# Solutions to Discussion Worksheet 12

## 1 Diffie-Hellman

1. Take Diffie-Hellman public parameters $g = 3, p = 17$. Alice and Bob use these parameters to perform a Diffie-Hellman key exchange. You're trying to acquire their shared key (with malicious intent).

   (a) Suppose that you see messages $m_1 = 14$ and $m_2 = 12$. What is the shared key computed by Alice and Bob?

   (b) Because Diffie-Hellman is not an encryption scheme, Alice and Bob are not able to send messages with Diffie-Hellman just yet. How could they use the output of the Diffie-Hellman exchange to encrypt and decrypt messages?

   **Solution:**

   (a) You can compute the random numbers chosen by Alice and Bob via brute-force (which is why it is important to choose $p$ large enough). Via brute-force, we find that the random numbers chosen are 9 and 13. Then the shared key is $3^{9\cdot 13} = 5 \pmod{17}$.

   (b) They will need to choose an encryption scheme - having just received a shared key from the Diffie-Hellman key exchange, they are highly motivated to choose a shared-key encryption scheme. One such scheme is the one-time pad with the shared key.

2. Daphne and Julie are writing the solutions to Homework 5. Daphne has a draft of the solutions on her laptop and wants to send them confidentially to Julie. Since they have not had a private moment to choose a shared secret key together, they decide to use the Diffie-Hellman protocol to establish one over the public network.

   (a) Suppose they use the Diffie–Hellman protocol with the small prime $p = 19$ and generator $g = 2$ for $\mathbb{Z}_p^*$. Daphne chooses secret exponent $a = 7$, and Julie chooses $b = 11$. Derive the values that they send each other and the secret key that each one computes, to conclude that they both compute the same value.

   You may use a calculator for basic arithmetic (multiplication and division), but beyond that you should use the efficient algorithms that a computer would employ. Show your work (repeated squaring etc.), but you can omit the details of modular reductions. Keep numbers small by reducing modulo $p$ where appropriate, and use negative number where they help.

   **Solution:** Daphne sends $g^a \bmod p$, which is $2^7 \bmod 19$. We can calculate this using repeated squaring as follows:

$$\begin{aligned}
2^1 &\equiv 2 \pmod{19} \\
2^2 &\equiv 4 \pmod{19} \\
2^4 &\equiv 16 \equiv -3 \pmod{19} \\
2^7 &\equiv 2^4 \cdot 2^2 \cdot 2 \equiv (-3)\cdot 2 \cdot 4 \equiv -24 \equiv -5 \equiv 14 \pmod{19}\,.
\end{aligned}$$

Julie sends $g^b \bmod p$, which is $2^{11} \bmod 19$. We will again use repeated squaring, but we can reuse the work we already did to calculate what Daphne sent.

$$2^{11} \equiv 2^7 \cdot 2^4 \equiv 14 \cdot (-3) \equiv -42 \equiv -4 \equiv 15 \pmod{19} .$$

Using repeated squaring, the key that Julie computes is Daphne's public message raised to Julie's secret exponent, modulo $p$:

$$\begin{aligned}(2^7)^{11} &\equiv (-5)^{11} \equiv 25^5 \cdot (-5) \pmod{19}\\ &\equiv 6^5 \cdot (-5) \pmod{19}\\ &\equiv (6^2)^2 \cdot 6 \cdot (-5) \pmod{19}\\ &\equiv (-2)^2 \cdot 6 \cdot (-5) \pmod{19}\\ &\equiv 24 \cdot (-5) \pmod{19}\\ &\equiv -5 \cdot 5 \equiv 13 \pmod{19} .\end{aligned}$$

Similarly, the secret key that Daphne computes is Julie's public message raised to Daphne's secret exponent, modulo $p$:

$$\begin{aligned}(2^{11})^7 &\equiv (-4)^7 \pmod{19}\\ &\equiv 16^3 \cdot (-4) \pmod{19}\\ &\equiv (-3)^3 \cdot (-4) \pmod{19}\\ &\equiv 27 \cdot 4 \pmod{19}\\ &\equiv 8 \cdot 4 \equiv 13 \pmod{19} .\end{aligned}$$

Thus, both Daphne and Julie are able to compute the same shared secret key, which is 13.

