Closed book. Closed notes. No electronic device.

1.

AES
Authenticity
Block cipher
Collision resistant
CA
Certificate
Certificate chain
CRL
Confidentiality
DES
Diffie-Helman
Dictionary attack
Euclid's algorithm
Eucler's theorem
Existential forgery
HMAC
Integrity
KDC
MAC
Mode
CBC
CTR
OFB
Non-repudiation
OCSP
OCSP stapling
One-time pad
Pre-image resistant
PGP
PKCS
PKI
RSA
Session key
Signing
Ticket
Totient function
Verification

For each description below, give the term ($\leq$ 4 words) that *best* describes it. In most cases, the term will be in the table at left. In a few cases, it will not.

1. This ensures that the data can be read only by the intended receiver.
   Solution: Confidentiality
2. This ensures that any modification to the data is detected by the intended receiver.
   Solution: Integrity
3. This ensures that data received was sent by the specified sender.
   Solution: Authenticity
4. This ensures that a third party can verify that the data was sent by the specified sender. **Solution:** Non-repudiation
5. This kind of crypto uses different keys for encryption and decryption.
   Solution: Asymmetric crypto. (Alternative: Public-key crypto)
6. The attack model in which the attacker has access to an encryption oracle but not a decryption oracle. **Solution:** Chosen plaintext attack
7. This symmetric block cipher supports only one key size. **Solution:** DES
8. This symmetric block cipher supports multiple key sizes. **Solution:** AES
9. This defines how to use a block cipher on arbitrary-size data. **Solution:** Mode
10. This ensures that encrypting the same message more than once results in different ciphertext. **Solution:** Initialization vector (IV)
11. This mode allows the block cipher encryption function calls to be made before the data is available. **Solution:** OFB (Alternative: CTR)
12. This mode allows encryption to be done in parallel.
    Solution: OFB (Alternative: CTR)
13. This mode allows a hash function to be used for encryption of arbitrary-size data.
    Solution: OFB (Alternative: CTR)
14. The property of a hash function that makes it hard to find a message m that hashes to a given number. **Solution:** Pre-image resistant
15. This is the standard method for generating MACs from block ciphers.
    Solution: ECBC (Encrypted Cipher Block Chaining)
16. This is the standard method for generating MACs from hash functions.
    Solution: HMAC (Hashed MAC)
17. This is the set of integers in $1, \cdots, n-1$ that are relatively prime to n.
    Solution: Z_n^*
18. This is the number of integers in $1, \cdots, n-1$ that are relatively prime to n.
    Solution: $\phi(n)$ (totient of n)
19. What do we need to efficiently compute the number of integers in $1, \cdots, n-1$ that are relatively prime to n. **Solution:** Prime factors of n
20. What allows us to efficiently compute $\phi(n)$. **Solution:** Prime factors of n
21. An attack that goes through a set of candidate passwords.
    Solution: Dictionary attack
22. This means that after a session-key is forgotten by the principals that used it, no one can decrypt data encrypted with that key. **Solution:** Perfect-forward secrecy
23. A public-key infrastructure that is not hierchical. **Solution:** PGP

2. Alice has an account with a server. The server makes her change her password every few months, to which Alice just increments a number in her password, e.g., `pwd1`, `pwd2`, $\cdots$.

Why does the server not complain that the new password is very much like her old one?

Solution Because the server does not have the old password (only a hash of it).

3. Let $[e, n]$ be the RSA public key of a server. Suppose someone gives you the prime factors of n, say p and q. Can you obtain the private key $[d, n]$? If not, explain briefly. If yes, briefly give the steps.

Solution
Yes
$\phi \leftarrow (p-1)\cdot(q-1)$
$d \leftarrow e^{-1}$mod-ϕ (using Euclid's algorithm).

4. A hash function $H()$ generates a 256-bit hash. How many random messages on average would one have to hash before finding two distinct messages that hash to the same value.

Solution: Of the order of $\sqrt{2^{256}}$ $(= 2^{128})$ messages

5. Is a strong password significantly better than a weak password against an online dictionary attack. Explain briefly.

Solution: No. Because failed attempts are limited in number/frequency

6. Is a strong password significantly better than a weak password against an offline dictionary attack. Explain briefly.

Solution: Yes, especially if the attacker wants any password out of a large set. A weak password will be cracked before a strong password.

