V Semester B.Tech. (CCE)

ICT 3172: INFORMATION SECURITY

IMPLEMENTATION REPORT

Elliptic Curve Cryptography

submitted by

Kurmilla C S S Kashyap - 2109531058
Abhiram Chitta - 210953076
Patruni Subrahmanya Abhiram - 210953150
Garakipati Abhinav - 210953262

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Introduction:

This paper embarks on a comprehensive exploration of the intricate world of the Elliptic Curve Cryptography (ECC) algorithm, shedding light on its profound relevance, particularly in the context of smart card technology. In the ever-changing digital landscape, where the secure and confidential exchange of sensitive information forms the backbone of countless transactions, the role of cryptographic systems takes center stage. These cryptographic systems, akin to vigilant sentinels, tirelessly uphold the secrecy and integrity of data, serving as the last line of defense against any attempts to disown or manipulate actions.

At its core, this paper sets out on an illuminating journey to unveil the mathematical underpinnings of elliptic curves, which are the bedrock of the ECC algorithm. In an age characterized by relentless digital evolution, the ECC algorithm emerges as an unwavering guardian of information security. It is steadfast in its commitment to preserving the confidentiality and integrity of invaluable data. The ECC algorithm plays a pivotal role in the domain of computer security, and its significance continues to expand and deepen as it becomes increasingly evident in the complex tapestry of modern technology and digital protection.

The relevance of ECC extends beyond the realm of theoretical cryptography, reaching into practical applications and domains that rely on its robust security measures. Smart cards, which have become an integral part of our daily lives, secure sensitive information across a range of applications, from identity cards to electronic payment systems and access control. The ECC algorithm proves to be an ideal companion for these smart cards, offering strong security with relatively small key sizes, making it a feasible solution for resource-constrained devices like smart cards.

To effectively apply ECC to smart cards, a deep understanding of the algorithm's properties, mathematical foundations, and implementation intricacies is essential. This paper strives to provide a comprehensive insight into these aspects, offering a deeper understanding of how ECC operates within the context of smart cards and the level of security it can deliver.

Methodology:

This paper delves deep into the world of Elliptic Curve Cryptography (ECC) and its role in enhancing the security of smart card technology. In today's digital age, secure information exchange is not only crucial but also the backbone of legal and profitable transactions. Cryptographic systems serve as the protectors of data, ensuring its confidentiality, integrity, and preventing the denial of actions.

At the core of ECC lies the elegance of elliptic curves, a mathematical framework that forms the foundation of the ECC algorithm. Understanding the intricacies of these curves is essential for grasping the functionality and benefits of ECC.

This paper extends its scope by providing a detailed comparison between ECC and the widely recognized RSA algorithm. The comparison encompasses various aspects, such as key generation, digital signature creation, and verification. The aim is to evaluate ECC's efficiency and security in comparison to RSA, highlighting its strengths.

The paper references Certicom, a Canadian company with a history dating back to the early 1980s. Certicom has played a pivotal role in ECC research and promotion, advancing both theoretical and practical aspects of ECC. Their work underscores the speed of ECC implementations, a crucial advantage in the field of cryptography.

Beyond theory, this paper explores the practical implications of ECC, particularly in the context of smart card technology. Smart cards have become integral in our daily lives, securing applications like identity cards, electronic payments, and access control. ECC's ability to offer robust security with relatively compact key sizes makes it an ideal choice for securing smart cards, which often operate with limited resources.

To effectively utilize ECC within smart card technology, a deep understanding of its properties and mathematical foundations is imperative. This paper aims to provide a comprehensive view of how ECC functions within the realm of smart cards, enhancing our comprehension of its practical applications.

In summary, this paper is not just a theoretical exploration of ECC as a cryptographic algorithm. It's a journey into the heart of digital security, with a specific focus on its tangible impact on smart card technology. As the digital world continues to evolve, ECC stands as a steadfast guardian, committed to preserving data confidentiality and integrity. Its relevance is steadily growing, promising a secure and trustworthy future in our interconnected digital ecosystem. ECC's importance, particularly in the domain of securing smart cards, cannot be overstated, and this paper contributes to a deeper understanding of this critical technology.

Implementation and Results:

Elliptic Curve Cryptography (ECC) is implemented to provide secure encryption and digital signatures in a manner similar to other public-key cryptosystems. Here's how ECC works and how it aids in encryption:

1. Key Pair Generation:

- ECC relies on a pair of keys: a public key and a private key. The public key is used for encryption and signature verification, while the private key is used for decryption and signing.
- The ECC key pair is generated using a specific elliptic curve, such as SECP256R1. The private key is randomly generated, and the corresponding public key is derived from it.

2. Encryption with ECC:

- To encrypt a message using ECC, the recipient's public key is used. The sender typically needs the recipient's public key to perform encryption.
- The plaintext message is first transformed into a numerical value.
- An ECC-specific encryption algorithm is applied, which uses the recipient's public key and the numerical representation of the message.
- The result is a ciphertext that can only be decrypted using the recipient's private key.

3. Decryption with ECC:

- The recipient, who possesses the private key, can decrypt the ciphertext to recover the original plaintext message.
- The private key is used with an ECC-specific decryption algorithm to transform the ciphertext back into the original numerical value.
- This numerical value is then converted back into the plaintext message.

4. Digital Signatures with ECC:

- ECC is also used for creating digital signatures, which ensure the authenticity and integrity of a message.
- To sign a message, the sender uses their private key. The sender computes a signature for the message using an ECC-specific signature algorithm.
- The signature is appended to the message and sent to the recipient.

