1. Answer:
   The secret message is: When using a stream cipher, never use the key more than once

   Solution:
   1. Convert each ciphertext to binary.
   2. XOR the ciphertexts together.
   3. Since there are spaces in the text, if we get continuous 1's, we know that it is not a valid ASCII character.
   4. By applying the hint that all the texts are in [a~z, A~Z], and assuming "the" in contained in the ciphertext, we can gradually get the secret message.

2. Answer:
   0x9e1c5f70a65ac519458e7f13b33

   Solution:
   1. Turn "attack at dawn" to int, xor it with 0x09e1c5f70a65ac519458e7e53f36 to get the key.
   2. Turn "attack at dusk" to int, xor it with the key to get the one time pad encryption.

3. Answer:
   CEGH

   Solution:
   1. For 1 or 2, select the half that 26 is not in, so 1 is selected.
   2. For 5 or 6, like 1., 6 is selected.
   3. For 11 or 12, 11 is selected.
   4. For 25 or 26, 26 is selected.

4. Answer:
   C

   Solution:
   By observing the tree, we can see for every layer, we need one key, so the total number of key should be $\log_2 n$.

5. (1)

   We choose 2 keys a1 and a2, it has:

   m*a1 mod p = c = m*a2 mod p.

   Because every b in Zp* has an inverse b^-1 s.t. b*b^-1=1 mod p, and a1 and a2 belongs to Zp*, a1 must be equal to a2, which means:

   Pr[E(a1, m)=c]=Pr[E(a2, m)=c].

   This implies Perfect Secrecy.

   (2)

   The definition of Perfect Secrecy is:

   > A cipher(E, D) over (K, M, C) has perfect secrecy if
   > For m0, m1 belongs to M, (|m0|=|m1|) and for c belongs to C,
   > Pr[E(k, m0)=c]=Pr[E(k, m1)=c] Where k<-R-K

   This cipher is OTP, for it, E(k, m)=c=k XOR m,

   c XOR m = k XOR m XOR m = k, k=1.

   Therefore, it's one-to-one, so it has perfect secrecy.

   (3)

   No, it's not, because OTP's keys are generated by random number generators and would be used only once, it would have nothing to do with the plaintext.

   (4)

   No, it doesn't.

   Perfect secrecy requires that one can conclude nothing about the plaintext from the ciphertext. With a public-key system, attackers can try all 1-bit messages, all 2-bit messages and so on. With enough computational power, the result would finally come out.

6. (1)

   Let's begin with x2-x1, it equals to a(x1-x0) mod p, so a = (x2-x1)(x1-x0) mod p, where the division is mod m by the Extended Euclidean Algorithm, and b is given by b=(x1-ax0) mod p.

   Thus, we get the defining formula because we have gotten a, b, and knows p , and can then predict the rest of the sequence.

   (2)

   This implies that it is not secure for us to use congruential generator as the keystream generator for stream cipher.

(3)
We've gotten a, b, and p, it makes that we only need 2 to know the rest.

(4)
As in (1), we can observe that we need 3 to know the rest.

7. Answer:
   (D)

   Solution:
   As p, q, and r are three distinct primes, $\phi(N) = \phi(pqr) = \phi(p) * \phi(q) * \phi(r)$, and $\phi(p) = p-1$, $\phi(q) = q-1$, $\phi(r) = r-1$.
   So, $\phi(N) = (p-1)(q-1)(r-1)$.

8. Answer:
   37

   Solution:
   First calculate $\phi(N)=(3-1)(5-1)(7-1)=48$.
   We then want to find gcd(13, 48).
   Start the Euclidean Algorithm, 48=3*13+9, 13=1*9+4, 9=2*4+1.
   We then start to do backwards:
   1=9-2*4
   =9-2*(13-1*9)
   =3*9-2*13
   =3*(48-3*13)-2*13
   =3*48-11*13
   So, we get that the modular inverse of 13 mod 48 = -11,
   d= e^(-1) mod 48 = -11 mod 48 = 37.
   So, d=37.

9. Answer:
   20814804c1767293bd9f1d9cab3bc3e7ac1e37bfb15599e5f40eef805488281d

   Solution:
   1. Split the ciphertext into 2 halves.
   2. Change the 2 parts into bytes, the first half is IV.

3. Encode the given plaintext and target plaintext.
4. Pad the plaintexts to 16 bytes.
5. XOR the two plaintexts.
6. XOR the result we got in the last step with the IV.
7. Get the new CBC we want.

10. Answer:
    AC

    Solution:
    The attacker gets g^x and g^y, so they can compute B by g^x*g^x*g^y*g^y, and can get D by g^x/g^y.
    While we know that g^(xy) is difficult to compute, so A and C are also apparently difficult to compute.