National College of Ireland

Title

MSc Research Project
Cyber Security

Sunaina Amin
Student ID: X23322683

School of Computing
National College of Ireland

Supervisor: Liam McCabe

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sunaina Amin
Student ID:	X23322683
Programme:	Master of Science in Cyber Security **Year:** 2024_2025
Module:	Research Program
Supervisor:	Liam McCabe
Submission Due Date:	11th August 2025
Project Title:	Secure Smart Home Using ECC

Word Count: 8704	**Page Count**: 27

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Sunaina Amin

Date: 11th August 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Table of Contents

Table of Acronyms

Acronyms	Full Form
AES	Advanced Encryption System
AVISPA	Automated Validation of Internet Security Protocols and Applications
CPU	Central Processing Unit
DNSSEC	Domain Name System Security Extensions
EAP_SCHNORR	Extensible Authentication Protocol – Schnorr
ECC	Elliptic Curve Cryptography
ECC_224	Elliptic Curve Cryptography with 224-bit key
ECDSA	Elliptic Curve Digital Signature Algorithm
GUI	Graphical User Interface
ICAC	International Conference on Automations and Computing
ICT	Information and Communication Technology
IDE	Integrated Development Environment
IECC_SAIN	Innovative ECC Based Secure Authentication in IoT Network
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
INVOTEK	Jurnal Inovasi Vokasional dan Teknologi
IT	Information Technology
JSON	JavaScript Object Notation
LAID	Lightweight Authentication for IoT Devices
MITM	Man-in-the-Middle
MQTT	Message Queuing Telemetry Transport
NXP	NXP Semiconductors
PEM	Privacy Enhanced Mail
PKCS	Public-Key Cryptography Standards
RSA	Rivest–Shamir–Adleman
SHA	Secure Hash Algorithm
VS	Visual Studio
WAN	Wide Area Network

Abstract

A smart home is embracing the use of the Internet of Things (IoT) at a rapid rate across the world. In the world, the number of connected IoT devices is expected to reach 27 billion by 2025 (IoT Analytics, 2024) and 29.4 billion by 2030 (Statista, 2024). There is also a rapid growth in the regional markets; in Thailand alone, the smart home users are projected to have more than 5.6 million users by the year 2029 (Statista, 2024b). The growth brings about major cyber threats, especially in device authentication and resistance against replay and man-in-the-middle attacks. Traditional cryptography, including Rivest-Shamir-Adleman (RSA), is becoming less applicable to IoT resources due to its high key size, high computation, and energy usage.

Elliptic Curve Cryptography (ECC) is a more effective alternative. According to elliptic curve mathematics, ECC has the same security level as RSA using significantly smaller key sizes (e.g., a 256-bit ECC key is as secure as a 3072-bit RSA key). The efficiency saves on the computation cost, minimizes power consumption, and creates shorter signatures, which makes ECC especially useful in a smart home where devices are running on limited resources.

This study invents and tests an ECC-based mutual authentication protocol that has been coded in Python in Visual Studio Code, with MQTT communications as the publisher and subscriber device to recreate a smart home IoT setting. As a comparison, RSA had also been implemented under the same conditions. The assessment was based on authentication time, CPU and memory consumption, signature size, throughput, and estimated energy consumption. The findings consistently show that ECC outperforms RSA in terms of faster authentication, lower memory costs, and reduced energy expenditure.

The paper finds that ECC provides a scalable, energy-efficient implementation of RSA in securing IoT smart homes by providing increased device authentication and performance in limited environments. Additional work in this field can involve hardware testing at the platform level, support on platforms like the Raspberry Pi or ESP32, support of lightweight protocols such as CoAP, and investigating hybrid ECC post-quantum strategies. Key Words: Elliptic Curve Cryptography, RSA, Smart Homes, Mutual Authentication, Secure, Internet of Things.

Keywords: Elliptic Curve Cryptography, Smart Homes, Internet of Things, Lightweight Authentication.

1. Introduction

The rising IoT framework in smart homes offers multiple advantages in automation, energy efficiency, and convenience to the users. Through the integration of smart devices like smart locks, lights, security cameras, and thermostats, smart homes can execute autonomous tasks relying on inter-device connectivity and user-managed commands. Smart home systems foster omnipresent and intelligent services through Information Communication Technology (ICT), thereby enhancing homes' quality of life (Ali, 2017).

However, the same characteristics that enable smart homes to function intelligently, always-on connectivity, dynamic communications, and heterogeneous device architecture, also create profound cybersecurity concerns. Smart devices generally employ wireless communication without centralized security control, making them vulnerable to a variety of attack vectors, including man-in-the-middle (MITM) attacks, unauthorized access, replay attacks, and impersonation. A vast majority of these vulnerabilities result from poor or a lack of authentication

protocols between devices (Zahan et al., 2020). With the growth of IoT and its increasing integration into daily life, secure mutual Authentication becomes increasingly important not just to protect privacy but also to maintain operational integrity. Authentication in this system needs to be lightweight, scalable, and power-efficient demands that conventional cryptographic protocols such as Rivest–Shamir–Adleman (RSA) do not satisfy. Though RSA has been the mainstay of public-key cryptography for decades, its performance takes a hit in setups where devices are limited by processing capacity, memory, and battery power. RSA needs large key sizes (20 or higher) to ensure reasonable levels of security, which also translates to longer computation times and greater energy consumption (Lauter, 2004). In resource-constrained environments such as smart homes, this is a severe limitation. In contrast, according to (Lauter ,2004), ECC(ECC) is a more appropriate alternative for IoT because it can achieve the same level of security with much smaller key sizes. For instance, a 224-bit ECC key provides similar security to a 2048-bit RSA key but with quicker performance and less computational overhead (Shabaan et al., 2023; Adeniyi et al., 2024). The mathematical basis of ECC enables compact key generation, signing, and verification procedures, making it a prime candidate for low-power, low-latency authentication schemes.

Several recent research works have proved ECC's superiority in IoT and smart home security. Khalique et al. (2025) suggested the LAID algorithm, an ECC-based lightweight authentication protocol for smart cities that reported 22% gain in energy and computational efficiency over RSA. In the same way, Ruswiansari et al. (2024) compared ECC and RSA in a real-world MQTT-based IoT setup on Raspberry Pi and proved that ECC provided quicker authentication times along with improved resilience to sniffing and replay attacks. Yang et al. (2023) implemented ECC in an Industrial IoT environment and validated its strength through the incorporation of token-based trust mechanisms to counter impersonation attacks. Together, these research works reinforce the increasing academic and industrial popularity of ECC in secure IoT communication.