5. Signature Verification:

- The recipient uses the sender's public key to verify the digital signature.
- An ECC-specific verification algorithm is applied to check the signature's validity. If the signature is valid, it means the message has not been tampered with and comes from the claimed sender.

ECC_sign.py

```python
import os
from cryptography.hazmat.primitives.asymmetric import ec
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.backends import default_backend

# Generate ECC keys
private_key = ec.generate_private_key(ec.SECP256R1(), default_backend())
public_key = private_key.public_key()

# Serialize and save the private key
private_pem = private_key.private_bytes(
    encoding=serialization.Encoding.PEM,
    format=serialization.PrivateFormat.PKCS8,
    encryption_algorithm=serialization.NoEncryption()
)

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

# Serialize and save the public key
public_pem = public_key.public_bytes(
    encoding=serialization.Encoding.PEM,
    format=serialization.PublicFormat.SubjectPublicKeyInfo
)

with open("public_key.pem", "wb") as f:
    f.write(public_pem)
```

ECC_sign.py

```python
with open("private_key.pem", "rb") as f:
    private_key = serialization.load_pem_private_key(
        f.read(),
        password=None,
        backend=default_backend()
    )

# Load the public key from a file (for verifying signatures and encryption)
with open("public_key.pem", "rb") as f:
    public_key = serialization.load_pem_public_key(f.read(), backend=default_backend())

# Sign a message with the private key
1 usage
def sign_message(private_key, message):
    signature = private_key.sign(
        message,
        ec.ECDSA(hashes.SHA256())
    )
    return signature
```

ECC_sign.py

```python
1 usage
def verify_signature(public_key, message, signature):
    try:
        public_key.verify(
            signature,
            message,
            ec.ECDSA(hashes.SHA256())
        )
        return True
    except Exception:
        return False

# Example usage
user_input = input("Enter a message: ")
message = user_input.encode("utf-8")

# Sign the message
signature = sign_message(private_key, message)

# Verify the signature
is_verified = verify_signature(public_key, message, signature)

if is_verified:
    print("Signature is verified: Message is authentic.")
else:
    print("Signature verification failed: Message may be tampered with.")

# Print the original message, signature, and verification result
print(f"Original Message: {message.decode()}")
print(f"Digital Signature: {signature}")
```

Execution output:

Run ECC_sign

```
C:\Python311\python.exe "C:\Users\Delll\OneDrive - Manipal Academy of Higher Education\Desktop\IS FISAC\ECC_sign.py"
Enter a message: Kashyap Abhiram Abhiram Abhinav
Signature is verified: Message is authentic.
Original Message: Kashyap Abhiram Abhiram Abhinav
Digital Signature: b"0E\x02 2\xa48\xfa\xef\x10\x94L _w\xa91'b\x0b\xa3\x94p\xc1\x95\xb7k`\xcf\xf2n+\x01P\xbe\xa0\x02!\x00\xb1H\x11Ul\xe2\xdcS`\xe6\xb6\xb5o\xba\xba\x13\x981\x13\
Process finished with exit code 0
```

Run ECC_sign

```
C:\Python311\python.exe "C:\Users\Delll\OneDrive - Manipal Academy of Higher Education\Desktop\IS FISAC\ECC_sign.py"
Enter a message: 210953058 210952076 210953150 210953262
Signature is verified: Message is authentic.
Original Message: 210953058 210952076 210953150 210953262
Digital Signature: b'0E\x02 P\xfbD\xcb=\x89\xf8j\x0e\xfai>h\x08\x9fU\x8d@\xa0\xb1\xaa\x95\x9a\xd6\x93z\\\xd3\xf5w\xd2t\x02!\x00\xb8\x03J\x04<\xd1N2\xd3\x97M\xb8\x8b\x88D\xf4\x7
Process finished with exit code 0
```

Elliptic Curve Cryptography (ECC) Advantages and Uses:

- **Enhanced Security:** ECC employs elliptic curves over finite fields to fortify communication systems against various attacks.
- **Efficiency and Speed:** ECC's compact key sizes reduce computational load, resulting in faster encryption and decryption, making it ideal for resource-limited environments.
- **Key Steps in ECC Implementation:** Key phases involve curve and field selection, private key generation, public key computation, and secure data exchange via encryption and decryption.
- **Mathematical Operations:** ECC relies on advanced mathematical operations like point addition and scalar multiplication on elliptic curves to ensure security, notably addressing the Elliptic Curve Discrete Logarithm Problem (ECDLP).
- **Efficient and Secure:** ECC's computational efficiency and resistance to attacks establish it as a premier choice for modern cryptographic applications, combining robust security with efficient performance.

Conclusion:

In conclusion, Elliptic Curve Cryptography (ECC) offers notable advantages, including faster performance and reduced memory usage compared to RSA. This makes ECC particularly suitable for resource-constrained environments like smart cards. While ECC adoption has been gradual, there's growing industry consensus, including implicit support from RSA Security, that ECC is the future of cryptography. The key size difference between ECC and RSA is expanding exponentially, favoring ECC. ECC benefits from a solid research foundation, with its cryptographic applications gaining recognition more recently. In contrast, RSA has a legacy of extensive research and numerous academic theses. It's worth noting that elliptic curves in cryptography were initially explored while seeking new approaches to challenge the RSA system, highlighting the dynamic nature of cryptographic innovation.

References:

[1] Kapoor, V., Abraham, V.S. and Singh, R., 2008. “Elliptic curve cryptography.” *Ubiquity*, *2008*(May), pp.1-8.