7. A server has N users and stores hashes of the users' passwords in a map P indexed by user id. Specifically, for user u, the entry $P(u)$ is $H^4(p)$, where p is u's password, H is a hash function, and $H^4(p)$ is $H(H(H(H(p))))$.

a. What would the entry be if the entries were also "salted".

   Solution: $P(u)$ is $[salt, H^4(salt\|p)]$

b. If N is 20, does salting improve security significantly? Explain briefly.

   Solution: No. It's unlikely that the same password would be used in a list of 20 users.

8. Let x be an element of Z_n and y denote its multiplicative-inverse-mod-n.

a. When does y exist?

   Solution: Iff $gcd(x, n) = 1$

b. Give the equation that x and y satisfy.

   Solution: $(x\cdot y)$ mod-n $=$ 1

9. Let $E(k, .)$ and $D(k, .)$ denote AES encryption and decryption using key k. Let message msg consist of blocks $[m_1, \cdots, m_n]$. Let $[c_0, c_1, \cdots, c_n]$ be the ciphertext resulting from encrypting msg using AES with key k in some mode.

a. Assume CBC mode. Express c_i in terms of $E()$ and $D()$ and any arguments, for $i = 1, \cdots, n$.

Solution:
$c_0 =$ random IV
$c_i = E(k,\ m_i \oplus c_{i-1})$ for $i = 1, \cdots, n$

b. Assume CTR mode. Express c_i in terms of $E()$ and $D()$ and any arguments, for $i = 1, \cdots, n$.

Solution:
$c_0 =$ random IV
$c_i = E(k,\ c_0 + i)\ \oplus\ m_1$ for $i = 1, \cdots, n$

10. For an arbitrary-size message msg, let $M(k, msg)$ denote the last block of CBC-AES encryption using key k and IV$=0$. Is $M(k, msg)$ a secure MAC. If you answer yes, explain briefly. If you answer no, give a counter example.

Solution:
No.

It is vulnerable to existential forgery.

1. Create message msg. Get its mac t $(= M(k,\ msg))$.
2. Create single-block message m.
3. Create single-block message $m \oplus t$. Get its mac t' $(= M(k,\ m \oplus t))$.

t' is a valid tag for message $msg \| m$. (The ciphertext for block m is $E(k,\ x \oplus m)$, where x is the ciphertext for the block preceding m. But x is the same as t.)

11. Why is a random pad needed for RSA encryption of a msg.

Solution:
Two reasons:

- So that repeated encryptions of the same msg yield different ciphertexts.
- So that scrambling happens even if the message is small and e is small.

12. Server B has a well-known fixed IP address and TCP port, and no other service can use that address and port. User A shares a password, say pwd, with B. A connects as follows:

1. Establish a shared key s with a standard (not authenticated) Diffie-Helman.
2. Send ["A", pwd] encrypted with key s to the server.
3. B authenticates the user if the password matches.

a. Assume an attacker that can only eavesdrop on messages. Does the above ensure that key s is securely shared between A and B. If you answer "no", give an attack. If you answer "yes", explain.

Solution:
Yes.
When A establishes a TCP connection to B's IP address and TCP port, A is assured it is talking to B (because no one else can use that address and port, and the attacker cannot tamper with messages). So after step 1, A is assured that the DH key s is shared with B.
After step 1, B is assured it has a DH key s with someone (need not be A). But after step 2, B is assured that the DH key is shared with A.

b. Repeat part a for an attacker that can eavesdrop and tamper with messages (intercept and change them).

Solution:
No.
The classic MITM (man-in-the-middle attack) works here.

- a1. A establishes a TCP connection to B.
- a2. A generates random x and sends g^xmod-p.
- a3. Attacker intercepts this and does the following:
  - ∗ generate random y
  - ∗ set DH key, say $t_A \leftarrow g^{x \cdot y}$mod-$p$
  - ∗ send g^ymod-p to B
- a4. When B receives the attacker's g^ymod-p (from a3), B generates random z, sets DH key, say $s_B \leftarrow g^{z \cdot y}$mod-$p$, and sends g^zmod-p.
- a5. Attacker intercepts this and does the following:
  - ∗ set DH key, say $t_B \leftarrow g^{z \cdot y}$mod-$p$
  - ∗ send g^ymod-p to A
- a6. When A receives the attacker's g^ymod-p (from a5), it sets DH key, say $s_A \leftarrow g^{x \cdot y}$mod-$p$.

