End-to-End RSA Encryption with Lightweight Key Management in Edge Computing

Kristella B. Jackson
California State University Long Beach, College of Engineering
April 12, 2025

ABSTRACT

Today, edge computing, also known as the edge, continues to grow in the market as technology continues to evolve to require higher efficiency, meaning faster computation and more power, and the edge not only allows devices to reduce their own workload, but also to offload from cloud computing and distribute it to multiple edge servers. This saves computing power and resources for both devices and the cloud, meaning longer battery life for devices and lower costs for cloud resources. However, despite the great potential of the edge for future technology, its implementation introduces security risks.

Due to the distributed nature of edge devices that continuously collect, process, and store large amounts of potentially sensitive and/or confidential data, the attack surface increases as more edge devices are in use. This paper aims to address a list of security concerns with edge computing and to demonstrate a solution using RSA encryption and digital signatures to protect and validate data. In future works, the solution can be further improved to more solidly resolve these concerns.

1. INTRODUCTION

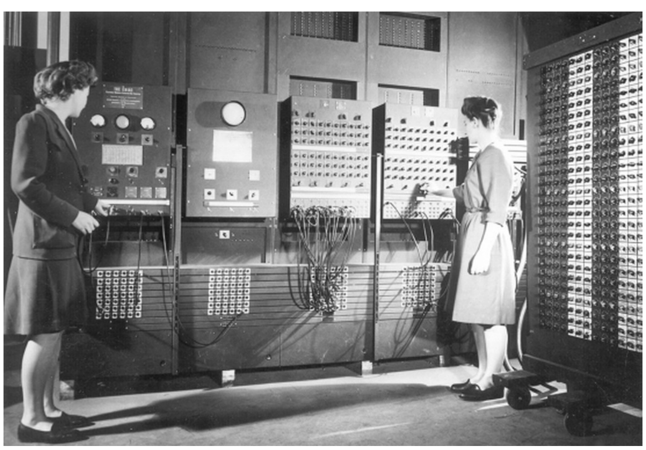

Figure 1: Operation of the ENIAC's control panel [6]

1.1 A Brief History of Computers

Computers have come a long way since the world's very first general-purpose electronic computer on the 14th of February 1946. The Electronic Numerical Integrator and Computer (ENIAC) was a very large computer with a room-sized mainframe that was constructed at Moore's School of Electrical Engineering, now called Penn's School of Engineering and Applied Science, and it was one of the first technologies to electronically compute complex mathematical problems at what was considered very high speeds at the time.

The amount of time that it would take even the smartest people to solve complex math equations with the help of function tables would be exponentially reduced when using the ENIAC which could perform 5000 additions per second. The amazement of the increase in efficiency in its time inspired the revolution of computer science and electrical engineering that continues today [6]. Figure 1 shows an old picture of Jean Bartik on the left and Frances Spence operating the ENIAC's main control panel.

Computers today are much smaller and mobile, especially laptops and smartphones, and users can do so much more than perform mathematical calculations; people can communicate through emails, use social media, get the news, research, write documents and presentations, develop and/or advertise products, simulate, and program and operate machines, you name it. With innovation, technology has evolved to be able to efficiently run multiple processes and perform numerous tasks, and part of this is with the help of edge computing.

1.2 Evolution of Edge Computing

Just like computers, edge computing was also part of the technology evolution. Edge computing began in the 1990s with Akamai's launch of its content delivery network (CDN), which it allowed static media to be cached closer to end users, and in 1997, Brian D. Nobel along with numerous contributors wrote a paper demonstrating a prototype model that allows applications to transfer certain tasks to strong servers when operating on resource-scarce mobile devices. This ability to offload tasks to servers in a heterogeneous distributed manner has become a main principle for technologies from businesses such as Google, Apple, and Amazon, and edge computing continues to grow in the market to meet the demands for better efficiency in evolving technologies [8, 10, 11].

