Homework 4 Written Solutions

Written Part

4. In defining `babyGiant` didn't always know the exact order of g, and couldn't necessarily check it directly. Therefore the exact proofs of correctness of these algorithms in class need to be modified to make sure our algorithms work. Let's do this here.

   (a) If `babyGiant` doesn't receive an order as an input it defaults to $|g| = p-1$. Explain why this is just assuming that g is a primitive root. Suppose g is not a primitive root but `babyGiant` still assumes $N = p-1$. Prove that `babyGiant` returns the correct logarithm.

   Proof. Note, I'm just going to rehash the proof we gave in class, and point out exactly why (and where) it works in this case too. It is ok for you to cite the proof from class as long as you fully explain why it still works in this case.
   The babysteps giantsteps algorithm computes two lists sets $n = \lfloor N \rfloor + 1$. Then computes two lists

   $$\{e, g, g^2, g^3, \cdots, g^n\} \quad \text{and} \quad \{h, hg^{-n}, hg^{-2n}, \cdots, hg^{-n^2}\}.$$

   To prove correctness, it suffices to show the lists overlap (since if $g^i = hg^{-nj}$ we get that $g^{i+nj} = h$ and our log is $i + nj$). Now suppose $g^x = h$, and divide x by n, so $x = qn + r$ for $0 \leq r < n$. In particular g^r is in the first list. We know that $x < |g| \leq N$ (here is the only place the proof differs, where $|g| \leq N$ instead of $|g| = N$). Then we can solve for q:

   $$q = \frac{x-r}{n} < \frac{N}{n} \leq n.$$

   Therefore hg^{-q} is in the second lists, but this equals g^r as $g^{qn+r} = g^x = h$. So the lists overlap.
   The main takeaway is that in class we saw if $N = |g|$ then the lists overlap. Therefore if $N > |g|$ the lists are longer than they need to be to overlap, so that of course they still do. □

   (b) Even if the order given as input is not entirely correct (but say a multiple of the actual order), the algorithms work. Still, there are drawbacks to not having the correct order. What are they? (e.g., if using $p-1$ in `babyGiant` always works, why don't we just always set $N = p - 1$?).

   Proof. We may end up doing many more computations if we do not put in the correct order. For instance, the lengths of the lists generated in `babyGiant` depends on the order the algorithm is fed. For example, if we were to use Pohlig Hellman while still allowing `babyGiant` to assume an order of $p - 1$ at each step we wouldn't save any time (and in fact could very likely end up losing time!). □

   (c) Baby steps-giant steps as we introduced in class took $\mathcal{O}(\sqrt{N} \log N)$ because of how long it took to find an element in both the baby steps list and the giant steps list. In question 2 we implemented a hash table instead of a list, and checking if an element is in a hash table takes $\mathcal{O}(1)$ steps (which doesn't depend on the size of the table!). Use this fact to prove that your implementation runs in $\mathcal{O}(\sqrt{N})$ steps instead.

Proof. First we generate the babystep list, which requires multiplying by g $n = \mathcal{O}(\sqrt{N})$ times. We then invert the last element g^n which takes $\mathcal{O}(\log N)$ using the extended Euclidean algorithm. Then we begin generating the giant step list. This requires going through a loop at most $n = \lfloor \sqrt{N} \rfloor + 1$ times, where each step in the loop consists of a multiplication by g^{-n} (which we already know because it is the the last element generated in the babysteps list), following by a check of whether the element generated is in the babysteps set (which is $\mathcal{O}(1)$). So this is $\mathcal{O}(\sqrt{N})$. Finally, when we find a match we check its index in the babysteps list, which requires 1 search through the babysteps list which is n elements long, so this search also takes $\mathcal{O}(\sqrt{N})$. In total we have:

$$\mathcal{O}(\sqrt{N}) + \mathcal{O}(\log N) + \mathcal{O}(\sqrt{N}) + \mathcal{O}(\sqrt{N}) = \mathcal{O}(\sqrt{N}).$$

□

In class we showed that an Elgamal oracle can solve the Diffie-Hellman problem. Let's show the other direction, and conclude that Elgamal and Diffie-Hellman are equally difficult.

5. Suppose you have access to an oracle who can solve the Diffie-Hellman problem. That is, for any prime p, given g^a and $g^b \mod p$, the oracle can tell you $g^{ab} \mod p$. Show that you can use this oracle to crack the Elgamal Public Key Cryptosystem. Precisely, suppose Alice publishes a prime p, an element $g \in \mathbb{F}_p^*$, and a public key A, and Bob sends the cipher text (c_1, c_2). Consult with the oracle to find the message Bob sent.