Building on this background work, this research presents a lightweight ECC-based mutual authentication protocol specifically designed for smart home systems. The uniqueness of this research lies in its minimalistic and replicable implementation: instead of using high-overhead tools such as Raspberry Pi, Java Servlets, or CrypTool (as in previous studies), this project implements and tests the complete protocol in Python using Visual Studio Code. By avoiding the use of external simulation software, this research illustrates that mutual authentication with high security can be achieved using open and ubiquitous development environments.

The protocol framework is based on MQTT communication, which is a widely adopted protocol in smart home networks thanks to its low overhead and publish/subscribe paradigm. A publisher (user or controller) issues digitally signed commands to a subscriber (smart device), and the latter authenticates the signature before performing the command. The protocol is realized with both RSA and ECC for direct comparison. Key performance indicators authentication time, CPU and memory utilization, and energy estimation, are measured under controlled conditions. The protocol also combines timestamp-based replay protection, making every authentication message time-sensitive and immune to duplication attacks.

1.1. Research Question:

How does ECC over RSA compare regarding security, computation, and power

consumption in supporting the mutual authentication in the IoT scaling of smart homes?

ECC is superior to RSA on every important aspect as far as mutual authentication in IoT goes. A theoretical study by Lauter (2004) demonstrates that ECC has a security-to-key-size and performance benefit; an experimental test witnesses such an advantage in Zahan et al**,** (2020) in an IoT setting. Collectively, they indicate that ECC is strictly better in terms of providing high security, computational performance, and energy awareness than the rest is which makes it the best cryptographic option to use in secure smart homes. The experimental analysis indicates how ECC is always better in terms of performance than RSA in the trial smart home scenario using MQTT. ECC attained reduced authentication time, less CPU/memory requirement, and consumes less energy, because it utilizes smaller key sizes and has less computational overhead. These findings are consistent with others in the literature that represent confidence in using ECC in resource-limited IoT applications where efficiency and security are both important factors. Although RSA can still be effective in older systems, the results verify ECC as a better option in lightweight networks and safe transmission in a smart home structure.

1.2. Mutual authentication

can be found explicitly as shown in the IoT-Devices Authentication phase of the proposed and ECC-based ECDSA scheme in the article referred to as Shaaban et al. (2023). They clarify that after loading the initialization step, “all distributed Fog nodes will use the ECDSA algorithms to offer keys and authenticate their IoT-devices accordingly and also to issue their IoT-devices with signed authentication requests which they send back to Fog node to be authenticated there.” The two-way verification is a kind of root that leaves no doubt in the legitimacy of each to the other before any communication is done. Shaaban et al. (2023) confirmed that their ECC-based method supports secure mutual authentication between IoT devices and fog nodes via a signed message exchange confirmed by both parties, and as compared to RSA, it needs less time and energy on computation, yet it is resistant to replay and impersonation.

1.3. The contributions of this research are fourfold

Design and development of an ECC-optimized mutual authentication protocol for smart home IoT environments. Benchmarking comparison between ECC and RSA on the same testing conditions, with MQTT communication and Python scripting. Analysis of performance measures such as latency, CPU and memory utilization, and replay resistance based on empirical results. Verification of minimal toolchain deployment, showing that robust IoT security measures can be constructed without the use of proprietary hardware or tools.

Moreover, this research is part of the industry's wider shift towards post-quantum cryptography and energy-efficient IoT solutions. As the need for secure, low-latency IoT communication continues to increase, lightweight cryptographic frameworks like ECC will increasingly form the backbone of the next generation of smart infrastructure. By presenting empirical evidence and an implementation model that has been tested, this project plays a part in that shift and provides a workable reference model for subsequent research in the field. In summary, this project closes the gap between theoretical cryptographic research and actual smart home deployment. It suggests a

toolchain-independent, resource-sparse protocol for secure device communication in IoT networks and verifies ECC as a practical, scalable substitute for RSA in mutual authentication protocols. The accelerated development with Visual Studio Code demonstrates that successful security research can be done with limited resources, allowing this research to be accessible, reproducible, and usable for both academia and industry.

2. Related Work

The Internet of Things (IoT) has ushered in a paradigm shift in the digital control and automation of smart environments. With this innovation, however, has come the problem of numerous security vulnerabilities, most prominently mutual authentication among devices. Traditional public key algorithms such as RSA, while suitable in conventional computing environments, are ill-suited to the constraints of IoT by way of their enormous key sizes and wide computational overhead (Lauter, 2004; Zehan et al., 2020). ECC(ECC) has emerged, though, as a lightweight and power-efficient option that provides secure parity with much smaller key sizes and is therefore highly suitable for smart home deployments.

Early research shows that ECC can deliver the equivalent cryptographic strength of RSA with very short key lengths, such as ECC-224, compared to RSA-2048, leading to better speed and less memory in wireless settings by Lauter (2004).

Zahan et al. (2020) also confirmed this by designing ECC-based digital signature protocols for smart home systems and demonstrating lower latency and better energy efficiency during real-time device authentication. Their research applied ECC alongside MQTT to simulate home automation activities, providing evidence that ECC is more suitable for systems with constrained energy and processing resources.

Following on from these underlying observations, Ruswiansari et al. (2024) took a comparative implementation approach with Raspberry Pi boards, comparing mutual authentication times between ECC and RSA within an MQTT context. ECC showed average mutual authentication times of 153 ms compared to RSA's 216 ms while utilizing fewer CPU and power resources. ECC was also less vulnerable to replay and sniffing attacks, especially if combined with symmetric encryption protocols such as AES. Their results provided quantifiable proof of ECC's performance edge in time-sensitive, resource-constrained environments.

Yusoff et al. (2024) suggested a more robust solution by integrating ECC into the MQTT protocol using Java Servlets and Raspberry Pi hardware. Energy consumption was reduced by 77.04%, while confidentiality increased by 87.68% in comparison to RSA-based models, they reported. Their use of NetBeans IDE, Mosquitto Broker, and dynamic generation of session keys contributed to the increased viability of ECC in practical smart home settings.

Further enhancements were offered by Shaaban et al. (2023), who proposed an efficient ECC-based authentication protocol for fog computing environments. Although not specifically working with smart homes, their protocol succeeded in minimizing energy usage and computational overhead, to the advantage of the generalizability of ECC for lightweight, secure IoT networks.

Likewise, Nyangaresi (2022) suggested an anonymous authentication protocol using ECC for resource-limited devices, focusing on both scalability and privacy.