1.3 Security Issues

Edge computing has played a role in the evolution of many mobile devices, 5G technologies, and other intelligent terminals that are used today, such as autonomous cars, industrial IoT devices (e.g., oil and gas failure detection), and numerous smart devices such as security cameras, mobile phones, home appliances, and to name a few. With the ability to offload

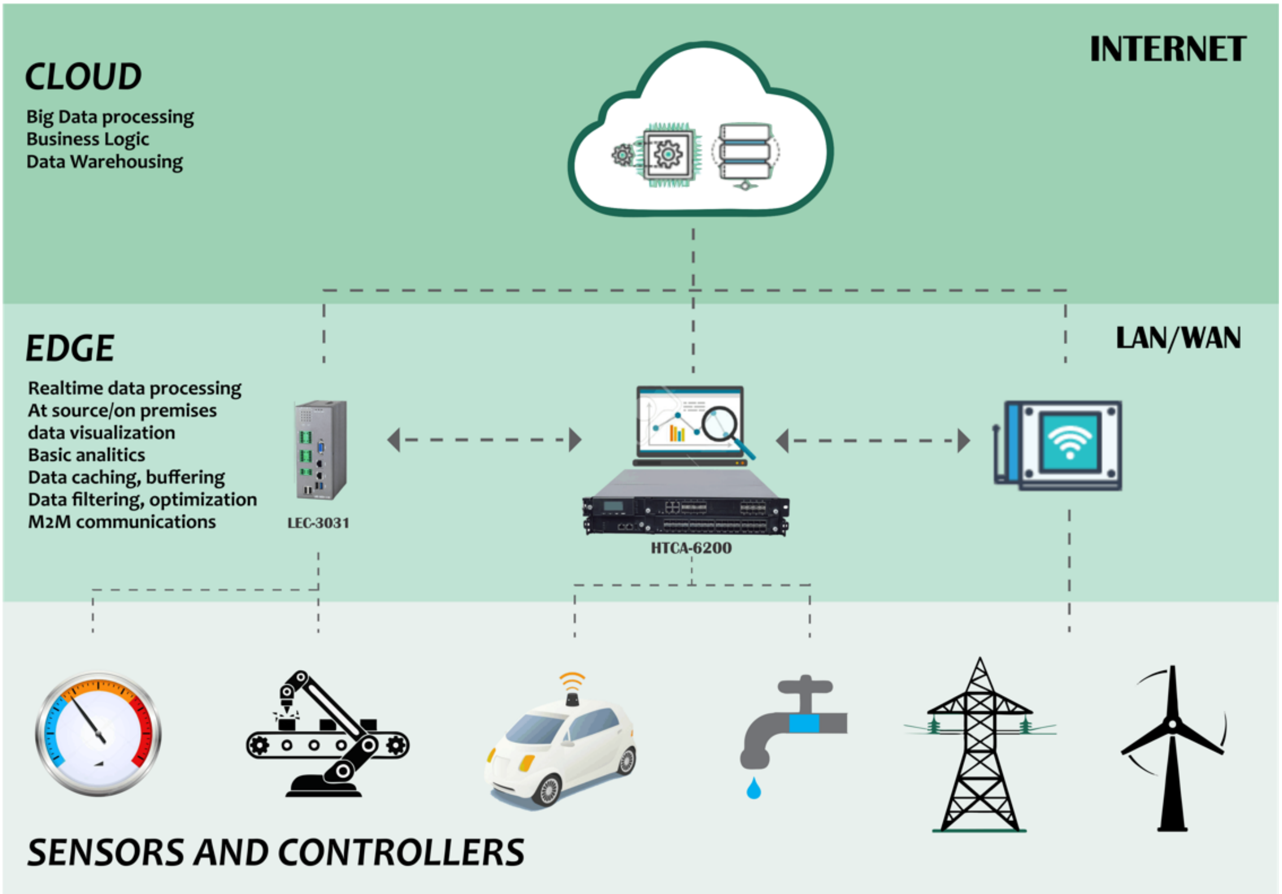

Figure 2: Edge Computing Diagram [3]

processes to edge servers instead of having to process directly on the user's end, the devices can perform faster and still save battery power and/or storage and memory space, allowing them to last longer.

Despite the ability to make data processing and storage more efficient, implementation of the edge also introduces some major security risks considering that devices nowadays also process sensitive and confidential data that need to be kept safe from unauthorized access. In addition to using the cloud and the increasing number of edge devices, this puts data privacy and integrity even more at risk [8]. Such security issues and risks include, but are not limited to:

- Password attacks
- Man-in-the-Middle (MitM) attacks
- Packet and Malware injection
- Compromised data from the cloud
- Manipulated data
- Account theft

1.4 Proposed Solution

For this paper, the main focus is to protect sensitive data over an edge network and maintain its integrity, and the proposed solution is to implement an edge network model that enables end-to-end protection and validation of a digital communication between an end user (client) and an edge server by using Rivest Shamir Adleman (RSA) encryption and digital signatures.