(Though this was not asked for, if we wish, we can also confirm by repeated squaring that $g^{ab} \equiv 2^{7\cdot 11} \equiv 2^{77} \equiv 13 \pmod{19}$.)

(b) After realizing that the parameters from the previous part are much too small (since they are breakable by hand), Daphne and Julie decide to use the Diffie–Hellman protocol with a different, huge prime $p$ and generator $g$ of $\mathbb{Z}_p^*$. As specified by the protocol, Daphne chooses secret $a \in \mathbb{Z}_p^*$ uniformly at random, and sends $x = g^a \bmod p$ to Julie. Similarly, Julie chooses secret $b \in \mathbb{Z}_p^*$ uniformly at random, and sends $y = g^b \bmod p$ to Daphne.

Unbeknownst to them, there is a student Malcolm in the middle of their network, who is able to intercept *and undetectably replace* what they send over the network with other values of his choice. (In class we considered only an *eavesdropper* Eve who can merely read all the network traffic; this Malcolm is more powerful.)

Specifically, Malcolm chooses a secret $c \in \mathbb{Z}_p^*$ himself, and sends $z = g^c \bmod p$ to both Daphne and Julie in place of their messages to each other. That is, Daphne thinks that Malcolm's message came from Julie, and Julie thinks that Malcolm's message came from Daphne.

(a) Give an expression for the key that Daphne $k_D$ computes, and the key $k_J$ that Julie computes, in terms of the secret exponents $a, b, c$ and the public values $p, g$.

**Solution:** Following the protocol, Daphne and Julie each raise the value they received (from Malcolm) to the power of their own secret exponents, modulo $p$. So, Daphne computes $k_D = z^a \bmod p = (g^c)^a \bmod p = g^{ac} \bmod p$, and Julie computes $k_J = z^b \bmod p = (g^c)^b \bmod p = g^{bc} \bmod p$.

(b) Daphne and Julie work on the homework solutions by encrypting their messages using the keys they computed in the previous part (which they believe are the same key). Malcolm knows the encryption scheme that Daphne and Julie are using. Describe how Malcolm can decrypt all of Daphne and Julie's communications without being detected, i.e., so that they believe they are communicating confidentially, and when they decrypt the ciphertexts they receive, they get exactly the messages that were intended for them.

A valid solution should show the following:

- how Malcolm can successfully decrypt all of the ciphertexts that Daphne and Julie send to each other;
- how Malcolm can avoid detection by replacing the ciphertexts that Daphne and Julie send with ones that will yield the intended messages when they decrypt (using what they believe to be their shared key).

**Solution:** Importantly, Malcolm can efficiently compute both $k_D$ and $k_J$ since he knows $c$ and intercepted $x = g^a \bmod p$ from Daphne and $y = g^b \bmod p$ from Julie. Specifically, Malcolm calculates $k_D \equiv x^c \equiv (g^a)^c \equiv g^{ac} \pmod{p}$ and $k_J \equiv y^c \equiv (g^b)^c = g^{bc} \pmod{p}$. Thus, when Daphne sends Julie a ciphertext that encrypts a message under $k_D$, Malcolm can intercept the ciphertext, decrypt and read it, reencrypt it with $k_J$, and send the result to Julie. He can do the same process in reverse when Julie sends an encrypted message to Daphne. Daphne and Julie will not notice that anything is wrong, because they each correctly decrypt the actual intended message from the other one.

3. In class we said that Diffie–Hellman is an interactive protocol for two parties to establish a shared secret over a public channel, but it was not until RSA that a full public-key encryption scheme was known. However, it turns out that, by viewing the Diffie–Hellman protocol in the right way, we can actually use it for public-key encryption! (This perspective was discovered several years after RSA was proposed, and is widely used in practice now.)