In contrast, RSA remains applicable in traditional and legacy IT networks, but not quite potent when applied in resource-constrained environments such as smart houses. It was mentioned that RSA has a high key size and thus takes up a lot of computational overhead, latency, and high energy consumption, which is not suitable in the environment of MQTT as used in constrained IoT devices by the author Verma (2022). What is more, MQTT itself is susceptible to man-in-the-middle (MITM), replay, and sniffing attacks unless augmented by lightweight cryptography techniques like ECC. Cryptographically robust ECC curves such as SECP256r1 and Curve25519 have been identified as well-suited to these highly constrained operating environments.

Adeniyi et al. (2024) also pointed out the wide use of ECC in IoT and Wireless Sensor Networks, showing that it works well in multi-layer networks that need strong security without losing its reliability and credibility. they also give details about how RSA uses more memory and is slower in the handshake process. Which makes it vulnerable to use in real real-world environment.

Choubisa and Jajal (2025) suggested a hybrid token-based and ECC authentication protocol that mitigated the possible vulnerabilities of ECC to session hijacking and replay attacks.

The system combined session tokens and ECC public-private key mechanisms, including solutions that did not require additional energy. The testbed used Python, Mininet, and Raspberry Pi to simulate authentication with varying latency and battery settings, with favorable outcomes for improved security and scalability.

For practical implementation and simulation, several tools have been created to evaluate and benchmark these cryptographic protocols effectively. For instance, Yusoff et al. (2024) used Java Servlets and Mosquitto on Ubuntu servers for ECC session key negotiation implementations. The same tools are being used in the current practicum project for simulating ECC authentication with realist simulation scenarios in terms of latency, throughput, and CPU usage metrics.

Moreover, this Research also presents a lightweight and compact ECC-based authentication system developed in pure Python with the use of Visual Studio Code. While the initial proposal entertained the use of simulation software such as CrypTool 2, Cisco Packet Tracer, and Raspberry Pi to emulate network and hardware constraints, the actual implementation intentionally goes the other way around with a lighter and more common environment. It thus offers real-time validation of ECC and RSA through MQTT-based communication and terminal prints, respectively, for efficient comparison of performance in terms of authentication time, memory used, and CPU utilization. By eliminating the need for advanced or resource-intensive software tools, the research demonstrates that meaningful and academically significant results can be achieved with minimal and common resources, making the study not only reproducible but also easily applicable for researchers and developers with limited infrastructure. As opposed to previous research, which either concentrated on theoretical frameworks or narrow-use cases (for example, industrial IoT or fog computing), this practicum is specifically crafted for use in smart and in-depth review

classified ECC-based applications in smart devices, health, grid, and home networks, and concluded that ECC is still the most reliable cryptographic solution for energy- and latency-constrained systems. Liu et al. (2023) went a step further to incorporate ECC in mobile computing environments detailed work on two-factor authentication system security. Their system provided double-layer security, ECC, and biometric authentication, showcasing the adaptability of ECC and its advancement for the future. ECC has been a commonly accepted industry standard for secure authentication in IoT, with the support of industry giants and researchers. Though RSA was the ordered standard previously, it is being phased out in applications where energy efficiency and speed are paramount. This study takes it a step further by implementing a mutual authentication protocol for a smart home using ECC. It offers theoretical research as well as device-level implementation and a demo. It also shows performance analysis, making it a good foundation and clear direction for real-world development. The need for lightweight and secure cryptographic primitives for IoT environments has encouraged detailed research into ECC(ECC). There is a need for efficient authentication protocols in smart home settings that balance security, performance, and overhead, given the limited computational and energy resources of IoT devices.

2.1. ECC in Resource-Constrained Device

Alghamdi (2025) proposes a highly lightweight ECC-based approach for IoT systems with constrained resources. Their approach utilizes ECC's advantage of providing robust security with shorter key sizes, enabling quicker real-time authentication with less processing power and memory consumption. This is well matched with smart home systems, which share the same limitations on processing power and energy usage (Alghamdi, 2025).

2.2. Token-Based Improvements to ECC

Chubosia Mukesh et al. (2025) propose a hybrid system that uses both token-based authentication and ECC to solve session management problems and replay attack protection issues. In this system, OAuth-like tokens are used along with ECC key exchange, enabling reduced overhead and support for multi-device smart homes. While this system adds token complexity, it shows the need for more complex ECC systems with better session management features (Mukesh et al., 2025).

2.3. MQTT Protocol-based ECC Implementation

Zainal et al. (2022) examined ECC utilization within Raspberry Pi-based smart home systems utilizing MQTT and Java Servlets. While their instrumentation differs from this research's basic Python-based infrastructure, their findings indicate that ECC enhances confidentiality and, in certain instances, can minimize energy consumption and delay (latency) by over 50%. These findings corroborate ECC as a suitable option for low-overhead home automation systems (Zainal et al., 2022).

2.4. Comparative Study of RSA and ECC in IoT Systems

A survey of ECC(ECC) implementations (Sukumar 2024) evaluated it and asserted that the security and performance of ECC are better than RSA, especially considering restricted IoT resources. In the analysis, ECC proved its superiority in authentication speed, low cost of

communication, and high cryptographic robustness, which makes it ideally edge and fog computing. These advantages are of particular concern to resource-poor settings like smart homes (Sukumar, 2024).

2.5. Security Evaluation and Performance Practices

Zulberti et al. (2024) created a hardware/software design framework to check ECC implementations even though their focus is on cars and embedded control systems, their results showed more than 40% better speed and energy efficiency it means these techniques can also be applied to home automation system it proves that ECC is scalable cryptographic solution not just a theoretical idea (Zulberti et al, 2024)

2.6. Gaps and Justification

Although there has been progress in ECC protocols throughout IoT domains, most studies adopt simulation tools or hardware-dependent environments like Raspberry Pi, AVISPA, or Java-based servlets. They could potentially hinder usability and reproducibility for developers or students looking for lightweight implementations. This study bridges the gap by using Python only in Visual Studio Code, proving that ECC-based authentication protocols can be designed, implemented, and benchmarked effectively in simplified environments. This decision increases the practicality and reproducibility of the results for extensive use in the cybersecurity community, particularly for small-scale or prototype smart home systems.

The deployment of ECC(ECC) in resource-constrained IoT devices has been an intense research interest for achieving a good trade-off between high security and low resource usage. Marin, Pawlowski, and Jara (2015) designed an optimized ECC approach that was specifically customized for heterogeneous IoT networks, incorporating MSP430 and NXP/Jennic 5148 processors into a secure communication protocol. They particularly focused on interoperability and scalability in constrained environments with the deployment of twisted Edwards curves and Montgomery ladder methods, which ensured more effective scalar multiplication and reduced memory utilization.

