1.

1. Submitted
2. 63E717E5CC3DB94530D66612E5D90214
3. I implemented the CBC-MAC algorithm because I needed to get the tag of the default message and key (fixed length). This method is secure when authenticating fixed length messages. CBC-MAC was also the MAC tagging method I am most familiar with.
4. My implementation handles messages of any fixed size. I will demonstrate why CBC-MAC fails against arbitrary length messages.
    - Take any one block message and request its tag t
    - Output forged tag t' as CBC(m, t XOR m)
    - Above is equivalent to Fk(t XOR t XOR m)
    - = Fk(m) = t
    - So therefore we can generate forged message tags

CBC-MAC does work for messages of any fixed length, because then we can pad it to be considered 'fixed length'.

2.

1. The following protocol is susceptible to an attack called the replay attack. In this scenario, Eve has access to the secret key of Bob - called Kb. Collaborating with Mallory, Eve can access the session key used to communicate between Alice and Bob, called Kab. The idea of the replay attack is to then spoof the identity of Bob and send a message with Kb using Kab to verify to the server that it is Bob (actually Eve). Now that Eve is authenticated, she can decrypt any further messages sent to Bob.
2. Timestamped signatures can be used to prevent the above attack. Timestamped Signatures guarantee that if the signing was done in between the issue and revoking dates that the signature can be trusted. So if Alice sees that the authenticating message made to her was from a timestamp further than the last day, she will know it is invalid

3.

1. This split architecture improves security if the attacker only holds access to either the login server or the honeychecker server. If the attacker compromises the honeychecker server, the security of the system is simply reduced to the level of security without the introduction of the honeyword system, essentially reverting it back to normal. If the attacker has compromised the login system, it still cannot crack the system, since only the hashes of passwords are stored. This means that the attacker still cannot determine the real password from the list of honeywords and can do no better than guess.
2. User's are not aware of the existence of honeywords, meaning that if the server setup is sufficiently secure, it is very likely that if an attacker tries to login to the server after cracking the password they will be using the honeyword to do so. This can set off an alarm that the server has been compromised, since there would be no way for a normal user to put they honeyword in as their password.
3. paSSword5. There is many different methods of choosing honeywords (called chaffing). One such method is the idea of chaffing by tweaking of digits. So for t=1 (t positions of password string) we could generate passwords like paSSword4, paSSword9, etc. We

cannot simply do this for some t digits of the password, since in the password is the word ‘password’. Meaning it would be obvious that they were fake. We could also use a Modeling syntax where we break it down into a sequence like paSS|word|9 = W4|W4|D1. Then we could generate honeywords from a dictionary like blue|cats|1 or four|wash|2.

4. We can easily eliminate the third choice, which would be referred to as a tough nut, or a arbitrary yet very hard to crack honeyword. So we are left with the choices Blink123 and Blink128. These are both equally likely to be generated by chaffing by tail tweaking, so I will simply choose Blink128, since it has the same digits as the band Blink182, making it the most likely to be real.

4.

1. Modular Exponentiation is the basis on which RSA cryptography works. Modular Exponentiation is the problem of computing c = a^b mod n, and the fact that you can calculate this quickly (2log2b) operations. This comes from the special property that a^2b = (a^b)^2. This can make it much faster to decrypt a message m from ciphertext c. i.e. c^d = (m^e)^d = m (mod n). To do this on incredibly large primes would be too inefficient if not for this method of efficient calculation.
2. Submitted