**John Carroll  
jcc0044/902521946**

Auburn University – Fall 2015 COMP 5730/6730 Instructor: A. Skjellum; TA: A. Ravipati September 9, 2015

Note: “SHORT WRITE UPS” comprise 4-5 complete English sentences; diagram if helpful; bullet points OK too. Cite any sources (you can assume that we know you will read and reference the articles given below).

Start in-class, finish by Friday night, turn in via Canvas by 11pm on 9/11/15.

**All Students:**

1) SHORT WRITE-UP: Read about discrete logarithm (10pts)

https://en.wikipedia.org/wiki/Discrete\_logarithm

Write a brief summary of how it works, and how you might use it; experiment with the prototype code provided.

How might this be relevant to encryption computations?

Answer:  
A discrete logarithm is an integer *k* solving the equation *bk* = *g*, where *b* and *g* are elements of a finite group.  
  
Computing discrete logarithms is believed to be difficult. No efficient general method for computing discrete logarithms on conventional computers is known, and several important algorithms in public-key cryptography base their security on the assumption that the discrete logarithm problem has no efficient solution.

2) SHORT WRITE-UP: Read about Diffie-Hellman Key Exchange (20pts)

D-H allows for two parties with no prior knowledge of each other to establish jointly a shared secret key (e.g., a session key for symmetric key crypto) over an insecure channel.

https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman\_key\_exchange

* a) Explain when this can work safely, when it will have limitations (see discussion of Eve in the article). Are there improvements?

**ANS:** The protocol is considered secure against eavesdroppers if *G* and *g* are chosen properly.

The eavesdropper ("Eve") would have to solve the Diffie–Hellman problem to obtain*gab*. This is currently considered difficult.

b) Relate to the discrete logarithm

**ANS:** An efficient algorithm to solve the discrete logarithm problem would make it easy to compute *a* or *b* and solve the Diffie–Hellman problem, making this and many other public key cryptosystems insecure.

c) Why bother with public-key crypto if this works?

**ANS:** Because The method was followed shortly afterwards by RSA, an implementation of public-key cryptography using asymmetric algorithms. Perhaps the biggest reason is because Diffie–Hellman cannot be used to sign certificates

Another variation is Elliptic Curve Diffie-Hellman.

https://en.wikipedia.org/wiki/Elliptic\_curve\_Diffie%E2%80%93Hellman

Why is this different or better? Why is Elliptic Curve Crypto?

**ANS:** It is an anonymous key agreement protocol that allows two parties, **each** having an elliptic curve public–private key pair, to establish a shared secret over an insecure channel. So it actually requires the two parties to have the same thing.  
  
It is cryptography because the shared secret may be directly used as a key, or to derive another key, which can then be used to **encrypt** subsequent communications using a symmetric key cipher. Also, another strengthening point is because it is a variant of the Diffie-Hellman protocol.

(3) Quick answers: Read about “Secret Sharing” (10pts)

https://en.wikipedia.org/wiki/Secret\_sharing

* a) Define it: **Secret sharing** (also called **secret splitting**) refers to methods for distributing a *secret* amongst a group of participants, each of whom is allocated a *share* of the secret.
* b) When useful?   
  **ANS:** Secret sharing schemes are ideal for storing information that is highly sensitive and highly important. Examples include: encryption keys, missile launch codes, and numbered bank accounts.
* c) What limitations?

Each share of the secret must be at least as large as the secret itself. This result is based in information theory, but can be understood intuitively. Given *t-1* shares, no information whatsoever can be determined about the secret. Thus, the final share must contain as much information as the secret itself. There is sometimes a workaround for this limitation by first compressing the secret before sharing it, but this is often not possible because many secrets (keys for example) look like high-quality random data and thus are hard to compress.

All secret sharing schemes use random bits. To distribute a one-bit secret among threshold *t* people, *t-1* random bits are necessary. To distribute a secret of arbitrary length entropy of *(t-1)\*length* is necessary.

**Grad students (undergrads may do also for more points):**

4) SHORT WRITE-UP: Read about Montgomery multiplication (10pts):

http://www.hackersdelight.org/MontgomeryMultiplication.pdf https://en.wikipedia.org/wiki/Montgomery\_modular\_multiplication

Write a brief summary of how it works, and how you might use it; experiment with the prototype code provided.

How might this be relevant to crypto computations?  
  
**ANS:** Montgomery multiplication is a method for performing fast modular multiplication. Given two numbers *a* and *b* modulo a positive integer *N*, Montgomery multiplication computes *ab mod N*.   
  
Many important cryptosystems such as RSA and Diffie–Hellman key exchange are based on arithmetic operations modulo a large number, and for these cryptosystems, the increased speed afforded by Montgomery multiplication can be important in practice.