669739189828343240 76

(b) [10 marks] Using your program, compute the shared secret between the client and the server. Submit the following values: yc, the client's public key the shared secret s. Place these values in a file called $dhparams$.

(c) [5 marks] Recall we said in most cases you should not write your own cryptographic implementations (i.e., the "don't roll your own" principle), yet clearly we're not applying this principle for this assignment. Why might this be considered a reasonable exception to the rule?

Solution. It's ok to roll your own for educational purposes since it won't be used to *actually* protect any information. The main objective here is to get you to explore the innner worksings of some of these crypto primitives in a hands-on way, while remaining mindful about the reckless nature of DIY crypto implementations.

Submit your answer to part (c), along with your `keyexchange` code file(s) and `dhparams`.

3. [25 marks] Investigate SSL/TLS Configurations

In this question you will examine a real-world web server's TLS configuration. For this question, 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.

(a) [5 marks] Using Google Chrome, navigate to the website click the green padlock in the URL bar, then click on the Connection tab. Then answer the following questions:

(i) What encryption bit-level did you connect at?

(ii) What is the key-exchange mechanism?

(iii) What is the digital signature mechanism?

(iv) What is the symmetric-key encryption mechanism (including mode of operation)?

(v) What is the message authentication mechanism?

Solution. Answers will vary. For the MAC, the answer will either be the ciphersuite's stated hash function (if using e.g., AES-CBC) or the Galois hash (GHASH) if using AES-GCM. If for example the ciphersuite was `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`, the answer would be 128/Elliptic curve Diffie-hellman/RSA/AES-GCM (ie counter mode)/GCM (Galois hash)

(b) [10 marks] Visit the Qualsys SSL Sever Test website (`https://www.ssllabs.com/ssltest/`) and input the URL of your website. Wait for it to complete its analysis, then answer the following questions:

(i) What letter grade did your website receive? Explain why you think the website received the grade it did (for example, if it got a bad mark, what contributed to it?)

(ii) Which versions of SSL/TLS does the website support?

(iii) List the cipher suite that is most preferred by the server. If the server has no preference, state as such.

Solution. Answers will vary depending on website. Common TLS versions are 1.0, 1.1, 1.2. An example ciphersuite is: `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`

(c) [10 marks] Scenario: you have been tasked with configuring an HTTPS enabled webserver. Which TLS ciphersuite would you select as your default? Consult:
http://www.thesprawl.org/research/tls-and-ssl-cipher-suites/
for a list of choices. Justify your decision in a few sentences. This question is meant to be somewhat open-ended (there is no single "right answer") so feel free to be creative or to cite anything that you think would help convince us of the soundness of your decision.

Solution. Answers will vary depending on if you value speed, security of interoperability. A common rationale would be to say “I’m just going to look at a company with a good security track record like Google and follow their lead.” Or you might have said something like “I don’t trust the NSA so I don’t want to use elliptic curve crypto.”