The RSA algorithm allows users to create two keys, a public and private key, that are mathematically linked to each other. The receiver can share their public key to anyone that needs to send protected data to them, and then the receiver can decrypt the data with their private key. Similarly, the sender can encrypt with their private key to digitally sign a message, and then the receiver decrypts with the shared corresponding public key to validate the signature [13]. For this experiment, a client and an edge server will generate their own key pair, and they will exchange each other's public keys to enable end-to-end encryption and validation. Then the client will be the first to send a unique, digitally signed encrypted message to the edge server, and once received, validated and decrypted, the edge server will return a "processed" digitally signed encrypted message back to the client.

To check for successful end-to-end encryption, a packet sniffer will be used to ensure the messages remain "hidden" during transit, and this packet sniffing will also serve as a passive man-in-the-middle attack demonstration. Upon arrival to the recipient, the decrypted message should be verified and match the original plaintext to check for successful decryption. As stated before, the keys are mathematically linked, so if both the public and private key get compromised, then it risks the privacy and integrity of data to being compromised as well for both communicating parties. The owner(s) of the keys must, at minimum, practice lightweight key management to minimize the risk of data security and integrity, such as practicing periodic key rotation and keeping the private key securely stored.

2. BACKGROUND

This section goes through more background information and vocabulary in detail related to edge computing, cybersecurity, and the RSA algorithm to help readers better understand the concepts in this paper.

2.1 Edge Computing

Table 1 gives a list of terms and definitions relevant to understanding edge computing.

Term	Definition
Cloud computing	Also known as the cloud, public cloud, or commercial cloud; A centralized computing model including massive amounts of resources that service end users by computing and storing large loads of data for them.
Edge computing	Also known as the edge, fog computing, edge cloud, or local computing; A more distributed computing model where storage and processing happens closer to the data source.
Edge device	Any device located at the edge that can generate and process data locally.
Edge server	A more powerful computing resource compared to an edge device; Located close to edge devices for more efficient data processing and storage.
Offload	To unload something from some kind of source.

Table 1: Edge computing terminology

Edge computing is widely used in the Internet of Things (IoT) considering that most of what we do today relies on the internet. With how evolved technology has become, the data volume for edge devices has also increased, meaning that to process larger volumes of data at reasonable speeds, especially in applications where fast, real-time processing is mandatory or dire, they require a lot more power to meet that demand. The same goes for resources in cloud computing, which is more centralized and remote compared to edge computing; by itself, the cloud would have to process large loads of data from millions of end-users almost at once, and would not be very efficient and doable for this centralized kind of fashion, so by implementing the edge, it decreases the amount of data that needs to be sent to the cloud by decentralizing computation to edge servers for data to be processed more locally, and this saves money and resources for the cloud [2, 3, 8]. Figure 2 shows a diagram of the general network with the edge in place.

The edge is built so that storage and computations are closer to the data source, allowing for lower latency, higher bandwidth, and better reliability for data storage and processing than when relying on cloud resources alone [9]; in cases where certain tasks are too voluminous to handle, require further analysis beyond the edge server(s)'s capabilities, or simply do not require real-time processing, edge servers can selectively offload those tasks to the cloud to maintain efficiency [7].

2.2 RSA Algorithm for Cybersecurity

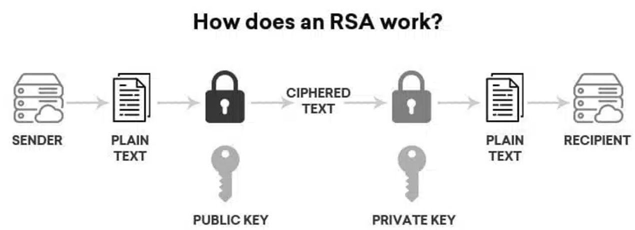

Figure 3: RSA Encryption Decryption [4]

Table 2 contains a list of terms relevant to understanding man-in-the-middle attacks and the use of RSA encryption for data protection in general networking.

