RSA Algorithm

```python
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import rsa, padding
from cryptography.hazmat.primitives import serialization

# Generate RSA Key Pair (Private and Public Keys)
def generate_key_pair():
    private_key = rsa.generate_private_key(
        public_exponent=65537,
        key_size=2048,  # Corrected key size to a valid value (2048 instead of
2848)
    )
    public_key = private_key.public_key()
    return private_key, public_key

# Sign Document
def sign_document(private_key, document):
    signature = private_key.sign(
        document,
        padding.PSS(
            mgf=padding.MGF1(hashes.SHA256()),  # Mask generation function with
SHA-256
            salt_length=padding.PSS.MAX_LENGTH,  # Default salt length
        ),
        hashes.SHA256(),  # Hash algorithm for signing
    )
    return signature

# Verify Signature
def verify_signature(public_key, document, signature):
    try:
        public_key.verify(
            signature,
            document,
            padding.PSS(
                mgf=padding.MGF1(hashes.SHA256()),  # Mask generation function with
SHA-256
                salt_length=padding.PSS.MAX_LENGTH,  # Default salt length
            ),
            hashes.SHA256(),  # Hash algorithm for verification
        )
        return True
    except Exception as e:
        return False

def main():
    # Generate RSA key pair
```

Submitted by Nanda Krishnan V *, MSc Cyber Security*

```
    private_key, public_key = generate_key_pair()

    # Document to Sign
    document = b"This was an introduction to Digital Signature with RSA Algorithm"

    # Sign the Document
    signature = sign_document(private_key, document)

    # Verify Signature
    if verify_signature(public_key, document, signature):
        print("Signature is valid. The document is not tampered.")
    else:
        print("Signature verification failed. Document may be tampered with.")

if __name__ == "__main__":
    main()
```

Output:

```
Signature is valid. The document is not tampered.
```

Tampered Data

```
def main():
    # Generate RSA key pair
    private_key, public_key = generate_key_pair()

    # Document to Sign
    document = b"This was an introduction to Digital Signature with RSA Algorithm"
    attack = b"Attacked"
    # Sign the Document
    signature = sign_document(private_key, document)

    # Verify Signature
    if verify_signature(public_key, attack, signature):
        print("Signature is valid. The document is not tampered.")
    else:
        print("Signature verification failed. Document may be tampered with.")

if __name__ == "__main__":
    main()
```

Output

```
Signature verification failed. Document may be tampered with.
```