   Let $p$ be a huge prime and $g$ be a generator of $\mathbb{Z}_p^*$, both public. To generate a public key, Alice simply performs her "side" of the DH protocol: she chooses a secret exponent $a \in \mathbb{Z}_p^* = \{1, \dots, p-1\}$ uniformly at random, and publishes $x = g^a \bmod p$ as her public key.

   (a) Suppose that somebody (Bob, or Charlie, or Dan, etc.) wants to send Alice a confidential message $m$ using her public key (and the other public data). Describe what the sender should do to accomplish this, and how Alice can compute the intended message from what is sent to her. (The algorithms run by Alice and the sender should be efficient.) Also give some brief intuition for why the method is plausibly secure against an eavesdropper. (You may assume that the DH protocol itself is secure, and that the sender has Alice's genuine public key.)

   *Hint*: Have the sender perform the other "side" of the DH protocol, and also compute and send something more. How can the message be hidden so that only Alice can compute it using her secret key?

**Solution:** The sender performs the other "side" of the DH protocol: he chooses a secret exponent $b \in \mathbb{Z}_p^* = \{1, \dots, p-1\}$ uniformly at random, and computes $y = g^b \bmod p$. Then, using Alice's public key $x$, he also computes the shared DH secret key $k = x^b \bmod p$; recall that this is $k = g^{ab} \bmod p$ because $x = g^a \bmod p$. He then encrypts the message $m$ with $k$ to get a ciphertext $c$, e.g., using $k$ as a one-time pad (or more generally, using $k$ as the key in a symmetric-key encryption scheme). Finally, he sends the ciphertext $(y, c)$ to Alice.

To decrypt the ciphertext $(y, c)$ using her secret key $a$, Alice treats $y$ just as in the DH protocol, using it and her secret exponent $a$ to compute the shared DH secret key $k = y^a \bmod p$; recall that this is the same $k = g^{ab} \bmod p$ the sender computed. She then uses $k$ to decrypt $c$ and obtain the message.

For security, the intuition is that an eavesdropper sees exactly the same public values as in the DH protocol—the fixed public values $p, g$, the public key $x = g^a \bmod p$, and the sender's public $y = g^b \bmod p$—along with the $c$ that hides $m$ under the shared DH secret key $k$. Under the DH assumption, an eavesdropper cannot compute $k$ from these public values, so the message is well hidden by $c$.

To be more careful, we should not use $k$ itself as a one-time pad, because while the eavesdropper cannot compute $k$ entirely (under the DH assumption), the eavesdropper might be able to compute some parts of $k$, i.e., $k$ might not appear entirely uniform. So, we should first "extract" some uniform-looking value from $k$, and use that as the one-time pad or other symmetric encryption key. Take EECS 388/475/575 for further details!

(b) In class we mentioned a security problem with the oversimplified RSA encryption scheme, as we presented it. Briefly describe the problem, and explain why the DH-style encryption scheme from the previous part does *not* suffer from this problem.

**Solution:** The problem is that the encryption algorithm is *deterministic*, so encrypting the same message (under the same public key) always yields the same ciphertext. This allows an eavesdropper to detect repeated messages, or even do frequency analysis if the set of actually encrypted messages is not very large.

The above DH-style encryption does not have this issue, because the encryption process is meaningfully *randomized*: every time we encrypt a (possibly repeated) message, we choose a fresh random exponent $b \in \mathbb{Z}_p^*$, compute $y = g^b \bmod p$ and $k = x^b \bmod p$, and hide the message using $k$. Each choice of $b$ is almost certain to be different (because $p$ is enormous), so each time we encrypt we get different values of $y$ and $k$, and therefore different ciphertexts (even if the messages are the same).

## 2 RSA

(a) What is $5^{376185} \bmod 376183$? (Hint: 376183 is prime.)

