Summer Project: Side Channel Attacks

Diclehan Erdal
Katrina Falkner, Yuval Yarom, Naomi Benger

January 28, 2014

Week 1

Learning goals:

- Set up GitHub.
- Learn how to use LaTeX.
- Commence reading/research.

Reading list:

- The Montgomery Powering Ladder

Progress/Tasks completed:
26/09/13

- Set up GitHub on my computer. Cloned the project repository and added additional resources to it. Learned how the GUI worked.
- Attempted to read The Montgomery Powering Ladder. Looked up what Abelian Groups and Lucas Chains are. Need to complete reading paper.
- Started a (rough) bibliography.
- Installed TexMaker and watched YouTube videos on tex document basics. Attempted to modify progress report, but couldn't view changes in the PDF, or create my own tex files.

Week 2

Learning goals:

- Create bibliography on TexMaker. i.e. work out/learn how bib files work
- Establish some cryptography basics to ensure better understanding of further reading.
- Reattempt The Montgomery Powering Ladder paper.

Reading list:

- The Montgomery Powering Ladder.
- Sections 19.7 McEliece, 19.8 Elliptic Curve Cryptosystems from Applied Cryptography.
- The following sections of An Introduction to Mathematical Cryptography: 1.1 Simple substitution ciphers, 1.2 Divisibility and greatest common divisors, 1.3 Modular arithmetic, 1.4 Prime numbers, unique factorization, and finite fields, 1.7 Symmetric and asymmetric ciphers, 5.1 Elliptic curves and 5.2 Elliptic curves over finite fields.

Progress/Tasks completed:
30/09/10

- Upon some web troubleshoot searches, discovered that installing MikTex and running TexMaker as admin solved the problem.

02/10/13

- Read sections 19.7 and 19.8 of Applied Cryptography. Didn't understand some of the notation in 19.7. The proceeding section was easy to understand, however I'm not familiar with how the Diffie-Hellman, ElGamal, or Schnorr algorithms work. ElGamal will be looked into (I believe there is something related to it in 20.1). Will look further into "RSA" as well as I've come across that before.
- Read the abstracts and introductions of the three papers on Side-Channel attacks analysis.

05/10/13

- Simple substitution ciphers: Simple, as the title suggests. Nothing new or immensely enlightening but a nice way to start the book.
- Divisibility and greatest common divisors: The Euclidean Algorithm was interesting. I was not familiar with it before unlike the other content of the section. I don't yet see how GCDs are directly relevant to Cryptography, but it was a fun read. Also, I learnt the concept of "relatively prime".
- Modular arithmetic: "$\mathbb{Z}/m\mathbb{Z}$ is the ring of integers modulo m". I recognise this notation and now know what it means! I was introduced to Eulers phi function and The fast powering algorithm in addition to that.

- Prime numbers, unique factorization, and finite fields: The Fundamental Theorem of Arithmetic was something I used to simplify surds but did not know of formally. Came across a written definition of a field. Learnt that that hollow F is used to represent fields, along with a value; that is equivalent to $\mathbb{Z}/p\mathbb{Z}$, and that the equality relationship between two elements belonging to each is different.

06/10/13

- With the aid of online resources, created LaTeX bibliography.

07/10/13

- Symmetric and asymmetric ciphers:
  Elliptic curves:
  Elliptic curves over finite fields:

- Read The Montgomery Powering Ladder.

Week 3

Learning goals:

- Digital Signatures (DSA)

Reading list:

- 20.1 Digital Signature Algorithm from Applied Cryptography.
- (if required)Relevant sections from Chapter 7 of An Introduction to Mathematical Cryptography

Progress/Tasks completed:

- 25?/11/13

Week 4*

Learning goals:

- Create OpenSSL EC key/s and a certificate
- Install Linux
- Build(?) OpenSSL

Reading list:

- Recovering OpenSSL ECDSA Nonces Using the Flush+Reload Cache Side-channel Attack
- Online resources

Progress/Tasks completed:
02/12/13

- Read *Recovering OpenSSL ECDSA Nonces Using the Flush+Reload Cache Side-channel Attack*.
- I downloaded Openssl 1.0.1e. Yuval showed me how to extract tar file using the CLI, how to build OpenSSL and brought ec2_mult.c to my attention.
- I was given the task of creating key/s and a certificate, and in addition to that the commands "man openssl", "man s_server", "openssl s_server" and "openssl s_client" to use afterwards. I tried reading the manuals but they didn't make a lot of sense to me. I commenced my key/cert making by running some Google searches on creating OpenSSL EC keys/certs. (The web browser I was using on the computer in the Honours lab kept crashing when I tried to open particular pages.)

