BABU BANARASI DAS UNIVERSITY

SCHOOL OF Computer Application

Department of Cybersecurity and forensics

Identity and Access Management.

Session 2025-26

(BCACSN13102)

PRACTICAL LAB FILE

SUBMITTED BY: -

NAME – ADARSH GUPTA

Roll No - 1240264

SECTION– BCA CS 22

SUBMITTED TO: -

Mr. Anand Kumar Gupta

INDEX

S.No.	Name of Experiments	Sign/ Remark
1.	RSA ENCRYPTION AND DECRYPTION	

RSA Encryption & Decryption — Practical Exercise Report

Title: RSA Encryption & Decryption Practical — Report

Author: [ADARSH GUPTA]

Team / Course: [BCA (CS&F) / IBM faculty practice]

Date: [2025-10-10]

Objective:

- Demonstrate RSA key generation, encryption, and decryption in a controlled lab environment.
- Explain security considerations: key sizes, padding schemes (PKCS#1 v1.5 vs OAEP), and common pitfalls.

---

1. Executive Summary

Briefly summaries the practical: what was implemented, tested, and the main conclusions (1–3 paragraphs). Example:

This exercise demonstrates generation of RSA key pairs, the encryption of plaintext using public keys, and decryption using private keys. Implementations were tested .The exercise highlights the importance of proper padding (OAEP recommended) and appropriate key sizes (2048+ bits) for modern security.

---

2. Scope & Environment

- **Scope:** Local lab only — no public key servers, no production data. All keys and test vectors are lab-generated.
- **Environment:**
    - OS: [WINDOWS]
    - Tools: CHROME :- BROWSER , DEVGLAN :- WEBSTIE FOR ENC AND DEC.

3. Authorization & Safe Practices

- Confirm instructor approval for the lab work.
- Use only sample data; do not use or publish private keys for production systems.
- Keep private keys locally and encrypted if stored.

---

4. Background (non-actionable overview)

- RSA is an asymmetric cryptosystem used for confidentiality and key exchange. Security depends on large prime generation, padding scheme, and key length. Modern guidance recommends RSA 2048 bits minimum; consider migrating to elliptic-curve algorithms for high-security needs.

---

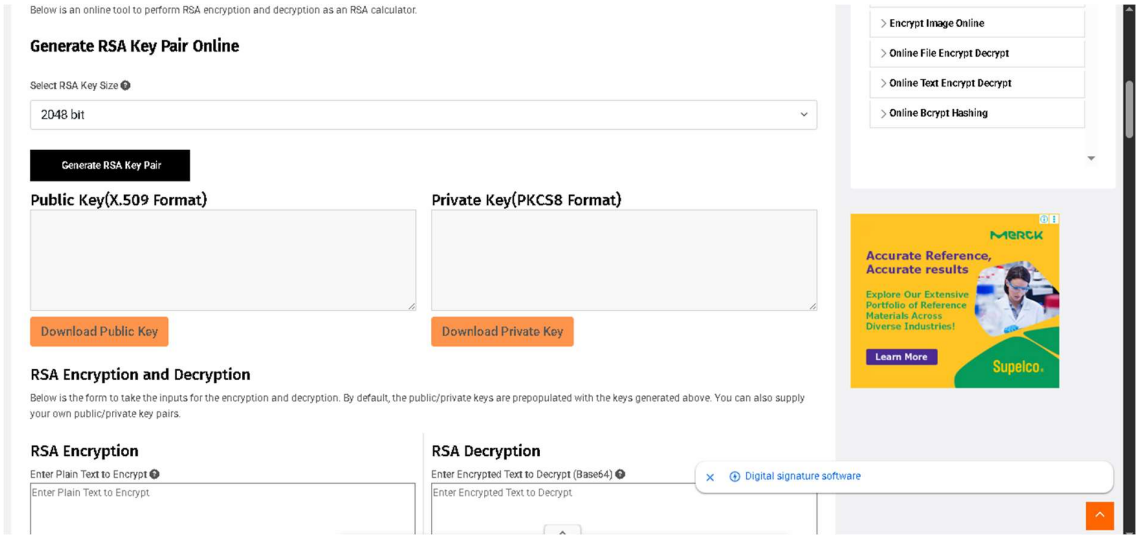
5. Methodology / Steps (High-level)

This section explains what was done without including overly prescriptive attack instructions.

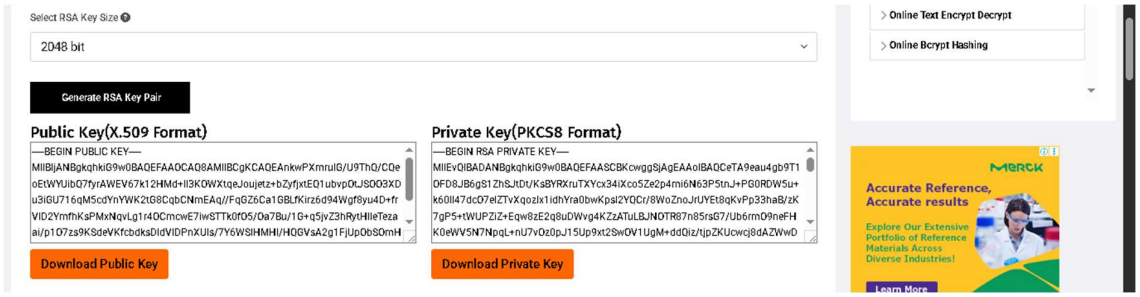
Step1) open chrome → in the search box type RSA encryption and decryption → click search

