EDK IN RASPBERRY Pi

[1]Lingeshwar. R, [2]Roshan. M.S.K, [3]Sudarson. M.V, [4]Siva Raja Sri.KG, [5]Dr. S. Mohandoss

Student
Cyber Forensics And Information Security
Dr. M.G.R. Educational and Research Institute, Chennai, India.

Abstract- **This short note delves into the world of Raspberry Pi, a tiny but powerful computer, and how it's used for keeping information safe through something called encryption. Imagine it like putting your data in a secret code that only the right people can understand. We'll also touch on a cool project called Kryptor, which is all about making sure this process is easy and secure. It's like a digital superhero, helping Raspberry Pi keep your information locked up tight, so only you and the ones you trust can access it. Kryptor, the star of our show, is a nifty project designed to simplify the encryption process on Raspberry Pi. It acts like a digital guardian, providing an easy-to-use interface for users to encrypt their data without needing to be tech wizards. With Kryptor, you can ensure that your sensitive information, whether it's personal documents or smart home data, stays confidential and protected. The project leverages the open-source spirit of Raspberry Pi, encouraging collaboration and improvement in the realm of data security. So, whether you're a tech enthusiast or just someone who wants to keep their digital life safe and sound, Kryptor and Raspberry Pi make for a dynamic duo in the world of secure computing**

INTRODUCTION

In the era of ubiquitous computing, ensuring the security and integrity of conveyed information is essential data sharing. The computer platform Raspberry Pi and hardware-based encryption are employed in this abstract to provide a special way to provide robust data security. The **"Kryptor,"** the proposed system, combines the power of hardware security Field-Programmable Gate Arrays (FPGAs) or Hardware Supervisory Modules (HSMs) having the flexibility of Raspberry Pi to provide a dependable and efficient encrypted data transit solution. FPGAs, or field-programmable gate arrays, and HSMs, or hardware security modules offer specialist hardware resources designed with cryptography in mind. These components may be used to speed up the data encryption and decryption processes, which will reduce the Raspberry Pi's processing workload. This collaboration enhances the system's both performance and security, which qualifies it for a range of uses, such as Protocols for secure communication and sending private information. The Raspberry Pi single-board computer serves as the brains of the Kryptor system. It is affordable and compact. The encryption process and key generation It oversees methods, as well as communication interfaces. Users might easily configure and manage the encryption parameters with a simple and intuitive interface, enabling seamless incorporation into an array of data transmission situations. Secure communications, industrial automation, and the Internet of Things (IoT) are just a few areas in which the Kryptor system is meant to satisfy the pressing need for secure data exchange. This system offers an optimal blend of hardware-accelerated Leveraging the strength of encryption and flexible data processing capabilities Raspberry Pi and HSM/FPGA devices. Thus, Kryptor provides a workable method of increasing data. security while maintaining efficacy in resource-constrained environments.

OBJECTIVES

1. **HSM/FPGA Configuration:** Set up and configure your HSM/FPGA to handle encryption. This may involve installing necessary drivers, configuring encryption algorithms, and generating encryption keys securely within the HSM/FPGA.
2. **Raspberry Pi Configuration:** Establish communication between the Raspberry Pi and the HSM/FPGA. This might involve connecting via interfaces like USB, SPI, I2C, or other protocols depending on the hardware capabilities.
3. **Encryption Process:** Write code on the Raspberry Pi to send data securely for encryption using the HSM/FPGA. This involves establishing a secure channel, sending plaintext data to the HSM/FPGA, and receiving encrypted data back.
4. **Decryption Process (if needed):** If decryption on the Raspberry Pi is necessary, set up a similar process to receive encrypted data and decrypt it using the HSM/FPGA.
5. **Data Transmission:** Ensure secure transmission of encrypted data between the Raspberry Pi and other systems if data is being sent further.

