Cryptography:

1. Generated a random key for salsa20 symmetric encryption using the **get_random_bytes** from **Crypto.random** library.

```
key = get_random_bytes(32)
Symmetric key generated:
b"\r\x1a\xb2\xc6I#\xd6\xb7=\x9aJ\xe0\xfa\xa3\x18j\xea\x84\xf4\xbd\xd4\x7f\xccX\
xd9\xcd\x181$t\x11'"
```

2. Generated Bob's public and private key by taking p,q as 512-bit prime numbers using RSA.

```
def generate_key(p, q):
    n = p * q
    phi = (p - 1) * (q - 1)
    e = 65537
    d = gmpy2.invert(e, phi)
    return (n, e), (n, d)
```

3. Alice encrypts the symmetric key using Bob's public key and returns the ciphertext:

```
def encrypt_key(key, public_key):
    n, e = public_key
    key_int = int.from_bytes(key, byteorder='big')
    c = gmpy2.powmod(key_int, e, n)
    return c
```

4. Bob decrypts the ciphertext using his private key to get the decrypted message:

```
def decrypt_key(c, private_key):
    n, d = private_key
    key_int = gmpy2.powmod(c, d, n)
    key_bytes = key_int.to_bytes(32, 'big')
    return key_bytes
```

5. Bob encrypts the message: **b"Hello, Alice!"** using salsa20.

```
message = b"Hello, Alice!"
cipher = Salsa20.new(key=key)
ciphertext = cipher.nonce + cipher.encrypt(message)
print("Part E: Ciphertext:", ciphertext)
Generated ciphertext: b'm\xcfN\x1e\xf3\xacW\x8d\x93\x1b\xa7\xab\xffl\xde\xf1]L:Va'
```

6. Alice decrypts the bob's message using generated ciphertext.

```
nonce = ciphertext[:8]
ciphertext = ciphertext[8:]
cipher = Salsa20.new(key=key, nonce=nonce)
decrypted_message = cipher.decrypt(ciphertext)
print("Part F: Decrypted Message:", decrypted_message)
print("Messages Match:", decrypted_message == message)
```