  [At this point, A and attacker share DH key s_A, and B and attacker share DH key s_B.]
- a7. A sends ["A", pwd] encrypted with key s_A.
- a8. Attacker intercepts this and does the following:
  - ∗ decrypts it (using s_A)
  - ∗ encrypts it using s_B and sends it to B.

  Attacker now has pwd.
- a9. B receives attacker's msg (from a7), verifies pwd, and now treats DH key s_B as shared with A (whereas it is actually shared with the attacker).

13. A domain has a CA X, which is the trust anchor for the domain's users.

a. What steps does a new user, say A, take upon joining the domain.

Solution:
A generates a new public-key pair, say $[sk_A, pk_A]$.
A gets from X a certificate for A's public key, say $cert_{X,A}$.
A gets X's public key.

b. What steps are taken when a user, say A, leaves the domain before its certificate expires.

Solution: X adds the certificate's serial number to the next CRL it issues, and gives a "not valid" response to any OCSP query for the certificate.

c. What steps are taken when X's secret key is exposed.

Solution:
X generates a new public-key pair
X issues a new certificate (using the new key) for every A_i
Every A_i deletes its old public key of X
Every A_i gets (securely) the new public key of X

Note
- X issuing a CRL using the old key is useless. (Could be issued by the attacker).
- X issuing a CRL using the new key is useless. (After A_i deletes X's old pub key, the old certs won't work.)

14. The users in domain `x.com` has a CA X as trust anchor. The users in domain `y.com` has a CA Y as trust anchor. One day, `x.com` and `y.com` are acquired by `z.com`, which has a CA Z as trust anchor.

List the steps that will allow users in all three domains to talk to each other. Minimize the number of new certificates that are issued.

Solution:
1. Z issues certificates for X and Y.
2. Users in `x.com` and `y.com` get Z's public key and add Z as a trust anchor.

15. A domain's authentication is handled by KDC X.

a. What steps does a new user, say A, take upon joining the domain.

Solution: A generates a new master key and shares it with X. X adds A and the key to its users table.

b. What steps are taken when a user, say A, leaves the domain.

Solution: X deletes its entry for the A in the users table.

c. What steps are taken when X's key (used to encrypt the user keys in the users table) is exposed.

Solution: X generates a new key, and asks all users to generate new master keys.

16. A domain's authentication is handled by KDC X. Tickets can have long expiry times. Consider the following:

1. A gets a post-dated ticket T from the KDC to interact with server B.
2. B changes its master key with the KDC.
3. A presents T to B.

What is the problem here? What is a solution?

Solution:

- **Problem:** Ticket T is encrypted with B's old master key (shared with X). So when B decrypts T with its current master key, B cannot make sense of the contents and will reject T.
- **One fix:** Version numbers to master keys:
  - The KDC stores for user A the current master key and its version number.
  - B remembers its old master keys and their version numbers (until tickets issued under them have expired).
  - Each ticket contains the version number (unencrypted) of the master key used to encrypt the ticket. So B knows which key to use to decrypt the ticket.

17. Authentication protocols

This problem has independent parts. Each part describes an authentication protocol that Alice (A) initiates to send a message m to Bob (B), and then asks one or more questions.

- The first question lists some properties: **confidentiality**, **integrity**, **authenticity**, **non-repudiation**, **none** and **broken**. Circle all of the *first four* properties that hold for the message. Circle **none** if *none* of the first four properties hold. Circle **broken** if the protocol requires Alice or Bob to do something they cannot (e.g., decrypt a message without the key); in this case, ignore the other properties and any additional questions in that problem.
- The additional questions, if any, have **true/false** answers.

Unless otherwise stated, symmetric keys are strong, and the attacker can eavesdrop and tamper with messages.