IMPLEMENTATION

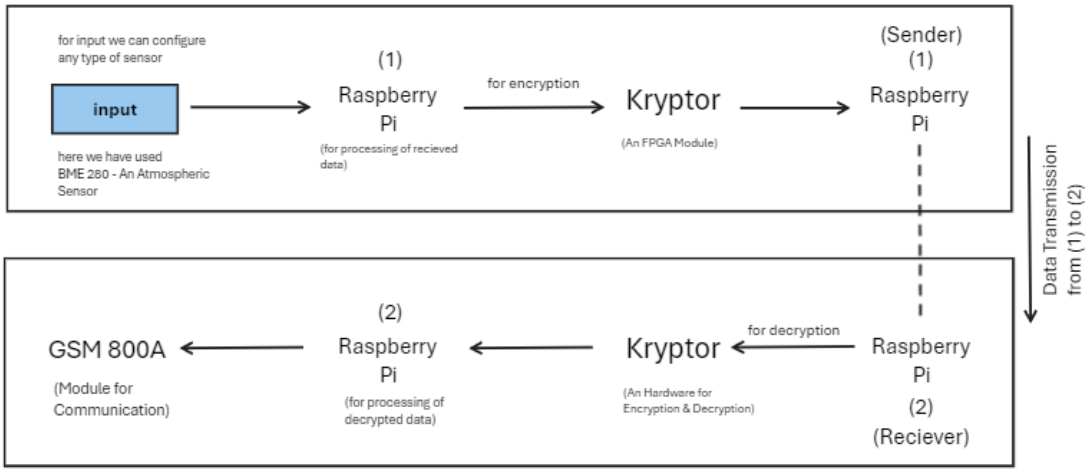

Figure: 1

The concept of Kryptor seems promising, combining the strengths of Raspberry Pi's adaptability with the specialized cryptographic capabilities of Hardware Security Modules (HSMs) or Field-Programmable Gate Arrays (FPGAs). Here's a breakdown and further insights.

Figure: 2

STRENGTHS OF KRYPTOR

Hardware Acceleration: HSMs and FPGAs are optimized for cryptographic operations, reducing the computational burden on the Raspberry Pi. This enhances performance while ensuring secure data processing.

Raspberry Pi as Control Hub: Leveraging Raspberry Pi as the control centre for key generation, encryption, and managing communication interfaces provides a user-friendly experience and easy integration into various scenarios.

Versatility: With applications across IoT, industrial automation, and secure communications, Kryptor addresses the growing need for secure data sharing in diverse fields.

Balanced Security and Efficiency: The fusion of hardware-accelerated encryption and Raspberry Pi's adaptable processing capabilities creates a balanced solution that enhances data security without compromising efficiency.

ENCRYPTION STANDARD

Symmetric encryption algorithms use a single key for both encryption and decryption processes. This key must be kept secret between the communicating parties to maintain the confidentiality of the data. The two main components of symmetric encryption are:

Key:

Secret Key Generation: The key is typically generated using a secure random number generator. The key length is crucial for the security of the encryption, and longer keys generally provide stronger protection.

Key Distribution: The biggest challenge in symmetric encryption is securely distributing the secret key to both the sender and the receiver.

Cipher:
Encryption Process: The plaintext, which is the original data, is combined with the secret key using a symmetric encryption algorithm to produce the ciphertext. The ciphertext is the scrambled and unreadable form of the original data.

Decryption Process: The recipient, possessing the same secret key, uses the symmetric decryption algorithm to convert the ciphertext back into the original plaintext.

Figure: 3

Symmetric Encryption Algorithms:
There are several widely used symmetric encryption algorithms, each with its unique characteristics:

Data Encryption Standard (DES):
DES was one of the earliest symmetric encryption algorithms.
It uses a 56-bit key, which is considered insecure for today's standards.

Advanced Encryption Standard (AES):
AES is the most widely used symmetric encryption algorithm today.
It supports key lengths of 128, 192, or 256 bits, providing a high level of security.

Triple DES (3DES):
3DES applies the DES algorithm three times consecutively to each data block.
It provides a higher level of security compared to DES, but it is slower.

Strengths and Weaknesses:
Strengths:
Efficiency: Symmetric encryption is generally faster than asymmetric encryption.
Security: With an appropriate key length, symmetric encryption provides strong security.

Weaknesses:
Key Distribution: The secure distribution of the secret key is a significant challenge.
Scalability: As the number of communicating parties increases, managing keys becomes more complex.