   Proof. We know (or have intercepted) A, c_1, and c_2. We also know the encryption means, so that Alice has a secret key a, and Bob chose a random k and is encrypting a message m. We don't know a, k, or m, but we do know:

   $$\begin{aligned} A &\equiv g^a \mod p \\ c_1 &\equiv g^k \mod p \\ c_2 &\equiv mA^k \mod p \end{aligned}$$

   We then consult with our Diffie Hellman Oracle, having them solve the DHP for $A = g^a$ and $c_1 = g^k$. They return $g^{ak} = A^k$. We then compute the inverse $A^{-k} \mod p$ (which is fast) and multiply by c_2 to recover m. □

Problem 2.10 in [HPS] gives an example of a cryptosystem where Alice and Bob need to send two rounds of messages back and forth to communicate. We reproduce it here. It might be fun to follow along on cocalc!

3. Alice and Bob decide on a prime $p = 32611$. The rest is secret. But any information crossing the middle channel (via the arrows) should be assumed to be intercepted by Eve.

	Alice	Eve	Bob
1.	Alice has message $m = 11111$		
2.	Alice chooses random $a = 3589$		
3.	Alice computes $u = m^a \mod p = 15950$	$\longrightarrow$	Bob recieves u
4.			Bob chooses random $b = 4037$.
5.	Alice recieves v	$\longleftarrow$	Bob Computes $v = u^b \mod p = 15422$
6.	Alice knows $a' = 15619$		
7.	Alice computes $w = v^{a'} \mod p = 27257$	$\longrightarrow$	Bob recieves w.
8.			Bob knows $b' = 31883$
9.			Bob computes $w^{b'} \mod p = 11111$. That's m!

(a) Notice how Alice knows a second exponent $a' = 15619$ in step 6. Where does this number come from and how does this relate to $a = 3589$ from step 2? Similarly how do Bob's exponenets $b = 4037$ and $b' = 31883$ relate? Use this information to explain how the algorithm works.

Notice that

$$aa' = 56056591 = 1 + 1719 * (32610) \equiv 1 \mod p - 1.$$

So $a' \equiv a^{-1} \mod p - 1$. Therefore for every x we have:

$$x^{aa'} = x^{1+1719(p-1)} = x(x^{p-1})^1 719 \equiv x \mod p$$

by Fermat's little theorem. Similarly $bb' \equiv 1 \mod p - 1$ so that for all nonzero x we have $x^{bb'} \equiv x \mod p$. Therefore:

$$w^{b'} \equiv v^{a'b'} \equiv u^{ba'b'} \equiv m^{aba'b'} = (m^{aa'})^{bb'} \equiv m \mod p.$$

(b) Formulate a general version of this algorithm using variables and show that it works in general.

We will use a table like above:

	Alice	Eve	Bob
1.	Alice has message m		
2.	Alice chooses a with $\gcd(a, p-1) = 1$		
3.	Alice computes $u = m^a \mod p$	$\longrightarrow$	Bob recieves u
4.			Bob chooses b with $\gcd(b, p-1) = 1$.
5.	Alice recieves v	$\longleftarrow$	Bob Computes $v = u^b \mod p$
6.	Alice computes $a' = a^{-1} \mod p - 1$		
7.	Alice computes $w = v^{a'} \mod p$	$\longrightarrow$	Bob recieves w.
8.			Bob computes $b' = b^{-1} \mod p - 1$
9.			Bob computes $w^{b'} \mod p$. That's m!

By Fermats little theorem, if $cc' \equiv 1 \mod p - 1$ then $cc' = 1 + k(p-1)$ so that for all nonzero x we have

$$x^{cc'} = x(x^{p-1})^k \equiv x \mod p.$$

Then we may compute as above that

$$w^{b'} \equiv v^{a'b'} \equiv u^{ba'b'} \equiv m^{aba'b'} = (m^{aa'})^{bb'} \equiv m \mod p$$

proving correctness.

Although I didn't ask this it is interesting to briefly discuss time complexity. Exponentiating in steps 3,5,7,9 is easy, and computing inverses in steps 6,8 is too. The tricky part may be steps 2 and 4, where Alice and Bob need to choose units $\mod (p-1)$. Checking whether a given number is prime to $p-1$ is easy, but on average how many would they need to check before they stumbled upon a unit? The answer certainly depends on $p-1$. For example, if $\varphi(p-1) = |(\mathbb{Z}/(p-1)\mathbb{Z})^*|$ is large, then you have a high probability of finding a unit. But what if $\varphi(p-1)$ is small?

