Question 1

(a) Doing a signature with RSA alone on a long message would be too slow (presumably using cipher block chaining). Suppose we could do division quickly. Would it be reasonable to compute an RSA signature on a long messages by first finding what the message equals, mod n, and signing that?

A message m can be signed by a private key $<d, n>$ by computing a signature s

$$s = m^d \ mod \ n$$

When performing a signature on a message, the size of m must be smaller than n. As such, $m \ mod \ n$ will simply be m. So first computing $m \ mod \ n$ will have no effect.

We saw in lecture that an adversary is able to change m and still have it appear to be signed by the authentic party (even if m' is a message that makes no sense). This can be done as follows

$$(m^d \ mod \ n)^2 \ mod \ n = (m^2)^d \ mod \ n$$

For these reasons this approach is **not reasonable**. An alternative approach to dealing with large messages is to pad and hash the message. With this approach, regardless of message size (can be larger than n) we sign a constant size message $H(m)$. This also prevents the attack discussed above of changing the message. How exactly to pad and hash messages is defined by the Public-Key Cryptography Standard (PKCS) as discussed in the text.

(b) Suppose Alice sends a message to Bob by representing each alphabetic character as an integer between 0 and 25 (A $\rightarrow$ 0, B $\rightarrow$ 1, ... , Z $\rightarrow$ 25) and then encrypting each number separately using RSA with a large e and a large n. Describe an efficient attack against this encryption method.

Alice's public key $<e, n>$ is known by everyone including the attacker. Since the attacker knows Alice's public key, they can create a ciphertext c of a message m by calculating m^e mod n. The attacker can create a table with 26 entires, mapping each of the 26 ciphertexts to the corresponding message (0, 1, ... , 25). The attacker then can easily look at a few messages and notice the pattern of 0 $\rightarrow$ A, 1 $\rightarrow$ B, etc. as this is a very simple cryptogram.

Now the attacker can listen between Alice and Bob, intercept Alice's ciphertext for each character of the message, decode it using the table, map digits back to characters, and generate Alice's message.

This is assuming the messages are not padded before encryption.

(c) Show that $(x^c \ \textbf{mod} \ n)^d \ \textbf{mod} \ n = x^{cd} \ \textbf{mod} \ n$.

We will start with the left hand side and show that it is equivalent to the right hand side

$$(x^c \ mod \ n)^d \ mod \ n$$

Modular arithmetic produces the remainder of some integer x when divided by n. This is similar to subtracting Kn from x for some integer K. For example $15 \ mod \ 4 \ = 3$ and $15 - (3 * 4) = 3$.

With this we can rewrite the LHS of our original equation

$$(x^c - Kn)^d \ mod \ n$$

Next we will expand the expression

$$(x^{cd} - ....) \ mod \ n$$

Where we will have x^{cd} and each other term in the expanded expression will be some multiple of n as a result of the Kn. Since each of these terms is a multiple of n, they will have no result on the $mod \ n$ operation.

$$x^{cd} \ mod \ n$$

We have arrived at the RHS side of the original equation, show their equality.

Question 2

(a) Encrypting the Diffie-Hellman value with the other side's public key prevents the person-in-the-middle attack. Why is this the case, given the attacker can encrypt whatever it wants with the other side's public key?

During a Diffie-Hellman key exchange, two parties (Alice and Bob) arrive at a shared private key that no one else knows. A person-in-the-middle attack is possible by an intruder (Trudy) sitting between Alice and Bob. Trudy maintains a shared public key with Alice K_{AT} and a shared public key with K_{BT} as outlined in the textbook on page 168.

This attack can be prevented by encrypting the Diffie-Hellman values $T_A = g^{S_A} \ mod \ p$ and $T_B = g^{S_B} \ mod \ p$ with the other side's public key. This disrupts Trudy's plan as she needs T_A to compute K_{AT} and she needs T_B to compute K_{BT}.

$$K_{AT} = T_A^{S_T} \ mod \ p$$
$$K_{BT} = T_B^{S_T} \ mod \ p$$

Where S_T is a number of Trudy's choosing. Trudy can't recover T_A because it's encrypted with Bob's public key (she would need Bob's private key) and likewise T_B is encrypted with Alice's public key.