Step2) click on the first website named “devglan”

Step3) scroll down you will see “**Generate RSA Key Pair Online**”

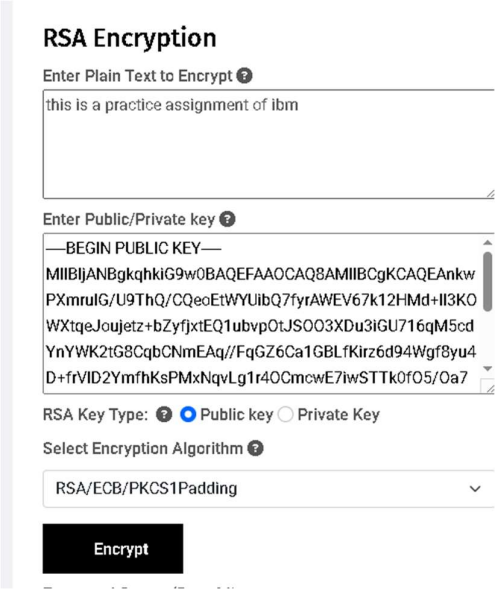


Step4) select the size of the key for e.g.: {2048 bit} → click on generate RSA key pair → both the public key and the private key will be generated.

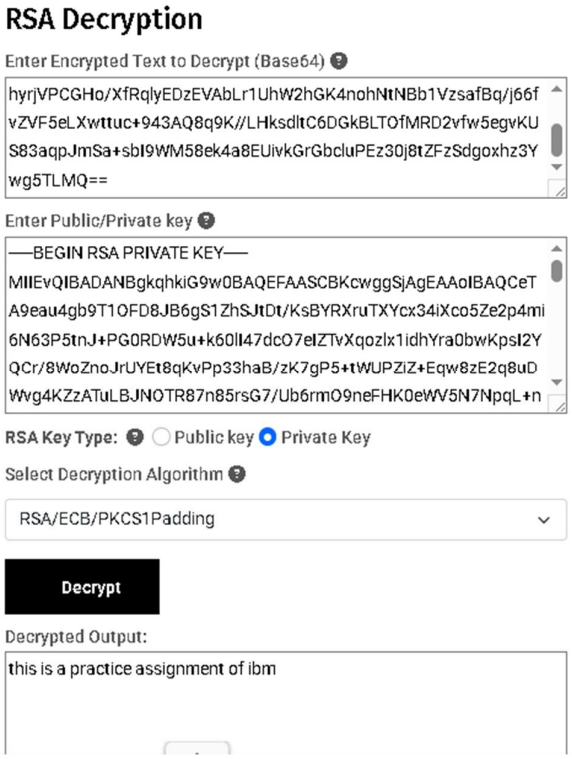


Step 5) scroll down a little bit you will see RSA encryption and decryption window this is for encryption of text.

Step 6) write any plain text you want to encrypt in the encryption window →click on encrypt.



Step7) encrypted output will be generated in the window →copy it and then past it in decryption window → click on decrypt then you will see the decrypted text.



---

6. Observations & Key Takeaways

- RSA cannot encrypt messages larger than modulus minus padding overhead — hybrid encryption (RSA encrypts symmetric key) is recommended for larger data.
- Always use OAEP (or a modern standardized padding) for RSA encryption to avoid chosen-ciphertext vulnerabilities.
- Use at least 2048-bit keys; prefer 3072-bit or stronger for long-term security depending on the threat model.
- Do not implement your own crypto primitives; use well-reviewed libraries and follow best practices.

---

7. Security Considerations & Common Pitfalls

- **Padding oracle vulnerabilities:** Avoid using vulnerable padding modes without authenticated encryption.
- **Deterministic encryption:** Avoid raw RSA encryption without padding (deterministic and insecure).

- **Key storage:** Never commit private keys to public repos. If you must include keys for demonstration, mark them clearly as test/demo keys and consider encrypting them or using ephemeral keys.
- **Hybrid crypto:** Use RSA to encrypt symmetric keys (AES-GCM) for large data.

---

8. Recommendations

- Use OAEP padding for encryption and PSS for signatures.
- Implement hybrid encryption for files/large payloads.
- Enforce secure key sizes and rotate keys periodically.
- Add comments in code showing how to move from demo to secure production practices.

---

9. Appendix — Files and Code

- rsa_examples/OpenSSL/ — commands used and sample PEM files (demo keys only).
- rsa_examples/python/ — generate_keys.py, encrypt.py, decrypt.py with usage examples.
- test vectors/ — plaintext.txt, cyphertext, sample outputs.
- Screenshots and console outputs: screenshots/

Important: If including sample private keys in the repo for demonstration, store them under demo keys/ and mark clearly. Best practice: do not publish private keys in public repos—use. Gi ignore to exclude them or make the repo private.

End of RSA practical report template.