Though they concentrated their research efforts on ECC arithmetic optimization for heterogeneous hardware, they also incorporated ECC into a bespoke Schnorr-based authentication and key negotiation protocol (EAP-SCHNORR), which was made to work securely under IEEE 802.15.4. ECC optimizations significantly lowered computational overhead and memory consumption on MSP430 devices by offloading more computationally intensive operations to more powerful nodes. This ordered arrangement accommodated cost-effective and scalable network topologies like extended star, which are prevalent in the context of smart homes.

While the initial study included hardware simulation on several platforms and network protocol levels, the findings on ECC's strengths in low-power environments are very applicable to this practicum. By comparison, the present project extends this work on a lightweight, software-only simulation created with Python through Visual Studio Code. It showcases mutual authentication with ECC and RSA in an MQTT-based publisher-subscriber model. This simplification makes

testing ECC security and performance possible without demanding hardware or multiprotocol environments, resonating with efficiency aims described in Marin et al.'s framework.

Additionally, the primary negotiation methods and replay attack prevention via timestamping investigated in this practicum align with the EAP-SCHNORR mechanism, which employs Schnorr signatures for secure authentication with low latency. Whereas Marin et al. employed a variety of cryptographic hash functions (SHA1, SHA256, SHA512) to evaluate network packet performance, this practicum focuses on real-time authentication success and system overhead (CPU/memory utilization), reaffirming ECC's feasibility in contemporary smart home environments.

3. Research Methodology

This chapter presents the systematic and experimental approach followed to develop and experimentally test an efficient mutual authentication protocol for IoT smart homes based on ECC(ECC). The implementation also considers Rivest-Shamir-Adleman (RSA) for comparison purposes. Both cryptographic methods are compared based on performance, key efficiency, energy consumption, and resilience to replay and impersonation attacks.

3.1. Research Strategy and Objectives

The approach takes an experimental-comparative design. Two prominent public-key cryptographic algorithms, ECC and RSA, are employed, and both are tested in both static key and dynamic key authentication modes. The objective is to evaluate authentication success, timing efficiency, signature verification performance, CPU/memory load, and system responsiveness in a smart home simulation.

3.2. Tools and Development Environment

Component	Technology
IDE	Visual Studio Code
Programming Language	Python 3.12
Cryptographic Libraries	Cryptography, ecdsa, rsa
Communication Protocol	MQTT using paho-mqtt
Socket Programming	Python socket module for key exchange
Performance Monitoring	Time, psutil, os libraries

Table 1: Tools and Environment

No outside simulators such as CrypTool, Packet Tracer, or Raspberry Pi were utilized. This was done intentionally to prove that a strong authentication protocol can be created and tested using simple scripting and logical simulation.

3.3. Protocol Structure

The mutual authentication protocol created in this research was designed after the communication and security needs of a common smart home system, in which devices are required to authenticate

the validity of messages before conducting sensitive tasks. The protocol framework follows a four-phase key generation, message signing, message transmission, and signature verification process. This framework was used uniformly for both ECC and RSA implementations to enable a direct and systematic efficiency and performance testing.

3.3.1. Key Generation

The initial step is the generation of cryptographic keys that form the basis for secure communications. For the static mode, ECC and RSA key pairs were pre-generated using the cryptography, ecdsa, and rsa Python libraries and saved as .pem files. These static keys are mimicking long-term device identities, which are typically used in commercial smart home installations. Under the dynamic mode, fresh key pairs are created at runtime for each session. For ECC, we chose the SECP256R1 curve, while RSA keys were generated using a 2048-bit modulus. Public keys are shared using Python sockets before the start of message transmission to guarantee session uniqueness and strong security.

3.3.2. Static Key

At first pre- pre-computed static keys were established and saved as .pem files for both ECC and RSA. This method made it easier to directly compare the performance of the algorithms since both were used with fixed credentials across multiple sessions. These static keys copy the way most real IoT systems keep long-term device identities.

3.3.3. Dynamic key

Generation mode was implemented, in which a new key pair is generated at runtime for each authentication session. Public keys are shared over a Python socket before secure MQTT communication. The dynamic approach proves to have stronger authentication since it provides a unique key for each session and prohibits credential reuse and offers better defense against replay and impersonation attacks.

Using both static and dynamic techniques, the project managed to analyze performance trade-offs while still highlighting the benefits of ephemeral, session-based security.

3.3.4. Message Signing

Once keys are generated, the publisher sends a command message, like Turn on Light or Open Garage Door, and adds a timestamp for freshness. The timestamp is used to defend against replay attacks.

The publisher's private key signs the timestamp and message.

For ECC, ecdsa, and cryptography libraries, along with SHA-256 hashing, are utilized.

For RSA, PKCS#1 v1.5 padding with SHA-256 hashing was used.

This creates a signed payload comprising the plaintext command and its digital signature**.**

3.3.5. Message Transmission

The signed payload is then sent to the subscriber. MQTT was used as the major communication channel because of its lightweight publish-subscribe model, which finds widespread applications in IoT systems. The publisher publishes the signed message to a specific topic, and the subscriber listens on that same topic to get notifications. In the dynamic mode, public keys are shared before MQTT communication is done using Python sockets, and then MQTT takes over to carry signed messages in real time. This two-layer communication arrangement enables secure key distribution without compromising lightweight messaging.

3.4. Publisher and Subscriber

In this project, the publisher and the subscriber are the two main factors in the smart home communication systems design. These entities were implemented in Python using MQTT and socket communication to simulate the communication of devices in the real world.

The publisher acts as the command creator, which is the smart home controller or user interface from which commands are created. It is responsible for building a command message, for example, Turn on Lights or Unlock Door, putting a timestamp to provide freshness, and signing the message with its private key (ECC or RSA). The signed message is published to the subscriber via the communication channel. Logically, the publisher is like the homeowner who clicks a button on a mobile app or smart hub to send a command.

The subscriber is the modern device in the house, such as a lock, light, or garage door. It is always listening for instructions from the publisher. When an event arrives, it looks at two things: The digital signature is identical to the publisher's public key to ensure the command is from a trusted source. The timestamp is recent to ensure it's not a replay of a stale command.

If both tests are successful, the device validates the command and carries out the operation. Otherwise, it denies it as an authentication failure.

This arrangement illustrates how actual smart homes operate, where a central controller issues commands and devices authenticate them for security. It illustrates how ECC and RSA can be used to safeguard IoT homes against unauthorized access.

3.4.1. Signature Verification

Upon receiving a message, the subscriber undergoes three verification processes:

Public Key Validation: The subscriber imports the publisher's public key (static. pem or dynamically received).