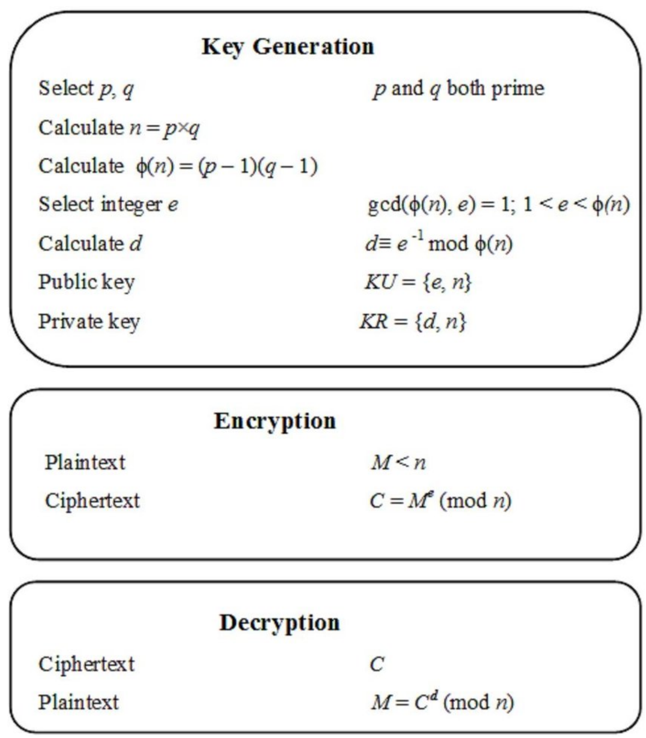

Figure 4: RSA Algorithm [12]

The RSA algorithm begins with the creation of the key pairs, the public and private keys. To generate the keys, first select two large prime numbers p and q (preferably of similarish size), then use p and q to calculate n and $\phi(n)$. Next, select an integer e that is greater than 1 and less than $\phi(n)$, and it must be co-prime with $\phi(n)$ (meaning that the greatest common denominator between e and $\phi(n)$ is 1), then use e to calculate for d. The result is a public key of (e, n) and a private key of (d, n).

To perform RSA encryption of a plain text message M, the message must be less than n, and you use the public key values e and n to calculate the output ciphertext C.

To read what is behind the ciphertext, RSA decryption needs to be performed on the ciphertext C by using the private key values d and n to calculate the output plaintext M. Refer to Figure 3 for a general understanding of RSA encryption and Figure 4 for the specific formulas used in each part of the RSA algorithm.

For digital signatures, the message to be sent must first go through a hashing algorithm (e.g., SHA-1, SHA-256 in Figure 5, etc.) and then encrypted using their own private key, then the sender attaches the signature to the original message and shares their public key for the receiver to use to decrypt the hash. The receiver can validate the message by putting the original message through the same hashing algorithm and check if theirs matches the decrypted hash value, and if they match, the message has not been altered and has maintained integrity.

The recommended minimum key size for strong enough network security is 2048 bits, and along with proper implementation of RSA encryption and verification in the edge and using simple key management practices, sensitive data can be sent securely through the edge network, and end users can maintain integrity of the data they receive.

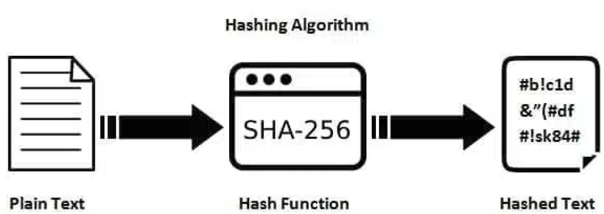

Figure 5: Hashing with SHA-256 [5]

3. IMPLEMENTATION

This portion of the paper will cover all the materials that were used to implement the experiment for end-to-end encryption and validation over an edge network. The section will first go over the general edge computing model built for the simulation, and then we will explain how the simulation runs for the model step-by-step as well as how the data is checked to verify proper transit. Note that this was all done on an Ubuntu 20.04 virtual machine in Oracle VirtualBox v6.1.16.