The following conventions are as in the slides:

$[sk_A, pk_A]$	Alice's public-key pair. Bob has pk_A.
$[sk_B, pk_B]$	Bob's public-key pair. Alice has pk_B.
$\mathsf{E_P}(pk, x)$	public-key encryption of x with public key pk
$\mathsf{Sgn}(sk, x)$	public-key signing of x with secret key sk
$\mathsf{E}(s, x)$	symmetric-key encryption of x in CBC mode using AES with key s
$\mathsf{D}(s, x)$	symmetric-key deryption of x in CBC mode using AES with key s
$\mathsf{MAC}(s, x)$	symmetric-key MAC (ECBC) of x using key s
$\mathsf{H}(x)$	SHA-256 hash function of x
$\mathsf{HMAC}(k, x)$	HMAC of x using key k and H

17.1.

A: generate a new symmetric key s
send $[\mathsf{E_P}(pk_B, s), \mathsf{E}(s, m),$
$\mathsf{Sgn}(sk_A, \mathsf{H}(m))]$

B: receive message
extract m

---

a. Circle all that hold: **confidentiality** **integrity** **authenticity** **non-repudiation** **none** **broken**

Solution:
- confidentiality: Yes. // s is new, m is encrypted by s, s is encrypted by pk_B
- integrity: Yes. // $H(m)$ is signed by sk_A, so any change to $E(s, m)$ is detected.
- authenticity: Yes. // $H(m)$ is signed by sk_A
- non-repudiation: Yes. // $H(m)$ is signed by sk_A

b. If s comes from a password and m has structure, this is vulnerable to an offline dictionary attack: **True** **False**

Solution: True // Attacker sees $E(s, m)$

c. This has perfect forward secrecy: True False

Solution: False // If attacker gets sk_B, it can decrypt $E(s, m)$

17.2. A and B share a symmetric key s.

A: generate random c_A
$n_A \leftarrow \mathsf{E}(s, [1, c_A])$
send $[n_A]$

B: receive message
$[x, c_A] \leftarrow \mathsf{D}(s, n_A)$
if $(x \neq 1)$ "FAIL"
generate random c_B
$n_B \leftarrow \mathsf{E}(s, [c_B, c_A + 1])$
send $[n_B]$

A: receive message
$[c_B, r_A] \leftarrow \mathsf{D}(s, n_B)$
if $(r_A \neq c_A + 1)$ "FAIL"
$r_B \leftarrow \mathsf{E}(s, c_B + 1)$
session key $x \leftarrow c_A \oplus c_B$
send $[r_B, \mathsf{E}(x, m), \mathsf{MAC}(x, m)]$

B: receive message
if $(\mathsf{D}(s, r_B) \neq c_B + 1)$ "FAIL"
extract msg m

---

a. Circle all that hold: **confidentiality** **integrity** **authenticity** **non-repudiation** **none** **broken**

Solution:
- confidentiality: Yes. // m is encrypted by s, s is not exposed
- integrity: Yes. // $\mathsf{MAC}(s,\ m)$ assures B that m was sent by A
- authenticity: Yes. // $\mathsf{MAC}(s,\ m)$ assures B that m was sent by A
- non-repudiation: No. // to a third-party (even knowing s), $\mathsf{MAC}(s,\ m)$ could have been generated by B

b. If s comes from a password, this is vulnerable to a dictionary attack: **True** **False**

Solution: True // Attacker sees $E(s, [1, c_A])$ and $E(s, [c_B,\ c_A + 1])$

c. This has **perfect forward secrecy**: **True** **False**

Solution: False // If attacker gets s, it can get x and decrypt $E(x, m)$

17.3. A and B do Diffie-Helman with parameters p and g.

A: generate random x $T_A \leftarrow g^x$mod-p send $[T_A]$	
	B: receive message generate random y $T_B \leftarrow g^y$mod-p $s \leftarrow {T_A}^y$ send $[T_B]$
A: receive message $s \leftarrow {T_B}^x$ send $[E(s, m)]$	
	B: receive message extract msg m

Circle all that hold: **confidentiality** **integrity** **authenticity** **non-repudiation** **none** **broken**

Solution: none // man-in-the-middle attack (see solution to 12b)

17.4. Repeat 17.4 assuming the attacker can only eavesdrop (but not tamper).
Circle all that hold: **confidentiality** **integrity** **authenticity** **non-repudiation** **none** **broken**

Solution:

- confidentiality: Yes. // see solution to 12a
- integrity: Yes. // Attacker can only eavesdrop
- authenticity: Yes. // Attacker can only eavesdrop
- non-repudiation: No. // to a third-party, B could have generated msg m