In-Lab:

1. **RSA Encryption with a 1024-bit Key:**

```
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
import binascii

# Generate RSA keys for testing
def generate_rsa_keys(bits=1024):
    key = RSA.generate(bits)
    private_key = key.export_key()
    public_key = key.publickey().export_key()
    return private_key, public_key

# RSA Encryption
def rsa_encrypt(plaintext, public_key):
    public_key = RSA.import_key(public_key)
    cipher = PKCS1_OAEP.new(public_key)
    encrypted = cipher.encrypt(plaintext.encode())
    return binascii.hexlify(encrypted).decode()
# Test RSA API
def test_rsa_api():
    # Generate RSA keys
    private_key, public_key = generate_rsa_keys()

    # Test data
    plaintext = "Hello, RSA Encryption!"
    print(f"Original Plaintext: {plaintext}")

    # Encrypt the plaintext
    encrypted_text = rsa_encrypt(plaintext, public_key)
    print(f"Encrypted Text: {encrypted_text}")

# Run the test
test_rsa_api()
```

2. **RSA Decryption with a 1024-bit Key:**

```
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
import binascii

# Generate RSA keys for testing
def generate_rsa_keys(bits=1024):
    key = RSA.generate(bits)
    private_key = key.export_key()
    public_key = key.publickey().export_key()
```

```
    return private_key, public_key



# RSA Decryption
def rsa_decrypt(ciphertext, private_key):
    private_key = RSA.import_key(private_key)
    cipher = PKCS1_OAEP.new(private_key)
    encrypted = binascii.unhexlify(ciphertext)
    decrypted = cipher.decrypt(encrypted)
    return decrypted.decode()

# Test RSA API
def test_rsa_api():
    # Generate RSA keys
    private_key, public_key = generate_rsa_keys()

    # Test data
    plaintext = "Hello, RSA Encryption!"
    print(f"Original Plaintext: {plaintext}")

    # Decrypt the ciphertext
    decrypted_text = rsa_decrypt(encrypted_text, private_key)
    print(f"Decrypted Text: {decrypted_text}")

# Run the test
test_rsa_api()
```