**Cryptography Project: CryptoBench**

**Student Name:** Seán Kitchen  
**Student Number:** C00287012  
**Submission Date:** 06/12/2024

**1. Introduction**

Cryptography is an essential aspect of securing digital communications. This project benchmarks the performance of various asymmetric cryptographic algorithms, including RSA, DSA, and ECC, in terms of keypair generation, encryption/decryption (where applicable), digital signing, and signature verification.  
The objective is to analyse and compare the performance of these algorithms across different key sizes and security levels, providing insights into their efficiency and suitability for various applications.

**2. Experimental Setup**

**2.1 Environment Details**

* **Machine Specifications**:
  + Processor: AMD Ryzen 7 5800H with Radeon Graphics
  + RAM: 16 GB
  + Operating System: Windows 11.
* **Software**:
  + Python Version: Python 3.13.1
  + Libraries Used:
    - cryptography (From cryptography.io)
    - matplotlib (From matplotlib.org)

**3. Methods**

**3.1 Algorithms and Key Sizes**

***RSA***

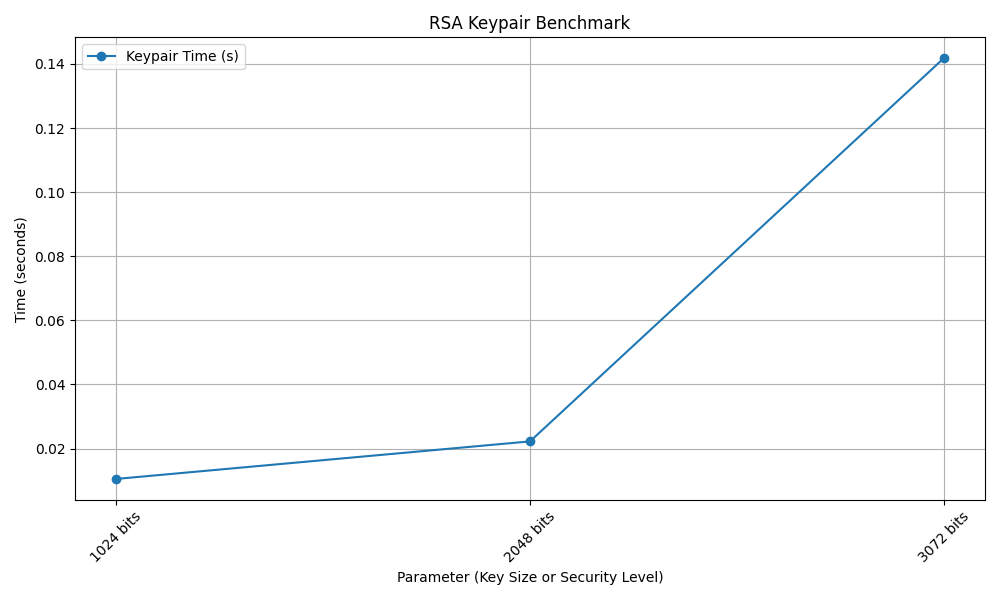
In this section, I analysed the performance of **RSA** for keypair generation, encryption/decryption, and signing/verification operations, using three key sizes: **1024 bits**, **2048 bits**, and **3072 bits**. The results are visualised with graphs generated by **Matplotlib**, providing insight into RSA’s scalability.

**1. Keypair Generation**

Keypair generation time increases with key size. The results show:

* **1024 bits**: 0.0105 seconds
* **2048 bits**: 0.0222 seconds
* **3072 bits**: 0.1417 seconds

As expected, the graph reveals a **non-linear increase** in generation time with larger key sizes, especially between 2048 bits and 3072 bits. This demonstrates the computational complexity of generating larger RSA keys.

****

**2. Encryption/Decryption**

RSA encryption and decryption times also increase with key size:

* **1024 bits**: 0.0180 seconds
* **2048 bits**: 0.0309 seconds
* **3072 bits**: 0.0547 seconds

The graph shows a **linear increase** in time with key size. Larger keys require more processing power for encryption and decryption, which is evident in the performance overhead for **3072-bit RSA**. A graph with a line going up

Description automatically generated

**3. Signing**

RSA signing and verification times remain relatively small but increase with key size:

* **1024 bits**: 0.0006 seconds
* **2048 bits**: 0.0010 seconds
* **3072 bits**: 0.0024 seconds

A graph with a line

Description automatically generated

**3. Verification**

* **1024 bits**: 0.0001 seconds
* **2048 bits**: 0.0001 seconds
* **3072 bits**: 0.0001 seconds

A graph with a line going up

Description automatically generated

**Conclusion**

RSA’s performance is directly tied to the key size. While **2048-bit RSA** offers a good balance of security and performance, **3072-bit RSA** is slower, especially for encryption and decryption operations. For large-scale applications, alternative algorithms like **ECC** may offer better performance for similar security levels.

***DSA***

**1. Keypair Generation**

The time to generate a DSA keypair increases with key size:

* **1024 bits**: 0.0807seconds
* **2048 bits**: 0.4355 seconds

A graph with a line

Description automatically generated

The graph shows a substantial increase in keypair generation time when moving from 1024 bits to 2048 bits. This reflects the increased complexity of generating larger keys.