There are two ways of doing this. First is one could choose a (or b) to be a prime number smaller than $p-1$. Of course, if an attacker knows this they can easily narrow down a search for a (or b) and find the message. That said, on average I believe $\varphi(n) \sim \frac{6}{\pi^2} n$, so you actually have a pretty good shot of finding a number prime to n by randomly guessing. One could also arrange for a prime p where $\varphi(p-1)$ is very large (which is actually protects you against discrete log attacks like we have/will discuss(ed)).

(c) Can a solution to the DLP break this cryptosystem? Justify your answer.

Proof. You can use a DLP oracle to break the cryptosystem. In particular, In step 3 you intercept u. Then in step 5 you intercept $v \equiv u^b$. You consult with the DLP oracle to compute $\log_u(v) = b$, and then can quickly compute $b' = b^{-1} \mod p-1$. In step 7 you intercept w and can yourself compute $w^{b'}$ which recovers m. □

(d) Can a solution to the DHP break this cryptosystem? Justify your answer.

Proof. You intercept $u = m^a = m^{abb'}$, $v = m^{ab}$, and $w = m^{aba'}$. In particular, you ask the Diffie-Hellman oracle to solve the DHP with

$$\begin{aligned} g &= v = m^{ab} \\ g^{a'} &= w = m^{aba'} \\ g^{b'} &= u = m^{aab} \end{aligned}$$

and the oracle tells you $g^{a'b'} = m^{aba'b'} = m$. □

4. In Homework 2 we studied square roots mod p. Let's use this to study square roots modulo p^e for some positive exponent e. Let p be a prime not equal to 2, and let b be an integer not divisible by p. Suppose further that b has a square root modulo p, i.e., the congruence:

$$x^2 \equiv b \mod p,$$

has a solution.

(a) The following fact may be handy. Show that $(a+b)^p = a^p + b^p + pab(\text{stuff})$. Conclude that $(a+b)^p = a^p + b^p \mod p$. (This is often called the *freshman's dream* by cynical math professors who have seen a similar formula on too many calculus exams).

Proof. The binomial theorem says:

$$(a+b)^p = \sum_{i=0}^{p} \binom{p}{i} a^i b^{p-i}$$

Pulling out the terms for $i = 0$ and $i = p$ this becomes:

$$(a+b)^p = a^p + b^p = \sum_{i=1}^{p-1} \binom{p}{i} a^i b^{p-i}$$

Since a and b divide each term in the sum, it only remains to show $\binom{p}{i}$ is divisible by p for each $i = 1, \cdots, p-1$. Recall that,

$$\binom{p}{i} = \frac{p!}{i!(p-i)!},$$

which has a factor of p in the numerator. But all the factors in the denominator are smaller than p, so the factor of p in the numerator cannot be cancelled out. The freshman's dream then follows by reducing mod p. □

(b) Show that for every exponent $e \geq 1$, b has a square root module p^e. That is, the congruence

$$x^2 \equiv b \mod p^e$$

has a solution. (**Hint:** Use induction on e, finding a solution modulo p^{e+1} by modifying the solution modulo p^e.)

Proof. We proceed by induction on e. For $e = 1$ we are assuming b has a square root. For the general case we may let β be a square root of b modulo p^e. This means that there is some integer k such that:

$$\beta^2 = b + kp^e.$$

One can raise both sides of this equation to the pth power and apply the Lemma:

$$\beta^{2p} = b^p + k^p p^{pe} + (b)(kp^e)(p)(\text{stuff}).$$

In particular:

$$\beta^{2p} \equiv b^p \mod p^{e+1}.$$

As b is not divisible by p, $\gcd(b, p^e) = 1$ so that we can divide the congruence above by b. In fact, we divide by b^{p-1} so we can solve for b.

$$b^{1-p}\beta^{2p} \equiv b \mod p^{e+1}.$$

Lastly, we notice that $1-p$ is even, so that the right hand side of the congruence becomes:

$$b^{1-p}\beta^{2p} = \left(b^{\frac{1-p}{2}}\beta^p\right)^2,$$

so that we see $b^{\frac{1-p}{2}}\beta^p$ is a square root of b modulo p^{e+1}. □

(c) Let $x = \alpha$ be a square root of b modulo p. Prove that in part (b) we can find a square root β of b mod p^e such that $\alpha \equiv \beta \mod p$.

Proof. Suppose $\beta^2 \equiv b \mod p^e$. Then in particular we know $\beta^2 \equiv b \mod p$. By HW2 Problem 8(a) we know $\beta \equiv \pm\alpha \mod p$. If $\beta \equiv \alpha \mod p$ then we are done, otherwise if we replace β with $-\beta$ we see that

$$(-\beta)^2 = \beta^2 \equiv b \mod p^e,$$

so that $(-\beta)$ is a square root of $b \mod p^e$ which is congruent to $\alpha \mod p$ as desired. □

(d) Suppose β, β' are two square roots of b mod p^e, and further that they are both equivalent to $\alpha \mod p$ as in part (c). Show that $\beta \equiv \beta' \mod p^e$.