(b) Suppose the public Diffie-Hellman key of Bob is $T_B = g^{S_B}$ mod p. How does Alice send a secret message m using the Diffie-Hellman scheme to Bob? [Assume that (g, p) are known to Alice and Bob ahead of time.]

Using the known and shared g and p, Alice first chooses a random number S_A and computes her public Diffie-Hellman key $T_A = g^{S_A}$ mod p. Next using Bob's public key T_B, Alice computes the shared key $K_{AB} = T_B^{S_A}$ mod p. This key is only obtainable by Alice and Bob, so Alice uses it to encrypt m using a secret key encryption technique. Alice sends $K_{AB}\{m\}$ and T_A to Bob.

To recover the message, Bob computes the shared key himself $K_{AB} = T_A^{S_B} \bmod p$. With this, Bob can decrypt $K_{AB}\{m\}$ using the same cryptographic algorithm Alice used and recover the message m.

(c) Let there be n people in a group. Each person in the group wishes to establish a secret with every other person in the group. Let us assume that each person can send broadcast messages to reach all the other members of the group. Show an efficient Diffie-Hellman exchanges that allows each member of the group to establish a secret with every other member of the group. How many broadcast messages does your scheme use?

The public prime base g and public prime modulus p are known to all n people in the group. Each person generates their own random number S_n to use as a private key. Using this private key, they generate a public key $T_n = g^{S_n} \ mod \ p$. Each person in the group takes turns broadcasting their T_n for every other person in the group to hear. Each person computes $n-1$ shared keys (one for each other person in the group) using the other person's public key and their own private key $secret_n \ = T_n^{S_{self}} \ mod \ p$.

Now each member of the group has $n-1$ secrets, one with each other member of the group. This was accomplished by sending out n broadcast messages.

Question 3

(a) Design your own zero knowledge proof system for interactive authentication using the ideas presented in Section 6.8 of the textbook. You must present arguments to show that your scheme is secure. (You can find a long list of NP-complete problems in the book by Michael Garey and David Johnson.)

I will base my zero knowledge proof system off the vertex cover problem which is NP-Complete. The vertex cover problem is the task of given a graph $G = \{V, E\}$, does there exist a subset $U \subset V$ of size k or less such that each edge of G is incident to at least one vertex in U.

First, Alice will create a graph $G = \{V, E\}$ and select a k. Alice also knows a vertex cover $U \subset V$ for G of size k. This can be done by first creating the vertex cover U and adding a sufficient number of vertices and edges to form a complex G while maintaining the vertex cover.

The graph G will be *public* information and the vertex cover U will be *private* to Alice.

When Alice wants to prove to Bob she is indeed Alice, she will generate graphs $G_1, G_2, ..., G_{30}$ which are isomorphic to G. Alice will have the mappings from $G \to G_i$ as well as the vertex covers U_i for each G_i.

Bob sorts the G_i into two sets. For set one Bob requests $G \to G_i$. For set two Bob requests the vertex cover U_i for each G_i. Bob can confirm the isomorphic mapping in poly time. Bob can also confirm the vertex cover in poly time (iterate over the vertices in U_i and mark which edges are incident to them. At the end of the iteration all edges should be marked.)

Bob now knows he is talking to Alice and Alice has revealed no information about her secret (the vertex cover of the original graph).

As with the isomorphic example in class, an adversary Trudy will respond to Bob incorrectly on average half the time. Say Trudy generates $G_1, G_2, ..., G_{30}$ and presents them to Bob. Trudy has

two choices when constructing these graphs. She can either construct a G_i that is isomorphic to Alice's public G. Or Trudy can construct a graph G_i which she knows a vertex cover for. A graph G_i can't be both as she doesn't know Alice's secret U. But, she doesn't know how Bob will partition the graphs into two sets. Thus Trudy has a $(1/2)^{30}$ chance to correctly construct $G_1, G_2, ..., G_{30}$ and fool Bob.

(b) Transform your scheme into a zero knowledge signature scheme and also show that your signature scheme is secure.