Signature Verification: The digital signature is verified with the message received.

Timestamp Freshness Check: The embedded timestamp is verified with system time to be inside a specified validity window.

If all checks succeed, the subscriber logs an "Authentication Success" message and executes the command.

If verification fails due to an invalid signature, an expired timestamp, or a mismatched key, the message is discarded and logged as "Authentication Failed." This provides robust protection against impersonation, tampering, and replay attacks. The structured protocol guarantees that both RSA and ECC implementations go through the same communication and verification processes, so their efficiency can be compared directly. Through the incorporation of both static and dynamic key modes, the protocol framework encompasses the performance-security trade-offs of long-term versus session-based authentication, enabling complete understanding of their suitability for smart home IoT settings.

4. Design Specification

This section demonstrates the blueprint design of the authentication system, which has been put in place to ensure the provision of security to IoT-based smart home environments. In contrast to the methodology that describes the research process, the design specification addresses the system architecture, the cryptographic configuration, as well as the comparison mechanism. The implementation uses both RSA and ECC within a publisher-subscriber paradigm that uses MQTT to communicate systematically to compare both parameters of performance against security.

4.1. Brief Introduction to the System

The smart home framework of authentication is based on a publisher-subscriber system of communication that is based on the way a controller (Publisher) and IoT devices (Subscribers) communicate with one another. The Publisher will make certain requests, like to turn on the light or unlocking a door, and that is signed with the cryptographic keys before sending it. The Subscriber checks the authenticity of each command before executing it. These entities communicate with each other by taking the help of the MQTT protocol, which is lightweight and suitable for IoT devices. RSA and ECC were used as the 2 models in this architecture, one to offer a basic (RSA) and one a minimalized (ECC) implementation.

4.2. RSA Design

The baseline scheme chosen was RSA, and key pairs of 2048 bits were generated with the use of the Python cryptography library. It also has the static (loading keys in.pem files) and dynamic (generating new key pairs at runtime) modes.

Publisher: Signs messages sent out using its key.

Subscriber: Checks the signature with the public authority of the Publisher.

Replay Protection: It is provided by inserting timestamps; messages beyond a 5-second freshness window are discarded.

Static RSA implementation is a motivation for the inadequacy of key sizes and computational overheads, and **dynamic RSA** is a simulation of session-based authentication that is much more secure.

4.3. Design of ECC

ECC was also adopted to be used as the main authentication method with the SECP256R1 specification and key pairs of 256 bits. This is the same security as RSA-3072 with smaller key sizes and less overhead. Dynamic key generation and both static PEM keys were combined.

Publisher: Signs messages with ECC private key.

Subscriber: ECC public key uses signatures.

Dynamic ECC: Creates key pairs per session, so there is freshness.

Replay Protection: Can be achieved by timestamps of messages.

The deployment is lightweight, energy-efficient, and scalable for IoT applications, with ECC being designed accordingly.

4.4. Artefact Implementation Tools

The implementation was done in **Python 3.12** using Visual Studio Code (VS Code). The libraries that were used are as follows:

Cryptography Crypto tools used to generate RSA and ECC keys, sign and verify signatures.

ecdsa - for ECC with SECP256R1.

paho -mqtt to use publisher-subscriber communication.

The project was implemented through the Bash shell using a Bash script and **Python language** using Python program. Both were aimed at measuring execution time and material resources used by the system. To this end, the use **of psutil and time** packages in Python, as well as the use of Bash timing commands, was utilized. Moreover, **base64 and JSON** have been used so that the encoding and structured messaging can be carried out efficiently. Even though tools like CrypTool2, Cisco Packet Tracer, and Raspberry Pi were part of the initial plan, the final execution was executed in VS Code only. This has been done due to the successful implementation of all targeted results in a controlled system, and the more sophisticated tools are to be employed later.

5. Implementation

The ultimate authentication framework was conducted in Visual Studio Code (VS Code) with Python 3.12 on macOS. Everything was run using the built-in Bash shell. It contained three fundamental steps in the design of key generation, authentication (RSA and ECC, static, and dynamic), and performance/comparison testing. In this part, the scripts that have been run, their roles, and their implementation outputs will be described.

Even though initially, CrypTool2, Cisco Packet Tracer, and Raspberry Pi were considered, the resulting solution was narrowed to VS Code. This hit a middle ground that had a controlled setting to address all the requirements of the research as well as the software-level validation as a future task.

5.1. Implementation of key generation

RSA and ECC authentication frameworks were based on key generation. Implementation of scripts was done to create the public and private keys of each algorithm.

Key generation:

Procedurally generated based **on ECDSA SECP256R1** curve, a 256-bit secret key and a public key are produced.

For re-use in static authentication mode, keys were exported to. pem format.

Files with contents: ecc_private_key.pem, ecc_public_key.pem.

RSA Key Generation:

It will generate a **2048-bit modulus**, which generates RSA pairs of private and public keys. They were also exported to .pem files so they could be used statically.

Files: rsa_private_key (pem). rsa_public_key (pem).

5.2. Introduction of Authentication

Both RSA and ECC were also authenticated over connections in both the static and dynamic key states, and it was simulated by taking real real-life publisher-subscriber communication system in a smart home.

5.3. ECC Authentication

publisher.py (ECC): publisher keys (generation or load), timestamped-signs outgoing messages and publishes them on MQTT.

subscriber.py (ECC) -Subscriber script to authenticate signatures using the publisher's public key and validate the timestamp to prevent replay attacks.

ecc_auth_dynamic.py - Dynamic ECC publisher that generates new key pairs on the fly to be used in session-based authentication.

ecc_auth_dynamic_subscriber.py -Dynamic ECC subscriber negotiates keys when the session starts and verifies the dynamic signature.

RSA Authentication

rsa_authentication_publisher.py Publisher script stores signatures that are produced against considered as the private key in the message it sends and inserts date fields.

rsa_authentication_subscriber.py – Subscriber script that verifies signatures using the RSA public key and rejects invalid or ageing messages.

rsa_auth_dynamic.py - Publisher script that creates short-term RSA keys at a time.

rsa_auth_dynamic_subscriber.py - Subscriber side of RSA dynamic mode authentication with replay protection and to maintain supplementary verification on a session basis.

5.4. Comparison Implementation

In order to be able to systematically test RSA and ECC, some Python scripts were created to collect and store key performance indicators.

Comparison of key size

check_key_size.py, check_rsa_key_size.py Loads keys in files ending in. pem and prints bit lengths (ECC = 256 bits, RSA = 2048 bits).