03/12/13

- I decided that I couldn't get away with trying to program on Windows forever and decided to give Linux a go. Based on my Googling, Ubuntu seemed to be the OS that would suit me the most. I downloaded versions 12.04 LTS and 13.10 from the Ubuntu website, and tried booting each from a USB via UNetbootin, but couldn't due errors including "invalid or corrupt kenel image" and "unable to find a medium containing live file system".

  I tried searching for these problems, particularly on AskUbuntu, but the solutions recommended either didn't work or I didn't understand them. (Some of my attempts include using Universal USB Installer, instead of UNetbootin, changing USB ports, trying a different computer, using a different USB, trying a different version and making sure there was a 32/64bit match with what I had downloaded and the computer.)

  Eventually, I ended up going further back and looking at the past versions of Ubuntu and downloaded 12.10. When I attempted to install that I didn't have any problems on the laptop I installed it on! I went ahead and installed it on my my desktop computer too, however upon doing that I couldn't find Ubuntu on startup or any boot menus. I know it installed because it split my hardrive the way it was supposed to. I tried Googling the problem and tried changing boot options, but I didn't really reach a successful conclusion and left it the way it is.

04/12/13

- With Ubuntu all set up, I thought it would be useful to learn basic CLI commands that I didn't already know. I took note of most of them (even the ones I was already familiar with). I found **cp** and **mv** interesting, and looked up some manuals including **man intro** and **man man**.

- I continued to my search for how to create OpenSSL EC keys/certs, referring to a some sites I had taken note of on Monday and some ones I came across today. I came across a lot of RSA keys, but eventually came across ways to create EC keys and certificates, and how to look up curve types, which is by the command **openssl ecparam -list_curves**.

- I generalised two main different ways to EC create keys: one method used one command, and the other used two commands, one for the private key, and one for the public key.

  First method in the general form (I'm guessing this is only a private key):

  ```
  openssl ecparam -out keyFileName.pem -name curveName -genkey
  ```

  Whereas the second method I generalised is in the form (first private key, then public key:

  ```
  openssl ecparam -genkey -name curveName -noout -out privateKeyFileName.pem
  ```

  ```
  openssl ec -in privateKeyFileName.pem -pubout -out publicKeyFileName.pem
  ```

  Based on the differences between the first and second key, I decided that the order of the arguments do not matter. I had another look at the openssl manual, which became less foreign to me when I read it this time. I noticed that there was a description for ”ecparam” under commands in the manual, which apparently is an EC parameter manipulator and creator. That made sense as to why it was used in the creation of keys and used when looking up curves. Also from the manual, I learnt that **genkey** is used in the ”Generation of Private Keys or Parameters”.

  I also worked out that -out is followed by the name of the file that will be created, and -in is used to use another file in the creation of a file. I know that the parameter after -name is the name of the curve, which can be looked up via the list_curves command I mentioned earlier. I'm not sure why -noout or -pubout were used.

- Similarly with creating keys, based on my Googling, I worked out a way to create certificates. A general command I wrote is as follows.

  ```
  openssl req -new -key aPrivateKey.pem -x509 -nodes -days N -out certificateName.pem
  ```

  In reference to the manual, I decided that **req** was simply a request for a certificate, -x509 was a standard parameter used to create certificates, as with -nodes and -new. I could see how what followed -key corresponded to a private key used to create the cert, and how -days were how long the cert would be valid for.

- After playing around with what I researched, I ended up generating keys and certs using the following commands

  ```
  openssl ecparam -genkey -name sect571r1 -noout -out privatekey.pem
  openssl ec -in privatekey.pem -pubout -out publickey.pem
  openssl req -new -key privatekey.pem -x509 -nodes -days 90 -out cert.pem
  ```

  I chose that particular curve because it was used for testing in the paper I read on Monday, and 90 days because that seemed reasonable for my project.

- I had a look at the s_server and s_client manuals as instructed on Monday. From both manuals I recognised cert and key parameters, and decided that I would possible use **openssl s_server -cert cert.pem -key privatekey.pem** and **openssl s_client -cert cert.pem -key privatekey.pem** with additional parameters.

- I installed gcc/g++ via the commands:

  ```
  sudo apt-get update
  sudo apt-get upgrade
  sudo apt-get install build-essential
  gcc -v
  make -v
  ```

  Then tested gcc with a c file and g++ with a c++ file.

05/12/13

- I showed Yuval the keys and certificate I made. He used **openssl s_server -key privatekey.pem -cert cert.pem &** as I had (almost) predicted and said the & was used to run it in the background. For s_client he used a command that didn't show up in my history afterwards for some reason, but it was something along the lines of **openssl s_client -connect localhost:4433**. I was reminded to write code combined with code that Yuval emailed me to measure the time/memory.

- I tried to time how long it took to print a simple message, but couldn't as I kept getting receiving *error: unsupported instruction 'mov'*.

- Additionally, I tried compiling random files in located in openssl-1.0.1e/crypto/ec (including ec2_mult.c), but had errors with that also.

