Questions:

1. In the TOR anonymizing overlay routing network, any one can set up a relay node.

   (a) What kind of information can a snoopy operator of a relay learn when it is the first relay on a path?
   
   When a middle relay is chosen as the first relay on a path, it can see who is using the TOR network to tunnel traffic. Note that because TOR traffic from the user to the exit relay is encrypted using a key shared by the user and exit relay, the first relay cannot see any unencrypted traffic.

   (b) What kind of information can a snoopy operator of a relay learn when it is the last relay on a path (i.e., it is an exit relay)?
   
   The exit relay can see which sites the users of TOR are connecting to. Also, if data is sent to these sites in the clear, the operator can see that too.

   (c) Suppose that an attacker deploys a large number of relay nodes, enough to comprise (not “compromise”) a significant fraction of all relay nodes in the network. Describe what the attacker might be able to learn by combining data gathered from both middle (i.e., non-exit) and exit relays.
   
   With enough relays, an attacker might get lucky and see the entire path from source to a destination in TOR, so it would know that a particular user connected to a particular site. Or it might be able to see enough of a path that by correlating timing information, it can infer which users are connecting to which sites.

   (d) Describe some risks inherent in operating an exit relay on your home network.
   
   If TOR users connect to illegal web sites from your home, you may get a visit from the police. Or if they download a copyrighted content, you may get a notice that you are being sued.

2. In Problem 3 of Assignment 2, you demonstrated that the FTP protocol sends user names and passwords to the FTP server “in the clear”, i.e., unencrypted,
which generally seems like a bad idea. One circumstance in which it might be
safe to send passwords in the clear is when a password can only be used once.  
(But FTP also sends data in the clear, so solving the password problem isn’t  
the end of the story.)

(a) One way to set up one-use-only passwords would be to store, in advance, a
list of passwords on the server and also somewhere accessible to the user,  
and then use these passwords sequentially. An observant student notices  
that if it is possible to securely store a large amount of shared private  
data on the server and on the user’s machine in advance, then another  
protocol could be employed that would solve the problem of sending pass-
words and data in the clear, and in fact provide the most secure type of  
communication possible. How would this be done?

The student is talking about a one-time pad of random bits. Communications
would be XORed with the one-time pad to encrypt and decrypt, and no bit  
in the pad would be used more than once. without access to the pad, it is
theoretically impossible to infer what bits were sent.

(b) A cryptographically secure pseudo-random number generator (CSPRNG)
is initialized with a key, and guarantees that, given the first $k$ bits that  
have been generated, there is no polynomial time algorithm for predicting  
the $(k + 1)^{st}$ bit with probability much higher than 50% without access to  
the key. How could a CSPRNG be used to set up one-use-only passwords?

A shared private key would be stored on the server and on the user’s machines.  
Matching passwords of fixed length would then be generated by both parties  
as strings of random bits using the CSPRNG.

3. In Problem 1 of Assignment 1, you helped out a fellow student named Bob who
wanted to be able to make secure connections using the SSH protocol with-
out having to type passwords all the time. Suppose Bob’s main account is on  
a university-administered Linux computer called livingontheedge.cs.duke.edu,  
and he frequently wants to open a connection to a computer owned by his  
employer, gatewaymachine.getrichquick.com. Bob’s employer has a policy of  
allowing SSH access only through the use of public-private key pairs, and not  
through password authentication. Bob thinks this is just fine, because he hates  
to type passwords anyway. So he creates a public-private key pair for himself, in  
which the private key has no passphrase. He then installs the public key on gate-
waymachine.getrichquick.com, and can log in from livingontheedge.cs.duke.edu  
using the private key without typing a passphrase.
(a) What protects Bob’s private key on `livingontheedge.cs.duke.edu`?

Bob’s account password on `livingontheedge.cs.duke.edu`, and the file permissions for the key (nobody but Bob can read).

(b) What sort of compromise is Bob’s private key vulnerable to?

Someone other than Bob is the administrator of the machine holding the key, and can become root and log in as Bob and look at the key.

(c) What type of compromise did Bob’s employer seek to limit by disallowing password authentication on `gatewaymachine.getrichquick.com`?

Bob’s employer was worried that if someone compromised `gatewaymachine.getrichquick.com`, they might be able to acquire the passwords of all the user accounts on the machine, which the users might also be using to access other machines.