Comparison of Signature Size

signature_size_comparison.py - Signs the same messages using RSA and ECC and prints out the signature sizes to be used when analyzing bandwidth overheads.

Comparison of Signature Time

signature_time_comparison.py - Keeps a record of signing and verification time of RSA and ECC with time.perf_counter().

Throughput, Latency

rsa_throughput_latency.py, ecc_throughput_latency.py – Sign/verify 1000 times, to measure throughput (ops/sec) and average latency.

Performance Monitoring

rsa_performance.py, ecc_performance.py- monitor CPU and memory on signing and verification, and measure efficiency.

Energy Estimation

energy_estimation.py - Takes execution time and CPU use and produces an estimate of energy in joules. These modular scripts of comparison primed the accurate measurement of each parameter of performance and created a possibility of transparency and repeatability.

5.5. Implementation Brief Description

The execution involved 3 major phases comprising key generation, authentication, and comparative testing. Both RSA and ECC are utilized in both forms of authentication: static and dynamic, with replay protection being done by using timestamps. Special scripts were developed to test the size of keys, signature, the time taken, throughput, amount of memory used, and the energy consumed. The results of the given implementations verified that both RSA and ECC are

working as intended and served as a basis for the comparative assessment that will be carried out in Section 6.

6. Evaluation and Results

The value judgment examines the results of the devised RSA and ECC authentication systems in the IoT system of smart homes. Various Python scripts in Visual Studio Code were used to test the implemented system, authenticating, managing keys and comparing their performance in a controlled simulation.

6.1. Results of Authentication

The authentication was done using publisher-subscriber messaging in the RSA as well as the ECC frameworks. The outputs confirmed that commands like an on command like a Turn on Lights and the Open Garage Door were only carried out when messages are signed using the publisher's private key and the verified messages succeeded at the subscriber using the publisher's public key.

ECC Authentication

The subscriber is used only to show the status of Authentication Successful in microseconds after receiving a message. Prevention of replay attacks was confirmed by the validation of the timestamps so that stale messages were discarded.

RSA Authentication

RSA authentication was successful; on the other hand, signing and verification functions incurred a bit more latency than those of ECC. Messages with an expiration date lower than the designed 5-second cap were simply rejected, as they should be, being resistant to replay attacks. It proves that RSA and ECC effectively reached message integrity and authenticity, where the latter proves to be better suited in real-time interaction with IoT, as it is executed faster.

6.2. Dynamic Authentication Results

Dynamic authentication was used in both RSA and ECC, whereby each communication session has its key pair generated and not pre-generated key pairs. The mechanism enhances immunity to replay and key-compromise attacks.

ECC Dynamic Authentication:

Messages signed using new keys were routinely authenticated successfully by the ECC dynamic subscriber ("Auth Success in 0.0006s"). Inclusion of the timestamps made sure that commands that were more than 5 seconds old were declined on replay. Its outputs ensure message integrity and freshness of communication.

RSA Dynamic Authentication:

RSA dynamic authentication also achieved successful results in the establishment of trust at the level of a session. Only commands signed with the newly generated RSA key could be accepted

by this subscriber, e.g. a command like “Turn on Lights”. But the RSA computational overhead was more prominent than ECC and signing and verification were a bit slower.

Comparison Insight:

Both algorithms were made more resilient to replay and impersonation attacks, due to dynamic authentication. ECC did, however, have a higher security advantage with smaller key sizes (256-bit ECC) (2048-bit RSA security) according to crypto analysts and lower processing requirements, meaning it is more practical in IoT devices with limited resources.

6.3. Comparison of Signature Time

Running the signature_time_comparison.py script gave quantifiable results in both algorithms:

ECC Results:

Algorithm	Signature Generation time (s)	Signature Verification Time(s)
ECC	~0.0042	~0.00007
RSA	~0.0007	~0.000035

Table 2: Shows results of ECC-based comparison

Nevertheless, even though RSA was marginally quicker in isolated signature generation and verification tests, it was found to be highly inefficient when considering the size of the key and energy consumption.

6.4. Comparison of signature sizes

The most important distinctions in signature sizes were listed in the signature_size_comparison.py script:

ECC: 64 bytes (88 bytes Base64 encoded).

RSA: 256 bytes (344 bytes Base64 encoded).

It is a better alternative to bandwidth-constrained IoT networks, as the extremely small ECC signatures make it suitable to have a minimal overhead of the communication via MQTT.

6.5. Comparison of Throughput and Latency

Critical path tests ecc_throughput_latency.py and rsa_throughput_latency.py were run:

ECC Results:

Algorithm	Signing Throughput(ops/sec)	Verification Throughput (ops/sec)	Signing Latency(s)	Verification Latency (s)
ECC	2489	649	0.000402	0.00154
RSA	~2465	~59,076	0.00040	0.000017

Table3 : Shows ECC and RSA Throughput and Latency results

RSA had very high verification throughput because of bigger values of pre-computation efficiencies; however, ECC had reasonable performance in both signing and verification, and required fewer resources.

6.6. Units of Energy Consumed Comparison

Estimates of energy usage were obtained using the energy_estimation.py script; this is used to calculate the amount of energy used per CPU time and with an average level of power:

ECC:

Algorithm	Signing Energy(J)	Verification Energy(J)
ECC	~0.83	~0.81
RSA	~1.39	~1.31

Table 4: Energy consumed units in ECC

ECC used nearly half the energy, with RSA proving it to apply to energy-constrained IoT device such as sensors and smart locks.

7. Discussion and Critical Analysis

The comparison has shown that both RSA and ECC offered secure credentials authentication and replay protection in the IoT network of the smart home; however, ECC performed significantly better than Key and overall superior to RSA in all aspects of efficiency, scalability, and applicability to both constrained and resource-limited devices. In academic studies and practitioner agreement, the results support these findings.

7.1. Replay Protection and Dynamic Authentication

This was highly secured by dynamic key generation and timestamp validation that ensured that there was no replay of a message or acceptance of stale messages. RSA and ECC both worked well under this model, although the fact that ECC has a lighter computational overhead allowed setup of sessions and verification of messages faster. It is concordant with Wang et al. (2023) who state that lightweight cryptographic protocols with an adaptive key management approach are more scalable and resilient in the context of IoT, and similarly with Liu et al. (2024) who validated that ECC-based authentication of MQTT prevents the replay attack effectively and is efficient but with a high level of detail.