Week 5*

Learning goals:

-

Reading list:

-

Progress/Tasks completed:
09/12/13

- Started writing up my progress report properly. (I need to update it daily from now on!)
- I need to make up for my lack of work today by doing something over the weekend.

10/12/13

- I showed Yuval my tester time measurement code, and he played around, modifying it to prevent the error. I'm not actually sure how he did it though.
- We changed the code to run a for loop up to a million instead of printing a message, and compared it with a for loop indexing up to twice that, two million. Despire receiving unstable results each time, it could be seen that doubling the index resulted in a larger number. It wasn't exactly twice as much, but a bit less.
- Each time we ran the code that printed a message, the time was different. According to Yuval, it isn't stable on laptops. He said we needed to use a different machine to run tests on.

11/12/13

- I was given a book titled Programming in C by Yuval, from Katrina.
- We added time measurements to the ec2_mult.c file. Yuval told me that I would have to remake OpenSSL every time I made changed to the code. He ran my server and client commands, and acheived time cycle measurements, from the changes made to ec2_mult.c. (I remember it being today)
- I was introduced to the ssh command, and given access to another computer/server which I could access. As I use the following command to access it, I will simply refer to it as *the GDI.* (Perhaps it would help if I learnt what it stood for.)
- As I am going to be running tests on GDI, Yuval introduced me to the text editor *Vi*, which I can use through terminal to modify the necessary files.

  ssh myUniUsername@gdi.cs.adelaide.edu.au

  Fixed code, ssh, introduced to vi,

12/12/13

- I tried getting time measurements of the self-signing certificate, but I couldn't work out how Yuval did it. I tried compiling ec2_mult.c after connecting to the server and client, but kept gettinge errors with the compiling. I tried looking through my history to see if any extra commands were used, but couldn't find anything different or useful.
- I was able to connect to gdi from home by connecting to uss.cs.adelaide.edu.au with ssh first, and connecting to gdi from there as I did at the University.

13/12/13

- After some intense Google searches, with the aid of a website that had different examples of the uses of **scp**, I worked out how to copy files to university servers: **scp fileName a1647317@uss(or gdi).cs.adelaide.edu.au:/location/fileName**

  I needed to add the argument **-r** while copying directories.
- I played around with Vi, using a wiki page to learn the basics. I found it a little bit annoying that whenever I tried using my arrow keys in the insert mode, random letters would be typed instead of blinking curser moving.
- I copied my timeMeasurements code to the GDI, and compared my results with ones I got from my laptop, and found that the GDI ones were much more stable (i.e. the values were closer together each time I ran the code).

14/12/13

- I read the introduction and Fundementals sections of *Programming in C*. I now I understand why I use ./a.out
- The Fundementals chaper and Vi really got me thinking about the development of Computer Science, and lead me to appreciate and take joy out of the little things a bit more.
- I learned that **make clean** ”removes all the temporary files created by past builds”.

Week 6*

Learning goals:

-

Reading list:

-

Progress/Tasks completed:
06/01/14

- I worked out how to use **wget**, but couldn't see how I would use it. I asked Yuval, and it turns out he suggested it so I could download OpenSSL onto the GDI. I decided to keep that in mind for future experiences as I had already copied my OpenSSL files using scp.
- I found out that the reason I couldn't get any time cycle measurment results with my server and client was because I needed to use openssl in the apps folder, and that compiling ec2_mult.c wasn't right. **openss-1.0.1e/apps/openssl s_server(and s_client)... etc.**

07/01/14

- Meeting with Naomi and Katrina

09/01/14

- mean (counter, then array+research), sd... 0's for sd. Think it's because decimal numbers are being cut off with unsigned long.

10/01/14

- Asked if I should run tests on all curves. A: not prime ones. Made a list from ecparam -list_curves

12/01/14

- "unsigned long array[]" problem while trying to use functions... Took forever.

Week 7*

Learning goals:

-

Reading list:

-

Progress/Tasks completed:
13/01/14

-

15/01/14

- Read error...

16-17/01/14

- Boot, drive wipe, recovery...

18/01/14

- Used try Ubuntu to connect to server and run tests.

Week 8*

Learning goals:

-

Reading list:

-

Progress/Tasks completed:
20/01/14

-

Week 9*

Learning goals:

-

Reading list:

-

Progress/Tasks completed:
27/01/14

-

IGNORE

IGNORE Learning goals:

- Side Channel Attacks (analysis, countermeasures).

Reading list:

- (if required)Relevant sections from Chapter 7 of An Introduction to Mathematical Cryptography
- Side Channel Attacks on Implementations of Curve-Based Cryptographic Primitives
- Countermeasures against Side-Channel Attacks for EC Cryptosystems
- Low-Cost Solutions for Preventing Simple Side-Channel Analysis Side-Channel Atomicity