**2. Signing**

The signing times for DSA are as follows:

* **1024 bits**: Signing: 0.0002 seconds,
* **2048 bits**: Signing: 0.0004 seconds,

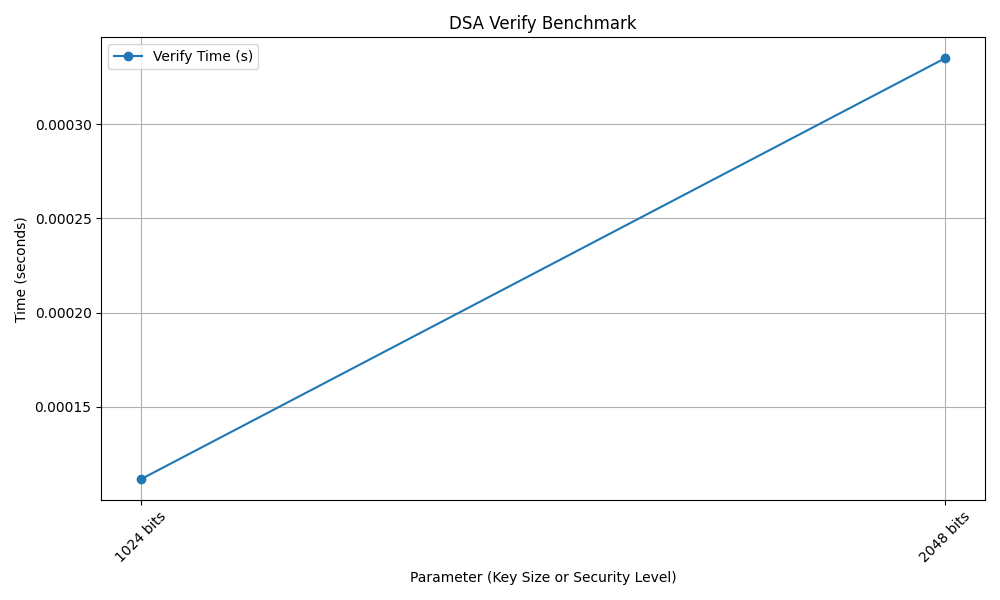
A graph with a line

Description automatically generated

**Verification**

The verification times for DSA are as follows:

* **1024 bits**: Signing: 0.0001 seconds,
* **2048 bits**: Signing: 0.0003 seconds,



The graphs demonstrate that **signing and verification times** for DSA are very efficient, even with larger keys. The increase in time is **modest** when moving to **2048 bits**.

**Conclusion**

DSA performs well for both keypair generation and signing operations. While 2048-bit DSA offers stronger security, the **keypair generation time** increases significantly compared to 1024-bit DSA. However, **signing and verification** remain efficient across both key sizes. DSA is a suitable choice for applications requiring digital signatures, particularly where key size and verification time are less critical.

***ECC***

In this section, I analysed the performance of Elliptic Curve Cryptography (ECC) for keypair generation, signing, and verification operations across five different security levels: 80-bit security, 112-bit security, 128-bit security, 192-bit security, and 256-bit security. The results are visualized with Matplotlib to highlight ECC’s efficiency.

**1. Keypair Generation**

ECC keypair generation times are efficient across all security levels:

* **80-bit security**: 0.0003 seconds
* **112-bit security**: 0.0004 seconds
* **128-bit security**: 0.0000 seconds
* **192-bit security**: 0.0008 seconds
* **256-bit security**: 0.0020 seconds

A graph with a line going up

Description automatically generated

The graph shows that **ECC keypair generation** is **extremely fast**, even for the largest security level (**256-bit**). ECC’s performance scales well with increasing security levels, with minimal impact on key generation time compared to RSA and DSA.

**2. Signing**

ECC signing times are as follows:

* **80-bit security**: Signing: 0.0133 seconds
* **112-bit security**: Signing: 0.0004 seconds
* **128-bit security**: Signing: 0.0001 seconds
* **192-bit security**: Signing: 0.009 seconds
* **256-bit security**: Signing: 0.0019 seconds

A graph with a line going up

Description automatically generated

**2. Verification**

ECC verification times are as follows:

* **80-bit security**: Verification: 0.0003
* **112-bit security**: Verification: 0.0004
* **128-bit security**: Verification: 0.0001
* **192-bit security**: Verification: 0.0007
* **256-bit security**: Verification: 0.0015

A graph with a line

Description automatically generated

The graphs show that ECC signing and verification times increase as the security level of the elliptic curve grows.

* **ECC Signing**: The signing time is quickestfor80-bit security, which requires less computational effort due to its smaller key size. As the security level increases, particularly with 256-bit security, the signing time increases, reflecting the higher computational complexity of larger curves. However, the increase is moderate, with 112-bit security offering a good balance between speed and security.
* **ECC Verification**: Similarly, **verification time** follows the same pattern. 80-bit security is the fastest, while 256-bit security is the slowest due to its larger key size and higher security. Like signing, the impact of increased security on verification time is relativelymodest, with 112-bit security again providing a good trade-off.

**3.2 Message Size**

A random 10KB message was used for encryption, decryption, signing and verification to test algorithm performance under realistic conditions.

**3.3 Metrics**

* **Keypair Generation Time**: Time taken to generate a private/public keypair.
* **Encryption/Decryption Time**: Time taken to encrypt and decrypt a message.
* **Signing Time**: Time taken to digitally sign a message.
* **Verification Time**: Time taken to verify a digital signature.

**3.4 Methodology**

* Each operation was performed 10 times for every key size or curve. The first run was excluded to account for setup overhead, and the average of the remaining runs was calculated.
* Results were recorded and visualized using Matplotlib.

### **4. Conclusion**

This project demonstrated the trade-offs in performance between RSA, DSA, and ECC. While RSA is widely used, ECC offers significant advantages in terms of computational efficiency and security per key size. DSA, being limited to signing, is suitable for specific use cases but less versatile overall.  
These results emphasize the importance of selecting the appropriate algorithm based on the use case, required security level, and system constraints.