3.1 Edge Model

The overall edge computing model was developed in C++ with the NS-3 library v.42 can be found here. The code includes the creation of 10 client/end-user nodes, 3 edge server nodes, and 1 cloud node, and each edge node is connected to three to four clients and connected to the cloud. Only the second and third edge servers are also directly connected to each other. The point-to-point connections are all assigned a unique IPv4 address before setting up the sockets for data

Term	Definition
Rivest Shamir Adleman (RSA)	A public-key cryptosystem that cryptosystem that enforces a mathematical relationship between a public and private key for encrypting and decrypting data, and it can be used for digital signatures for secure communication and maintain integrity.
End-to-End Encryption	A security method for encrypting data that is only intended for the sender and the receiver to read to ensure privacy and security of data even if it is intercepted.
Private Key	An RSA key generated first; It is used for decrypting data, and the key cannot be openly shared with anyone (hence "private" key).
Public Key	A smaller RSA key derived from the private key pair; It is used for encrypting data, and anyone can use the key to send encrypted messages to the recipient (hence "public" key).
Ciphertext	The scrambled, unreadable text as a result from encryption.
Plaintext	The readable text prior to encryption or as a result from decryption.
Hashing	A one-way algorithm for producing a fixed-length string value with some message as input.
Digital Signature	An algorithm for hashing a message and encrypting it with the private key, which is then sent to and validated by the receiver with the corresponding public key.
Packet	A unit of data, typically broken down from a larger message that is transmitted across a network; It contains the piece of data itself (the payload) and the header containing the route information.
Packet Sniffing	A passive attack where data packets are simply observed while the packets are in transit, and the data is not altered in any way, nor are there additional packets of data from the attacker.
Man-in-the -Middle (MitM)	An attacker that intercepts end-to-end communication by pretending to be both the sender and receiver with the intent to manipulate or steal sensitive data, or insert foreign data.