7.2. Key Size--Signature Size Efficiency

The experiments pointed out that ECC is at least as secure as standard signature algorithms (256-bit ECC = 2048-bit RSA). In a like manner, ECC signatures were four times smaller than RSA (64 bytes against 256 bytes). This shallowness is essential when MQTT communication is involved, as smaller messages reduce bandwidth intensity and make messages to be delivered faster. Sridevi et al. (2020) have shown higher security-per-bit performance of ECC than RSA, and Cogito Group (2024) affirm that exponentially longer keys of RSA render it progressively unrealistic in IoT

applications. The comparative study by St. Mary (2024) further confirms the effectiveness of ECC in the realization of digital signatures, especially in memory and bandwidth scenarios.

7.3. Latency Versus Performance Trade-offs in Authentication

In some circumstances, RSA could provide higher peak verification throughput (=59,000 ops/sec), however, results were not consistent and required heavy resource usage. ECC, on the other hand, had much more stable throughput (=2,489 signing ops/sec, 649 verification ops/sec) with close to 50 % less latency on signing and verification. IoT Systems, which are subject to unpredictability or responsiveness rather than rare peak performance, benefit more from ECC as a solution. Wang et al. (2023) also found that ECC is better able to scale in constrained IoT environments with an RSA key size at an exponential cost, which makes it not feasible over the long term.

7.4. Resource Constraint & Energy Saving.

Among the most compelling results was energy efficiency, which was less than ECC: ECC had 40 % lower energy to sign and verify than RSA. This has a direct impact on extending the life of the devices in battery-powered IoT devices like locks, cameras, and sensors. The efficiency of ECC has also been mentioned by Sridevi et al. (2020) and Cogito Group (2024) gives slight credit to energy efficiency as a reason to use ECC on embedded systems. These conclusions are supported by Rana (2024), who accentuates the popularization of ECC in IoT, mobile, and blockchain environments because of its shorter keys and reduced power consumption.

7.5. Improved Security in MQTT

MQTT does not support strong authentication and replay protection by default and is at risk of man-in-the-middle and injection attacks. The artefact of stamping ECC-based signatures with timestamps provided strong protection to MQTT and did not add high overheads to it. Similar findings are presented by Liu et al. (2024), according to which ECC improvements have high authentication and confidentiality and remain lightweight. Also, the fact that the IETF draft on SM2 (2024) emphasizes the global shift to standards-based international communication with elliptic curves (which also reaffirms the popularity of ECC).

7.6. Precarious boundaries and views

Despite the superiority of ECC, RSA is widely used because it has a long legacy, backward compatibility, and a lot of library integration (Cogito Group, 2024). RSA can be needed even in an inefficient manner in settings where backward compatibility is crucial. What is more, experiments designed within the framework of this project were performed on macOS in a controlled VS Code environment, not on physical IoT devices. According to Wang et al. (2023), more problematic side effects may be revealed by direct implementation to a limited device like Raspberry Pi or ESP32, including power consumption and network connection volatility. Lastly, although ECC is efficiency-oriented nowadays, both RSA and ECC are prone to post-quantum cryptanalysis, which will require post-quantum (or even hybrid) systems in smart homes in the future (rana 2024).

8. Conclusion & Future Work

This work aimed to test the hypothesis that the Elliptic Curve Cryptography (ECC) provides a more effective and scalable approach than the Rivest-Shamir-Adleman (RSA) to the mutual authentication in the IoT smart house. The study compared the authentication time, CPU and memory consumption, the size of signatures, the throughput, the latency, and the consumption of energy by implementing both algorithms under the same circumstances in Python with the MQTT protocol.

The findings proved the claims that both ECC and RSA are secure in the authentication and replay protection, but ECC always surpassed RSA in the metrics associated with efficiency. Precisely, ECC was able to attain comparable security using much smaller key sizes, signatures, and less computational load, and incurring less energy usage. These advantages make ECC more appropriate to resource-constrained IoT locations like smart homes, where the lifetime, speed, and energy efficiency of devices are important.

Consequently, the current study has been able to respond to the following research question: ECC offers a secure, lightweight, and scalable alternative to RSA to support mutual authentication in smart home IoT systems.

The continuation of these software-based results to hardware-based experiments with platforms like Raspberry Pi or ESP32 in the future would be worthwhile since it is possible to see factors such as power consumption, the stability of wireless transmission, and physical device limitations more in real life. More research on the hybrid techniques that integrate ECC, and post-quantum cryptography would also provide long-term resilience as quantum threats become possible. Moreover, a combination of ECC and lightweight IoT communication protocols like CoAP or LoRaWAN can also serve as further confirmation of its usefulness in a wider smart home and industrial IoT setting.

9. Reference

Adeniyi, A.E., Jimoh, R.G., and Awotunde, J.B. (2024). A systematic review on elliptic curve cryptography algorithm for Internet of things: Categorization, application areas, and security. *Computers & Electrical Engineering*, 118, pp.109330–109330. doi:https://doi.org/10.1016/j.compeleceng.2024.109330.

Alghamdi, A.M. (2025). Design and analysis of a lightweight and robust authentication protocol for securing the resource-constrained IIoT environment. *PLoS ONE*, 20(2), pp.e0318064–e0318064. doi:https://doi.org/10.1371/journal.pone.0318064.

Ali, W., Dustgeer, G., Awais, M. and Shah, M.A. (2017). IoT based smart home: Security challenges, security requirements and solutions. *2017 23rd International Conference on Automation and Computing (ICAC)*. doi:https://doi.org/10.23919/iconac.2017.8082057.

Azrin Zahan, Hossain, M.S., Rahman, Z. and Shezan Arefin (2020). *Smart home IoT use case with elliptic curve based digital signature: an evaluation on security and...* [online]

ResearchGate. Available at: https://www.researchgate.net/publication/340883209_Smart_home_IoT_use_case_with_elliptic_curve_based_digital_signature_an_evaluation_on_security_and_performance_analysis

Chen, L., Moody, D., Regenscheid, A., Robinson, A. and Randall, K. (2023). *Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters*. [online] csrc.nist.gov. Available at: https://csrc.nist.gov/pubs/sp/800/186/final.

Choubisa, M. and Jajal, B. (2025). Lightweight ECC and Token-Based Authentication Mechanism for IoT. *International Journal of Scientific Research in Computer Science and Engineering*, 13(1), pp.32–37. doi:https://doi.org/10.26438/ijsrcse.v13i1.611.

Cogito Group. (2024). *RSA vs ECC*. [online] Available at: https://cogitogroup.net/rsa-vs-ecc/.

Eclipse Mosquitto. (2018). *Eclipse Mosquitto*. [online] Available at: https://mosquitto.org.

Heino, J., Gupta, A., Antti Hakkala and Seppo Virtanen (2022). On Usability of Hash Fingerprinting for Endpoint Application Identification. doi:https://doi.org/10.1109/csr54599.2022.9850305.