4. Suppose that a content provider gives a copy of its private keys to a Web hosting service that will operate its Web site and host its home page, so that end-user browsers can connected securely to the Web servers operated by the Web hosting service using the standard SSL/TLS protocol. The content provider doesn’t fully trust the Web hosting service, so it decides that when really sensitive information must be communicated, such as the user’s password, the Web site will direct the client to connect securely to a server operated by content provider rather than the hosting service. You have been asked by the content provider to evaluate whether this approach is sound, assuming that the Web hosting service is malicious. You are immediately skeptical about whether this approach provides much security. What sort of attacks can the hosting service pull off, given that it has the private keys?

First, the hosting service can modify the site contents and serve whatever it wants, starting with the home page. One such modification would be to have the browser connect to a server belonging to the hosting service rather than to the content provider for gathering passwords. Once it has these passwords, it can connect to the content provider’s site, impersonating the user. Now it can convincingly spoof the content provider’s web site, and gather even more information. Later the hosting service can log in as the user and manipulate the account, e.g., drain all of the cash out of the account.

5. One minor vulnerability of the RSA cryptosystem is that given two signed messages $m_1$ and $m_2$, i.e., $m_1^d \pmod{n}$ and $m_2^d \pmod{n}$, where $d$ is the signer’s private key, by simply multiplying these signatures together to form $m_1^d \cdot m_2^d \pmod{n} = (m_1 \cdot m_2)^d \pmod{n}$, it is possible to forge, the signature of a
new message $m_1 \cdot m_2$, which is the product of messages $m_1$ and $m_2$, without knowing $d$. Recall that in order to use the RSA cryptosystem it is first necessary to convert a message, such as a string of text, into an integer so that modular arithmetic operations can be performed on it, such as raising it to the power $d$. In general, the product of two integers representing strings will not yield an integer representing an interesting new string. But if each of the original messages are simply integers, and these integers are directly signed, then the new message will also be an integer. For example if $m_1$ is the integer 290 and $m_2$ is the integer 2, then given $290^d \pmod{n}$ and $2^d \pmod{n}$, it is possible to forge the signature $580^d \pmod{n}$, i.e., the signature for a message that is the integer 580.

There are several straightforward ways to protect against this vulnerability. You should explain in just a few sentences how ONE of the following two approaches solves the problem.

(a) The first approach is to establish a convention that every message that is to be signed should be padded with a string of $k$ trailing random bits, where the length of the string, $k$, is fixed and known to all who obey the convention. For example, the signature for message $m_1$ would be $(m_1 \cdot 2^k + r_1)^d \pmod{n}$, where $r_1$ is a random $k$-bit integer. (Note that to verify the signature, the random string $r_1$ and the original message $m_1$ must also be known.)

Suppose $m_1$ and $m_2$ are padded using strings representing integers $r_1$ and $r_2$, so that with the padding, the messages become $m_1 \cdot 2^k + r_1$ and $m_2 \cdot 2^k + r_2$. Then the product of the two signatures is

$$(m_1 \cdot m_2 \cdot 2^{2k} + m_2 \cdot r_1 \cdot 2^k + m_1 \cdot r_2 \cdot 2^k + r_1 \cdot r_2)^d \pmod{n}.$$

But the forger needs this product to be a signature of the form

$$(m_1 \cdot m_2 \cdot 2^k + r_3)^d \pmod{n},$$

where $r_3$ is a $k$-bit string, and the product is not guaranteed to have this form (and is unlikely to have this form).

(b) The second approach is to establish a convention that rather than signing the message, a hash of the message should be signed instead, e.g., the signature of $m_1$ would be $(MD5(m_1))^d \pmod{n}$, where $m_1$ is viewed as a bit string when input to the MD5 hash function. Note that it is not the case in general that $MD5(m_1) \cdot MD5(m_2) = MD5(m_1 \cdot m_2)$. 


If the attacker simply multiplies the two signatures together, the result is
\[(MD5(m_1) \cdot MD5(m_2))^d \pmod{n}\]. But this is not the same as
\[(MD5(m_1 \cdot m_2))^d \pmod{n}\].

Note that the attacker knows \(m_1\) and \(m_2\), and hence can compute \(m_1 \cdot m_2\), and \(MD5(m_1 \cdot m_2)\), but without the private key \(d\), cannot sign \(MD5(m_1 \cdot m_2)\).

Also, even if the attacker could find a message \(m_3\) such that \(MD5(m_3) = MD5(m_1) \cdot MD5(m_2)\) (which should be very difficult since MD5 is believed to be a one-way function), the message \(m_3\) would probably be a random string of bits.