RSA Key Pair Generation and Management

Introduction

RSA (Rivest-Shamir-Adleman) is a widely used asymmetric encryption algorithm that relies on the difficulty of factoring large prime numbers. This document outlines the steps for generating, storing, loading, and managing RSA key pairs using Python's `cryptography` library.

Prerequisites

Ensure you have the `cryptography` library installed:

pip install cryptography

Generating an RSA Key Pair

The following Python script generates an RSA key pair with a key size of 2048 bits:

```
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives import serialization

# Generate the RSA key pair
private_key = rsa.generate_private_key(
    public_exponent=65537,
    key_size=2048
)

# Extract the public key
public_key = private_key.public_key()
```

Storing the RSA Key Pair

Saving the Private Key

To store the private key securely in PEM format:

with open("private_key.pem", "wb") as f:

```
    f.write(private_key.private_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PrivateFormat.PKCS8,
        encryption_algorithm=serialization.BestAvailableEncryption(b"strongpassword")
    ))
```

Saving the Public Key

To save the public key in PEM format:

```
with open("public_key.pem", "wb") as f:
    f.write(public_key.public_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PublicFormat.SubjectPublicKeyInfo
    ))
```

Loading the RSA Key Pair

Loading the Private Key

To load the private key from a PEM file:

```
from cryptography.hazmat.primitives import serialization

with open("private_key.pem", "rb") as f:
    private_key = serialization.load_pem_private_key(
        f.read(),
        password=b"strongpassword"
    )
```

Loading the Public Key

To load the public key from a PEM file:

```
with open("public_key.pem", "rb") as f:
    public_key = serialization.load_pem_public_key(
        f.read()
    )
```

Key Management Best Practices

- **Use Strong Encryption:** Always encrypt private keys with a strong passphrase.
- **Secure Storage:** Store private keys in a secure location with limited access.

- **Key Rotation:** Regularly generate new keys and rotate old ones to enhance security.
- **Backup:** Keep secure backups of private keys in case of accidental loss.
- **Use Trusted Libraries:** Always use well-maintained cryptographic libraries like `cryptography`.

Conclusion

This document provides a comprehensive guide to generating, storing, loading, and managing RSA key pairs securely. Proper key management ensures the integrity and confidentiality of cryptographic operations, making RSA a reliable choice for secure communications.