A. 5

B. 25
C. 125
D. 625
E. 376182

**Solution: C**. By Fermat's Little Theorem, $a^{1+k(p-1)} \equiv a \pmod{p}$ for all $z \in \mathbb{Z}_p^*$ if $p$ is prime. We solve the problem by seeing that $5^{376185} = 5^{1+1(376183-1)} \cdot 5^2$. Here, $k = 1$ and $p = 576183$, so $5^{1+1(376183-1)} \equiv 5 \pmod{376183}$. Then, $5^{376185} \equiv 5 \cdot 5^2 \equiv 125 \pmod{376183}$.

(b) Compute $5^{376376}$ mod 35 using the Euler's theorem (the generalization of Fermat's little theorem to non-prime moduli). You can use a calculator for simple arithmetic, but show all of your work for the methods you're using (repeated squaring, Fermat's little theorem, or anything else).

**Solution:** We can apply Euler's theorem here, namely that $a^{k(p-1)(q-1)+1} \equiv a \pmod{n}$. Pick $p = 7$ and $q = 5$, we have $(p-1)(q-1) = (7-1)(5-1) = 24$

$$\begin{aligned} 5^{376376} &\equiv 5^{15682\cdot 24+1+7} \pmod{35} \\ &\equiv \left(5^{15682\cdot 24+1}\right) \cdot 5^7 \pmod{35} \\ &\equiv 5 \cdot 5^7 \pmod{35} \\ &\equiv 5^8 \pmod{35} \end{aligned}$$

From here, we apply repeated squaring:

$$\begin{aligned} &5^1 \equiv 5 \pmod{35} \\ &5^2 \equiv 5^1 \cdot 5^1 \equiv 25 \pmod{35} \\ &5^4 \equiv 5^2 \cdot 5^2 \equiv 25 \cdot 25 \equiv 625 \equiv 30 \pmod{35} \\ &5^8 \equiv 5^4 \cdot 5^4 \equiv 30 \cdot 30 \equiv 900 \equiv 25 \pmod{35} \end{aligned}$$

Then, we have that $5^{376376} \equiv 5^8 \equiv 25 \pmod{35}$.

(c) (From Winter 2019) Suppose Alice and Bob want to use the RSA system with modulus $n = 35 = 5 \times 7$. Which of the following pairs of numbers are valid choices for private key $d$ and public key $e$?

A. $d = 3, e = 8$
B. $d = 7, e = 5$
C. $d = 9, e = 4$
D. $d = 5, e = 5$
E. $d = 2, e = 17$

**Solution: D.** Here, we have $p = 5$ and $q = 7$. Recall that $e \times d \equiv 1$ (mod $(p-1)(q-1)$. $(p-1)(q-1)$ in this case would be $(5-1) \times (7-1) = 24$. The only pair of keys whose product is congruent to 1 (mod 24) is $e = 5$ and $d = 5$. Note that $5 \cdot 5 = 25 \equiv 1 \pmod{24}$.

(d) Prof. Wein is sending exam questions to the EECS 376 course staff. To ensure that the staff can verify that the questions have not been altered, she uses an RSA-based signature scheme with $n = 55$ and public key $e = 27$. What would the signed message $(m, s)$ be, if $m = 52$?

**Solution:**
Given that the public modulus $n = 55$, there are only 2 possible factors for $p$ and $q$: 5 and 11. Thus, $(p-1)(q-1) = 40$ can be used to find the professor's private key. By construction, the private key is the modular inverse of e, where:

$$e \cdot d \equiv 1 \bmod 40$$

Thus, the extended Euclidean algorithm can be used as follows to find $d$, knowing $e = 27$:

$$\begin{aligned}40 &= 1 \cdot 27 + 13\\ 27 &= 2 \cdot 13 + 1\\ 13 &= 13 \cdot 1 + 0\end{aligned}$$

Isolating the remainders we can find the Bézout coefficients and thus the modular inverse of 27:

$$\begin{aligned}1 &= 27 - 2 \cdot 13\\ 13 &= 40 - 1 \cdot 27\end{aligned}$$

Combining:

$$1 = 27 - 2 \cdot (40 - 1 \cdot 27) = 3 \cdot 27 - 2 \cdot 40$$

Thus $d = 3$ is the Bézout coefficient of 27 and is the modular inverse. The signed messages, $s$, is constructed as follows:

$$s = (52)^3 \bmod 55$$

52 is equivalent to $-3$ when working modulus 55, we simplify as follows:

$$s = (52)^3 \equiv (-3)^3 \equiv (-27) \bmod 55$$

The final result:

$$s = 28 \bmod 55 = 28$$

(e) Let $n = pq$ where $p \neq q$ are two large unknown primes. Given an efficient algorithm to compute $(p-1)(q-1)$, describe an efficient algorithm to factor $n$.