We can transform the system from part (a) into a signature scheme for a message m. Alice wants to sign a message without another person (Bob) in the loop, she wants to sign it in general. To remove Bob we will instead use a random bit pattern to determine how to separate $G_1, G_2, ..., G_{30}$ into two groups. Using a message digest, Alice can produce a hash of m, the public graph G, the private vertex cover U, and $G_1, G_2, ..., G_{30}$.

$$MD(m, G, U, G_1, G_2, ..., G_{30}) = 10101110011010101001110 11....$$

This message digest will serve as our "random" bit pattern. At index i, if the bit is 1 we will produce the mapping $G \to G_i$ and if the bit is a 0 we will produce the vertex cover U_i. The final signature of the message m will be the 30 G_i along with the two sets of responses, one for each G_i:

$$sign(m) = \{G_1, G_2, ..., G_{30}\}, \{G \to G_1, ..., G \to G_x\}, \{U_2, ..., U_y\}$$

Recall from part (a) that if Trudy is to produce $G_1, G_2, ..., G_{30}$ she must decide if she is going to construct a graph she knows $G \to G_i$ or U_i for. But, the graphs G_i must be computed before the message digest is computed (which reveals the challenge required for each G_i).

Thus, if Alice's U is truly a secret, there is a very high probability that she will be the only person able to compute MD and produce the correct challenges relating to the bit pattern. And similar to authentication, Alice doesn't "leak" any information about her secret in the process of signing a message.

Question 4

(a) Briefly describe the properties of the wireless channel between a pair of wireless nodes that enable these nodes to extract a symmetric/secret bit sequence?

For Alice and Bob to generate a shared secret, bits are extracted from the wireless channel between them using a statistic called Received signal strength (RSS). The key extraction uses both spatial and temporal variations in the channel that both Alice and Bob can detect "identically" (similarities are found using the authors algorithms and protocols). A secret key bit pattern is extracted using a quantizer. An example RSS quantizer can set upper and lower thresholds for the signal. A signal above the upper threshold for a given frame of time would produce a 1 bit and similarly a signal under the lower threshold produces 0 bit.

There exists complications with Alice and Bob extracting identical information from a channel. Typical wireless transceivers are half duplex and they can't recieve and transmit a signal simultaneously. Additional complications such as noise, interference, and manufacturing variations will cause differences in bit streams. The authors use the Cascade protocol to agree on

shared bits. Due to the spatial element of the channel, if an adversary Eve is attempting to eavesdrop she must be physically very close to Alice or Bob.

(b) What is the similarity between the secret key extraction presented in this paper and the Diffie-Hellman cryptosystem?

The secret key extraction presented and the Diffie-Hellman cryptosystem are similar in that there is no authentication. Both systems provide a means for two parties to agree upon a shared secret key using publicly broadcasted information (RSS and the Diffie-Hellman values).

The weakness in both of these models is they are vulnerable to person-in-the-middle attacks. Diffie-Hellman copes with this by encrypting values with the other side's public key. The secret key extraction presented in the paper makes the assumption in their adversary model that Eve can't perform a person-in-the-middle attack.

(c) Does this method of secret key extraction from the wireless channel between Alice and Bob provide perfect forward secrecy? Explain briefly.

This method of secret key extraction does provide perfect forward secrecy. Each time Alice and Bob want to communicate, a new shared secret key is generated for use with the current session only. The secret extracted bits depend on the spatial variation between Alice and Bob as well as a variety of other factors that affect the channel such as noise and manufacturing variations. There are no long-term secret keys that can become comprised and used to reveal information about the current session secret key.

(d) Why is it not useful to extract secret bits from wireless channels in static environments? How can an adversary make Alice and Bob agree upon a predictable key pattern?

For Alice and Bob to establish a shared secret key, they send each other probes and measure the RSS values of these probes. In a static environment, the channel has a low reciprocity. That is, the channel curves for Alice and Bob do not follow each other as well as they do in *mobile* and *intermediate* settings (defined in the paper). The reasoning is the variations in the channel is generated by hardware imperfections and thermal effects. These variations are not shared by Alice and Bob. This makes it difficult for the extraction algorithm to extract shared bits and it takes 7-8 minutes to generate a key in a *static* environment.

In a *static* environment, it's also possible for an adversary Eve to make Alice and Bob agree upon a predictable key pattern. By doing this, Eve is aware of this predictable pattern and can generate a session key for herself. This is done by intermittently obstructing the path between the sender and receiver in a pattern. As shown in the paper, the path can be physically manipulated by Eve leaning in and out of the path of the signal. This "predictable channel" attack can be done without precise machinery, the rough movements of Eve is enough.