Use Cases:
Bulk Data Encryption:
Symmetric encryption is well-suited for encrypting large volumes of data efficiently.

Communication Security:
It is commonly used in securing communication channels between two entities.

POTENTIAL CONSIDERATIONS
Integration Complexity: Ensuring seamless integration between Raspberry Pi and HSMs/FPGAs might require careful development and compatibility checks to maximize the system's effectiveness.

Security Protocols: Robust security protocols and best practices for key management and data transmission should be implemented to fortify the overall security of the system.

Scalability: Evaluating Kryptor's scalability to handle varying data volumes and processing requirements across different applications will be essential for its widespread adoption.

FUTURE IMPLICATIONS

Further Research and Development: Continual refinement and research into Kryptor's design could lead to enhancements in secure data transfer techniques, contributing to a more connected and secure digital ecosystem.

Industry Adoption: The success of Kryptor will depend on its adoption within industries and applications that prioritize data security, potentially shaping future standards for secure data transmission.

CONCLUSION

In conclusion, the synergy between encrypted data via Kryptor, utilizing Hardware Security Modules (HSMs) or Field-Programmable Gate Arrays (FPGAs) in conjunction with Raspberry Pi, represents a highly compelling and versatile solution. This fusion of advanced security hardware and Raspberry Pi's accessibility offers numerous benefits.

REFERENCES:

[1] https://dl.acm.org/doi/abs/10.1145/3022860.3022862 “Processing Over Encrypted Data: Between Theory and Practice” - Eyad Saleh, Ahmad Alsa'deh, Ahmad Kayed, Christoph Meinel

[2] https://ieeexplore.ieee.org/abstract/document/8095259/ “Self-reconfigurable cryptographical coprocessor for data streaming encryption in tasks of telemetry and the Internet of Things” - Heorhii Vorobets; Oleksandr Vorobets; Valentyna Horditsa; Volodymyr Tarasenko; Olha Vorobets.

[3] https://ieeexplore.ieee.org/abstract/document/9491920/ “Securing CAN FD by implementing AES-128, SHA256, and Message Counter based on FPGA” - Farag Mohamed E. Lagnf; Subramaniam Ganesan.

[4] https://ieeexplore.ieee.org/abstract/document/6339242/ “FPGAs for trusted cloud computing” - Ken Eguro; Ramarathnam Venkatesan.

[5] https://ieeexplore.ieee.org/abstract/document/8279795/ “Flexible and low-cost HSM based on non-volatile FPGAs” - Diogo Parrinha; Ricardo Chaves.

[6] https://ieeexplore.ieee.org/abstract/document/8863722/ “Toward Data Security in Edge Intelligent IIoT” - Yong Yu; Ruonan Chen; Huilin Li; Yannan Li; Aikui Tian.

[7] https://ijcna.org/Manuscripts/Volume-2/Issue-1/Vol-2-issue-1-M-04.pdf “Exploring IOT Application Using Raspberry Pi”- Cheah Wai Zhao Quest International University Perak.

[8]https://search.ebscohost.com/login.aspx?direct=true&profile=ehost&scope=site&authtype=crawler&jrnl=09765697&AN=124636676&h=ItEU%2BU9rE7toxN1DYniX%2FApkxw89O%2B5MyoZECEIbfesfikLyv%2BpncneNf5hwcB8JRDZ4pmEhEmG2zxVBUDTaRg%3D%3D&crl=c “A Comparative Study of Arduino, Raspberry Pi and ESP8266 as IoT Development Board” - Patnaik Patnaikuni, Dinkar R.

[9] https://ieeexplore.ieee.org/abstract/document/6782081/,“Secure sensor node with Raspberry Pi” - Soham Banerjee; Divyashikha Sethia; Tanuj Mittal; Ujjwal Arora; Akash Chauhan.

[10] https://www.petsymposium.org/2021/files/papers/issue4/popets-2021-0072.pd “Fortified Multi-Party Computation: Taking Advantage of Simple Secure Hardware Modules”- Brandon Broadnax, Alexander Koch, Jeremias Mechler, Tobias Müller, Jörn Müller-Quade, and Matthias Nage.