**Solution:** Since $(p-1)(q-1) = pq-p-q+1 = n-p-q+1$, we have the following two equations with two unknowns:

$$\begin{cases} p+q = n-(p-1)(q-1)+1 \\ pq = n \end{cases}$$

Let $k = n-(p-1)(q-1)+1 = p+q$ (which can be computed in polynomial time, since we can calculate $(p-1)(q-1)$ in polynomial time). Note that the polynomial $f(x) = (x-p)(x-q) = x^2-(p+q)x+pq = x^2-kx+n$ has roots $p$ and $q$. Applying the quadratic formula gives:

$$p = \frac{k+\sqrt{k^2-4n}}{2}, \quad q = \frac{k-\sqrt{k^2-4n}}{2}$$

Since each of the operations listed above (exponentiation, addition, division) can be computed in polynomial time w.r.t $n$, an algorithm that computes $k$ and then carries out the above operations to derive $p$ and $q$ will also run in polynomial time.

(f) Here you will run the RSA signature scheme on some small "toy" parameters. You may use a calculator for basic arithmetic (multiplication and division), but beyond that you should use the efficient algorithms that a computer would employ (where applicable), and show your work. Keep the numbers small by reducing modulo $n$ where appropriate, and use negative numbers where they help.

Let $n = 77$ be the public modulus and $e = 7$ be the public exponent.

(a) Compute the private exponent $d$ using the extended Euclidean algorithm (provided below).

**Input:** integers $x \geq y \geq 0$, not both zero
**Output:** a triple $(g, a, b)$ of integers where $g = \gcd(x, y)$ and $ax + by = g$

```
function ExtendedEuclid($x, y$)
    if $y = 0$ then
        return $(x, 1, 0)$ // Base case: $1x + 0y = \gcd(x, y) = x$
    else
        Write $x = qy + r$ for an integer $q$, where $0 \leq r < y$
        $(g, a', b') \leftarrow$ ExtendedEuclid($y, r$)
        $a \leftarrow b'$
        $b \leftarrow a' - b'q$
        return $(g, a, b)$
```

**Solution:** We can factor the modulus as $n = 7 \cdot 11$ using brute force (trying to divide it by various primes), or just by inspection. That is, $p = 7$ and $q = 11$ (or vice versa). From here we can compute $\phi(n) = (p-1)(q-1) = (7-1)(11-1) = 60$. We now need to find a $d$ such that $7 \cdot d \equiv 1 \pmod{60}$. For this we will use the extended Euclidean algorithm on $x = 60, y = 7$:

| input | | division | | rec ans | | output | | |
|---|---|---|---|---|---|---|---|---|
| $x$ | $y$ | $q$ | $r$ | $a'$ | $b'$ | $g$ | $a$ | $b$ |
| 60 | 7 | 8 | 4 | −1 | 2 | 1 | 2 | −17 |
| 7 | 4 | 1 | 3 | 1 | −1 | 1 | −1 | 2 |
| 4 | 3 | 1 | 1 | 0 | 1 | 1 | 1 | −1 |
| 3 | 1 | 3 | 0 | 1 | 0 | 1 | 0 | 1 |

From this, we see that $2 \cdot 60 - 17 \cdot 7 = 1$, so $-17 \cdot 7 \equiv 1 \pmod{60}$. We conclude that $d = -17 \bmod 60 = 43$.

(b) Next, compute the correct signature for $m = 3$.