Ijraset.com. (2024). *Advancements and Comparative Analysis of Digital Signature Algorithms: A Review*. [online] Available at: https://www.ijraset.com/research-paper/advancements-and-comparative-analysis-of-digital-signature-algorithms.

Khalique, A., Siddiqui, F., Ahad, M.A., and Hussain, I. (2025). Lightweight authentication for IoT devices (LAID) in sustainable smart cities. *Scientific Reports*, 15(1). doi:https://doi.org/10.1038/s41598-025-10181-0.

Lahraoui, Y., Jebrane, J., Amal, Y., Lazaar, S., and Lee, C.-C. (2025). IECC-SAIN: Innovative ECC-Based Approach for Secure Authentication in IoT Networks. *Computer Modeling in Engineering & Sciences*, [online] 0(0), pp.1–10. doi:https://doi.org/10.32604/cmes.2025.067778.

Lauter, K. (2004). *THE ADVANTAGES OF ELLIPTIC CURVE CRYPTOGRAPHY FOR WIRELESS SECURITY*. [online] Available at: https://citeseerx.ist.psu.edu/document

Li, M. and Hu, S. (2024). A Lightweight ECC-Based Authentication and Key Agreement Protocol for IoT with Dynamic Authentication Credentials. *Sensors*, 24(24), p.7967. doi:https://doi.org/10.3390/s24247967.

Liu, K., Zhou, Z., Cao, Q., Xu, G., Wang, C., Gao, Y., Zeng, W., and Xu, G. (2023). A Robust and Effective Two-Factor Authentication (2FA) Protocol Based on ECC for Mobile Computing. *Applied Sciences*, [online] 13(7), p.4425. doi:https://doi.org/10.3390/app13074425.

Liu, Z., Liang, T., Lyu, J. and Lang, D. (2024). A security-enhanced scheme for the MQTT protocol based on domestic cryptographic algorithm. *Computer communications*, 221, pp.1–9. doi:https://doi.org/10.1016/j.comcom.2024.04.013.

Lopez, M.I. and Barsoum, A. (2022a). Traditional Public-Key Cryptosystems and Elliptic Curve Cryptography. *International Journal of Cyber Research and Education*, 4(1), pp.1–14. doi:https://doi.org/10.4018/ijcre.309688.

M Gobi, R Sridevi and R Rahini (2020). *A Comparative Study on the Performance and the Security of RSA and ECC Algorithm*. [online] Available at: https://www.researchgate.net/publication/344788441_A_Comparative_Study_on_the_Performance_and_the_Security_of_RSA_and_ECC_Algorithm.

Maretha Ruswiansari, Febrianto Surya Kusumah, Sigit Wasista and Mohamad Ridwan (2024). Secure Communication ECC-Based between IoT Device and Server. *INVOTEK Jurnal Inovasi Vokasional dan Teknologi*, [online] 24(1), pp.9–18. doi:https://doi.org/10.24036/invotek.v24i1.1165.

Marin, L., Pawlowski, M. and Jara, A. (2015). Optimized ECC Implementation for Secure Communication between Heterogeneous IoT Devices. *Sensors*, 15(9), pp.21478–21499. doi:https://doi.org/10.3390/s150921478.

Nyangaresi, V.O. (2022). Lightweight anonymous authentication protocol for resource-constrained smart home devices based on elliptic curve cryptography. *Journal of Systems Architecture*, 133, p.102763. doi:https://doi.org/10.1016/j.sysarc.2022.102763.

Patel, C. (2022). *Secure Lightweight Authentication for Multi-User IoT Environment*. [online] arXiv.org. Available at: https://arxiv.org/abs/2207.10353

Sukumar, N. and Ranganathan, S.G. (2024). *A Survey of Security Approaches Based on Elliptic Curve Cryptography*. [online] Ssrn.com. Available at: https://papers.ssrn.com/sol3/

Sinha, S. (2024). *State of IoT 2024: Number of connected IoT devices growing 13% to 18.8 billion globally*. [online] IoT Analytics. Available at: https://iot-analytics.com/number-connected-iot-devices/.

Statista. (2024). *Thailand: smart home users 2028| Statista*. [online] Available at: https://www.statista.com/forecasts/1410404/thailand-smart-home-users.

Vailshery, L.S. (2022). *Global number of connected IoT devices 2015-2025*. [online] Statista. Available at: https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/.

Vailshery, L.S. (2022). *Global number of connected IoT devices 2015-2025*. [online] Statista. Available at: https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/.

Vankayalapati, Sai Sreekar, Mookherji, S. and Odelu, V. (2023). *A Security Enhanced Authentication Protocol*. [online] arXiv.org. Available at: https://arxiv.org/abs/2312.15250

Verma, P. (2022). MQTT Security Challenges and Solutions. [online] 11(4). Available at: https://www.researchgate.net/publication/360919708_MQTT_Security_Challenges_and_Solutions.

Wang, H., Muñoz-González, L., Hameed, M.Z., Eklund, D. and Raza, S. (2023). SparSFA: Towards Robust and Communication-Efficient Peer-to-Peer Federated Learning. *Computers & Security*, p.103182. doi:https://doi.org/10.1016/j.cose.2023.103182.

Yang, Y., Lee, S.-H., Wang, J., Yang, C.-S., Huang, Y.-M. and Hou, T.-W. (2023). Lightweight Authentication Mechanism for Industrial IoT Environment Combining Elliptic Curve Cryptography and Trusted Token. *Sensors*, 23(10), pp.4970–4970. doi:https://doi.org/10.3390/s23104970.

Yusoff, M., Ishak, M.K., Lukman and Mohd (2024). Improving Smart Home Security via MQTT: Maximizing Data Privacy and Device Authentication Using Elliptic Curve

Cryptography. *Computer Systems Science and Engineering*, 0(0), pp.1–10. doi:https://doi.org/10.32604/csse.2024.056741.

Zhang, C., Liu, Y., Leng, F., Zhao, Q. and He, Z. (2023). *SM2 Digital Signature Algorithm for DNSSEC*. [online] IETF Datatracker. Available at: https://datatracker.ietf.org/doc/draft-cuiling-dnsop-sm2-alg/10/

Zulberti, L., Di Matteo, S., Nannipieri, P., Saponara, S. and Fanucci, L. (2022). A Script-Based Cycle-True Verification Framework to Speed-Up Hardware and Software Co-Design: Performance Evaluation on ECC Accelerator Use-Case. *Electronics*, 11(22), p.3704. doi:https://doi.org/10.3390/electronics11223704.