Part I (5 Marks / Question):

Q1	**Q2**	**Q3**	**Q4**
BD	B	AC	ABCD
Q5	**Q6**	**Q7**	
AD	BD	CD	

Part II:

Q8 (4 marks):

In CTR mode, if b_1 and b_2 share the same counter c,

1. The plaintext b_1 is encrypted by $c_1 = b_1 \oplus BLOCK\_CIPHER\_ENC(k, n||c)$.
2. The plaintext b_2 is encrypted by $c_2 = b_2 \oplus BLOCK\_CIPHER\_ENC(k, n||c)$.

If b_1 is leaked, $BLOCK\_CIPHER\_ENC(k, n||c)$ can be recovered by $c_1 \oplus b_1$.

Thus, b_2 is also revealed by $c_2 \oplus BLOCK\_CIPHER\_ENC(k, n||c)$. b_1's leak leads to another block's leak.

Q9 (6 marks):

In brute-force attacks, an attacker tries all possible passwords, for a given hash from the database.

For each tried password, it calculates its hash and check if it is equal to the given hash.

If an efficient hash function is used, the attacker can also know whether its tried password is correct or not quickly, which accelerates the attack speed.

Q10 (6 marks):

Since VMK is generated from PIN, if BitLocker uses VMK to encrypt the data volume directly, BitLocker must re-encrypt the entire volume if PIN is changed or leaked, which is time-consuming. In current design, changing PIN only leads to re-encrypting FVEK, which is efficient.

Q11 (8 marks):

Step 1: Calculate the frequency of each character in ciphertext.

Step 2: Given the frequency of a character in ciphertext, find the plaintext character whose frequency is equal to it.

Step 3: Use the difference between the character in plaintext and its corresponding character in ciphertext as the key.

This attack is from the feature that Caesar cipher is a one-to-one map. One character in plaintext is map to a unique character in ciphertext. Thus, the frequency of a character in plaintext is equal to the frequency of its corresponding character in ciphertext.

Q12 (8 marks):

Salts are a part of a hash function's input together with passwords, e.g., $h(s||p)$ (s is the salt, and p is the password). If an attacker generates a rainbow table for Database A (s_a), this attacker cannot reuse this same rainbow table for Database B, because this rainbow table uses s_a as the salt part, i.e., $s_a||p$ in the last example. In Database B, all the hash inputs are $s_b||p$. The old rainbow table cannot be reused for Database B. The attacker must re-generate a new rainbow table, so the efficiency is reduced.

Q13 (8 marks):

In online attacks, the attacker must communicate with the victim server to check if a guessed password is correct or not.

In offline attacks, the attacker can dump the user database from the server, and brute force the password locally.

Online attacks can be defended by using rate-limiting to limit attacker's attempt speed, such as CAPTCHA.

Offline attacks can be defended by password access control, so that the server can refuse the dump request. Or use salts to make the attack speed slower.

Q14 (10 marks):

First bit: $ks_1 = [(0101 + 0000) \oplus 1010] \; mod \; 2 = 1$, $c_1 = p_1 \oplus ks_1 = 1 \oplus 1 = 0$

Second bit: $ks_2 = [(0101 + 0001) \oplus 1010] \; mod \; 2 = 0$, $c_2 = p_2 \oplus ks_2 = 1 \oplus 0 = 1$

Third bit: $ks_3 = [(0101 + 0010) \oplus 1010] \; mod \; 2 = 1$, $c_3 = p_3 \oplus ks_3 = 0 \oplus 1 = 1$

Fourth bit: $ks_4 = [(0101 + 0011) \oplus 1010] \; mod \; 2 = 0$, $c_4 = p_4 \oplus ks_4 = 1 \oplus 0 = 1$

The ciphertext is 0111.

Q15 (15 marks):

Plaintext: 1011.

$L_0 = 10$

$R_0 = 11$

Round 1:

Round key: $K_1 = 01$

$L_1 = R_0 = 11$

$R_1 = L_0 \oplus F(R_0, K_1) = 10 \oplus F(11, 01) = 10 \oplus 01 = 11$

Round 2:

Round key: $K_2 = 11$

$L_2 = R_1 = 11$

$R_2 = L_1 \oplus F(R_1, K_2) = 11 \oplus F(11, 11) = 11 \oplus 10 = 01$

Round 3:

Round key: $K_3 = 10$

$L_3 = R_2 = 01$

$R_3 = L_2 \oplus F(R_2, K_3) = 11 \oplus F(01, 10) = 11 \oplus 11 = 00$

Round 4:

Round key: $K_4 = 00$

$L_4 = R_3 = 00$

$R_4 = L_3 \oplus F(R_3, K_4) = 01 \oplus F(00, 00) = 01 \oplus 01 = 00$

The ciphertext is 0000.