**Solution:** To sign $m$, we raise $m$ to the power of the private key $d$, modulo $n$. This gives the signature $s = m^d \bmod n$.
To start, we calculate powers $m^{2^i} \pmod{n}$ by repeated squaring for later fast modular exponentiation:

$$\begin{aligned}
3^0 &\equiv 1 \pmod{77}\\
3^1 &\equiv 3 \pmod{77}\\
3^2 &\equiv 9 \pmod{77}\\
3^4 &\equiv 9^2 \equiv 81 \equiv 4 \pmod{77}\\
3^8 &\equiv 4^2 \equiv 16 \pmod{77}\\
3^{16} &\equiv 16^2 \equiv 256 \equiv 25 \pmod{77}\\
3^{32} &\equiv 25^2 \equiv 625 \equiv 9 \pmod{77}\ .
\end{aligned}$$

Now we can compute

$$\begin{aligned}
m^d &\equiv 3^{43} \pmod{77}\\
&\equiv 3^{32} \cdot 3^8 \cdot 3^2 \cdot 3^1 \pmod{77}\\
&\equiv 9 \cdot 16 \cdot 9 \cdot 3 \pmod{77}\\
&\equiv (9 \cdot 9) \cdot 48 \pmod{77}\\
&\equiv 4 \cdot 48 \pmod{77}\\
&\equiv 38 \pmod{77}\ .
\end{aligned}$$

So our signature is $s = 38$.

(c) Show how one would verify, using only the public information $n, e$, that the signature $s$ you found in the previous part is correct for $m = 3$.

**Solution:** To verify that signature $s = 38$ is correct, we raise $s$ to the power of the public key $e$, modulo $n$. If this results in $m$, then we conclude that the signature is valid for this message and public key.

To start, we calculate powers $s^{2^i} \pmod{77}$:

$$\begin{aligned}
38^1 &\equiv 38 \pmod{77}\\
38^2 &\equiv 1444 \equiv 58 \equiv -19 \pmod{77}\\
38^4 &\equiv (-19)^2 \equiv 361 \equiv 53 \equiv -24 \pmod{77}\,.
\end{aligned}$$

Next, we raise $s$ to the $e$th power, modulo $n$:

$$\begin{aligned}
s^e &\equiv 38^7 \pmod{77}\\
&\equiv 38^4 \cdot 38^2 \cdot 38 \pmod{77}\\
&\equiv (-24)\cdot(-19)\cdot 38 \pmod{77}\\
&\equiv 3 \pmod{77}.
\end{aligned}$$

Because $s^e \equiv m \pmod{n}$, we conclude that the signature $s = 38$ is valid for $m$.

(d) Now suppose that we had a huge (non-"toy") RSA modulus $n$ that was, say, 1000 bits or more in size (i.e., $n \geq 2^{1000}$). Which algorithm(s) that you used in the previous parts are *feasible* for an ordinary computer to perform for this size, and which would be *infeasible* for even a state-of-the-art supercomputer to perform? Explain your reasoning.

**Solution:**

| Algorithms Used | Feasible | Part(s) Used |
|---|---|---|
| Factorization | No | a |
| Extended Euclidean | Yes | a |
| Fast Modular Exponentiation | Yes | b,c |

Brute-force factorization is not feasible for such huge numbers, because factoring $n \approx 2^{1000}$ in this way requires attempting to divide $n$ by about $\sqrt{n} \approx 2^{500}$ candidate divisors. This is (far) more than the number of elementary particles in the universe! In general, trial division on a $b$-bit number $n \approx 2^b$ takes about $2^{b/2}$ attempts, which is exponential in the size of the number (its bit length).
As we saw earlier in the term, the (extended) Euclidean algorithm is efficient: its running time is a (small) polynomial in the bit length of the input numbers. So, it can feasibly be run on an ordinary computer with 1000-bit numbers.
As we have also seen in lecture and discussion, fast modular exponentiation is efficient: the number of modular multiplications it performs is linear in the bit length of the exponent, and modular multiplication is also efficient.