1. Chosen Ciphertext Attack on RSA: To decrypt a ciphertext c without being given the decryption key, the adversary is allowed to ask for the decryption of polynomially many ciphertexts different from the ciphertext.

   The main reason for chosen ciphertext attack on RSA is because RSA function is multiplicative. Suppose $f(x) = x^e \bmod n$. Then $f(xy) = f(x)f(y) \pmod n$. Similarly, f^{-1} is multiplicative. It is because of this multiplicative property, if anyone can decrypt 1% of RSA instances then he/she can decrypt the remaining 100% by a randomized reduction.
   - A chosen-ciphertext attack on RSA
     - Eva wants to compute D(c) for some c.
     - Eva choses a random m and form $c1 = cm^e = cE(m) \bmod n$.
     - Eva asks for the decryption of c1. Suppose m1 = D(c1).
     - $D(c1) = D(cE(m)) = D(c)D(E(m)) = D(c)m \bmod n$. (multiplicative property)
     - $D(c) = D(c1)\, m^{-1} = m1m^{-1} \bmod n$

2. Bit-commitment from any one-way permutation:
   a. One can create a bit-commitment scheme from any one-way function that is injective. The scheme relies on the fact that every one-way function can be modified to possess a computationally hard-core predicate (while retaining the injective property). Let f be an injective one-way function, with h as the hard-core predicate. Then to commit to a bit a Alice picks a random input x and sends the triple $(h, f(x), a \oplus h(x))$ to Bob. To decommit, Alice simply sends x to Bob. Bob verifies by computing $f(x)$ and comparing to the committed value. This scheme is concealing because for Bob to recover a he must recover $h(x)$. Since h is a computationally hard-core predicate, recovering $h(x)$ from $f(x)$ with probability greater than one-half is as hard as inverting f. Perfect binding follows from the fact that f is injective and thus $f(x)$ has exactly one preimage.
   b. Similarly, as above, Bob commits to a bit b and picks a random input y and sends the triple $(h, f(y), b \oplus h(y))$ to Alice. To decommit, Bob simply sends y to Alice.
   c. In this way, both Alice and Bob commit to bits a and b without actually exchanging them.

3. The challenge-response identification scheme whereby the verifier sends a random challenge ciphertext for the prover to decrypt is not a good idea because an adversary may attempt to extract information by impersonating as a verifier and choosing strategic rather than random challenges. In this way the verifier/impersonated adversary may extract useful critical information about the prover's secret key using strategic challenges instead of random ones.

4. Zero-knowledge proof is a method by which one party (the prover) can prove to another party (the verifier) that they know a value *x*, without conveying any information beyond this point. *Computational* zero-knowledge, on the other hand, only requires that the simulated proof cannot be distinguished from the real proof *by a computationally bounded adversary*. I think the Zero-knowledge proof discussed in class is informational Zero-knowledge because depending on a fair coin toss of Victor, if it is 0, Peggy reveals a square root of x, hence showing that x is indeed a square; if it is 1 Peggy reveals a square root of xv, hence showing that v is a square provided x is a square which is the only information.