Proof. Since $\beta \equiv \beta' \equiv \alpha \mod p$, then we know that $\beta + \beta' \equiv 2\alpha \mod p$. Since p is odd, $2\alpha \not\equiv 0 \mod p$, so that p does not divide $\beta + \beta'$. By assumption, p^e divides $\beta^2 - \beta'^2 = (\beta - \beta')(\beta + \beta')$. But we just saw that $\gcd(p^e, \beta + \beta') = 1$, so that in fact $p^e | (\beta - \beta')$ completing the proof. □

(e) Conclude that the congruence $x^2 \equiv b \mod p^e$ has either 2 solutions or 0 solutions. (Use HW2 Probem 8).

Proof. Suppose there aren't 0 solutions. We let β is a solution, and let α be reduction of $\beta \mod p$. Then $-\beta$ is another solution. If γ is a third solution the $\gamma \equiv \pm\beta \mod p$ by HW2 Problem 8(a), so that by part (c) above we have that $\gamma \equiv \pm\beta$. So there are exactly 2 solutions. □

5. Let G and H be groups, and denote their multiplication rules by $*_G$ and $*_H$ respectively. A function $\varphi : G \to H$ is called a *homomorphism* if for all $a, b \in G$:

$$\varphi(a *_G b) = \varphi(a) *_H \varphi(g).$$

(a) Recall that the identiy of a group $e \in G$ is an element such that for every $g \in G$,$e * g = g * e = g$. Show that the identity element of a group G is unique.

Proof. Suppose e_1 and e_2 were both idenitites of G. Then we have $e_1 = e_1 * e_2 = e_2$, where in the first equality we use that e_2 is an identity element, and in the second we use that e_1 is. □

(b) Let e_G be the identity of G and e_H the identity of H. Show that if φ is a homomorphism then $\varphi(e_G) = e_H$

Proof. Let $g \in G$ and call $\varphi(g) = h$. We notice that:

$$h = \varphi(g) = \varphi(g *_G e_G) = \varphi(g) *_H \varphi(e_G) = h *_H \varphi(e_G).$$

If we multiply both sides (on the left) by h^{-1}, we get:

$$h^{-1} *_H h = h^{-1} *_H (h *_H \varphi(e_G)).$$

The left hand side of this equation becomes e_H, and by associativity we may move the parenteses on the righthand side so that we have:

$$e_H = (h^{-1} *_H h) *_H \varphi(e_G) = e_H *_H \varphi(e_G) = \varphi(e_G),$$

and we are done. □

(c) Let $g \in G$. Recall that an inverse to g is an element g^{-1} such that $g * g^{-1} = g^{-1} * g = 1$. Show that g^{-1} is unique.

Proof. Suppose x and y were both invereses for g. In particular $x * g = e = g * y$ so that we have:

$$x = x * e = x * (g * y) = (x * g) * y = e * y = y.$$

□

(d) Show that $\varphi(g^{-1}) = \varphi(g)^{-1}$ for all $g \in G$.

Proof. Applying part (b) we have:

$$\varphi(g^{-1}) *_H \varphi(g) = \varphi(g^{-1} *_G g) = \varphi(e_G) = e_H,$$

and

$$\varphi(g) *_H \varphi(g^{-1}) = \varphi(g *_G g^{-1}) = \varphi(e_G) = e_H.$$

Therefore $\varphi(g^{-1})$ satisfies the property of being the inverse so we are done. □

(e) Suppose $\varphi : G \to H$ is a homomorphism and is bijective. Show that the inverse $\varphi^{-1} : H \to G$ is a homomorphism as well. Such a map is called an *isomorphism*. We then call G and H *isomorphic*. This is a notion of being the *same* group in some sense.

Proof. Fix $h, h' \in H$. Since φ is surjective we know that $h = \varphi(g)$ and $h' = \varphi(g')$ for some $g, g' \in G$. Then

$$h *_H h' = \varphi(g) *_H \varphi(g') = \varphi(g *_G g').$$

In particular

$$\varphi^{-1}(h *_H h') = g *_G g' = \varphi^{-1}(h) *_G \varphi^{-1}(h'),$$

which completes the proof. □

(f) Prove that $\mathbb{F}_p^*$ and $\mathbb{Z}/(p-1)\mathbb{Z}$ are isomorphic. (You are welcome to cite work already done in past homeworks).

Proof. To be a homomorphism we must show $\log_g(ab) = \log_g(a) + \log_g(b)$ which is HW3 6(d). We must show further that it is bijective which is HW3 6(c). □