Table 2: RSA and cybersecurity terminology

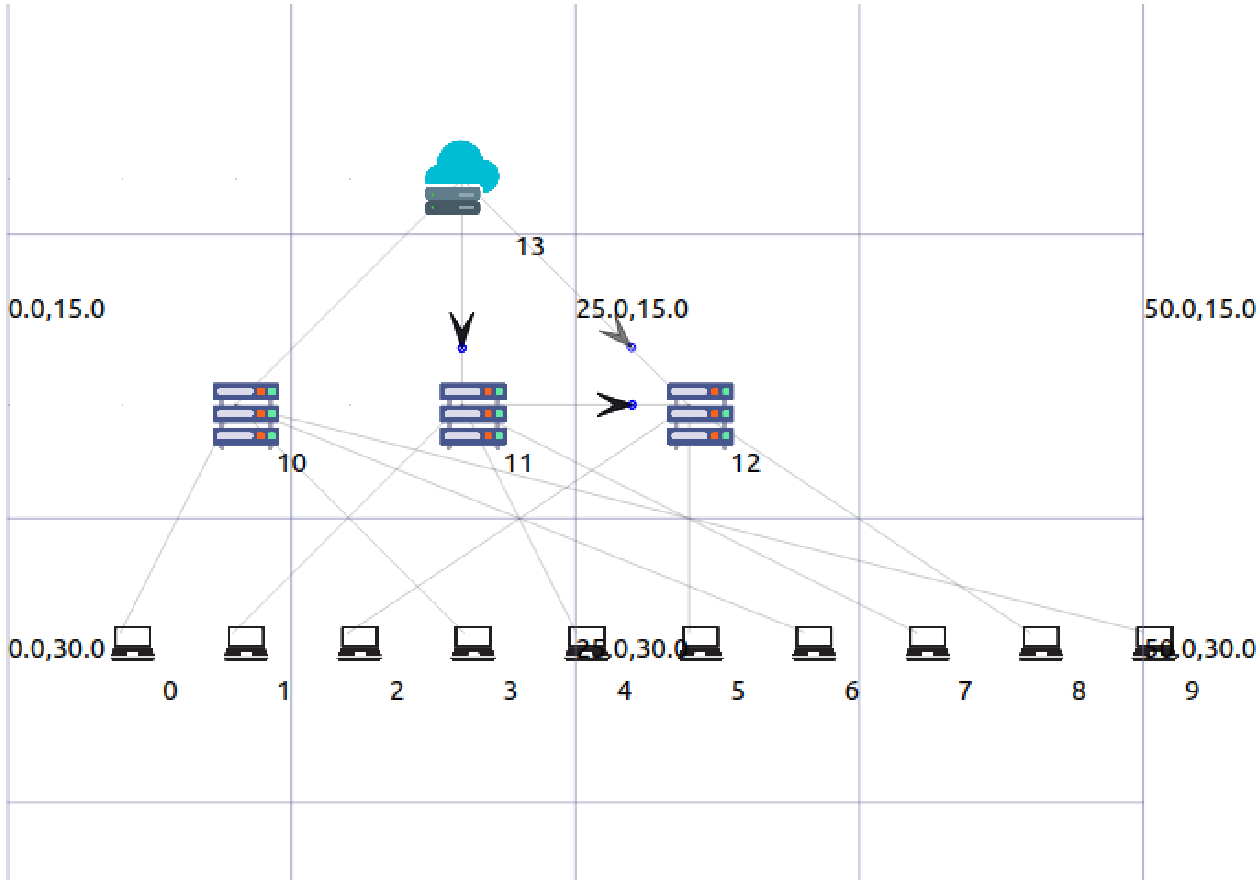

Figure 6: Edge Computing Model Simulation in NetAnim

routing. Each node is then assigned their own socket along with a pre-made message that they will send to the specified destination. In this case, each client will send their message to an edge node, and then the edge node will return a modified message to the corresponding clients.

The end-to-end security version of the edge model is generally the same in terms of the connections, but this time client node 0 and edge node 12 have RSA encryption and digital signatures enabled. To achieve this implementation, an OpenSSL v1.1.1f library built in to the Ubuntu virtual machine was used, and it is an open-source secure sockets layer protocol for ensuring privacy, authentication, and data integrity in any kind of internet communications.

3.2 Simulation

When running the code in the command prompt with the .\ns3 command, a .xml file is created, which then the NetAnim program is used to open the file and run the edge network simulation. Here is a detailed breakdown of the simulation run (refer to Figure 6 for the simulated model):

1. Edge server connections to client nodes 0 through 9 are assigned IPv4 addresses 10.1.1.1 through 10.1.10.1 respectively, connection of edge nodes 11 and 12 is assigned 10.1.11.1, and cloud connections to edge nodes 10 through 12 are assigned 10.1.12.1 through 10.1.14.1 respectively.

2. Client node 0 will send a unique data packet to edge node 12 while the rest of them will send their unique packets to edge node 11.

3. The data packets reach the edge nodes directly connected to them first.

4. Edge server 11 returns packets to clients 1, 4, and 7 first, while edge server 12 routes packets from clients 2, 5, and 8 to edge server 11, and edge server 10 routes packets from clients 0, 3, 6, and 9 to the cloud.

5. Edge server 11 returns packets to edge server 12 that will then be sent correspondingly to clients 2, 5, and 8, while the cloud routes client packets 3, 6, and 9 to edge server 11 and client packet 0 to edge server 12.

6. Data packets from edge server 11 and 12 are sent to the cloud, where they will then route back to edge server 10 and then distribute to their respective clients.

After running the code in the command prompt, it outputs what is happening between the communicating nodes. For example, the NS-3 program will display that client 0 with IP address 10.1.1.1 has sent its message to edge server 12 with IP address 10.1.14.1 and include a timestamp of when that happened, and then once reached, it will output when the message made it and then display the received message:

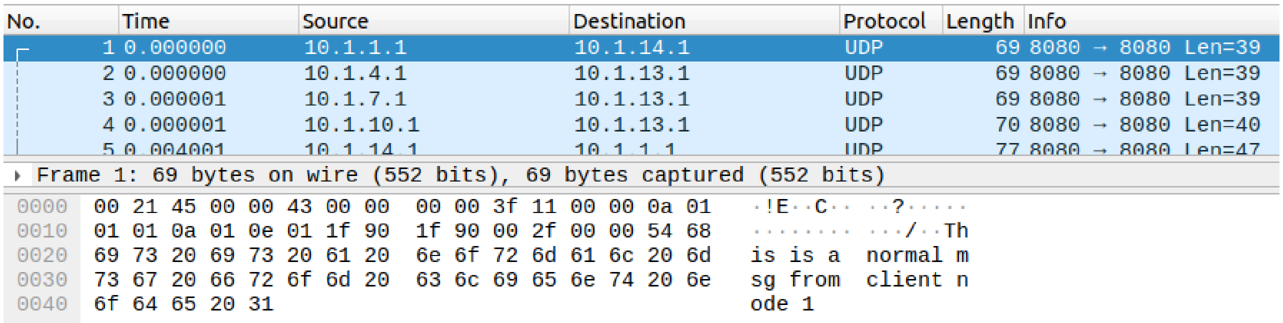

Figure 7: Message for Normal Edge Model in Wireshark

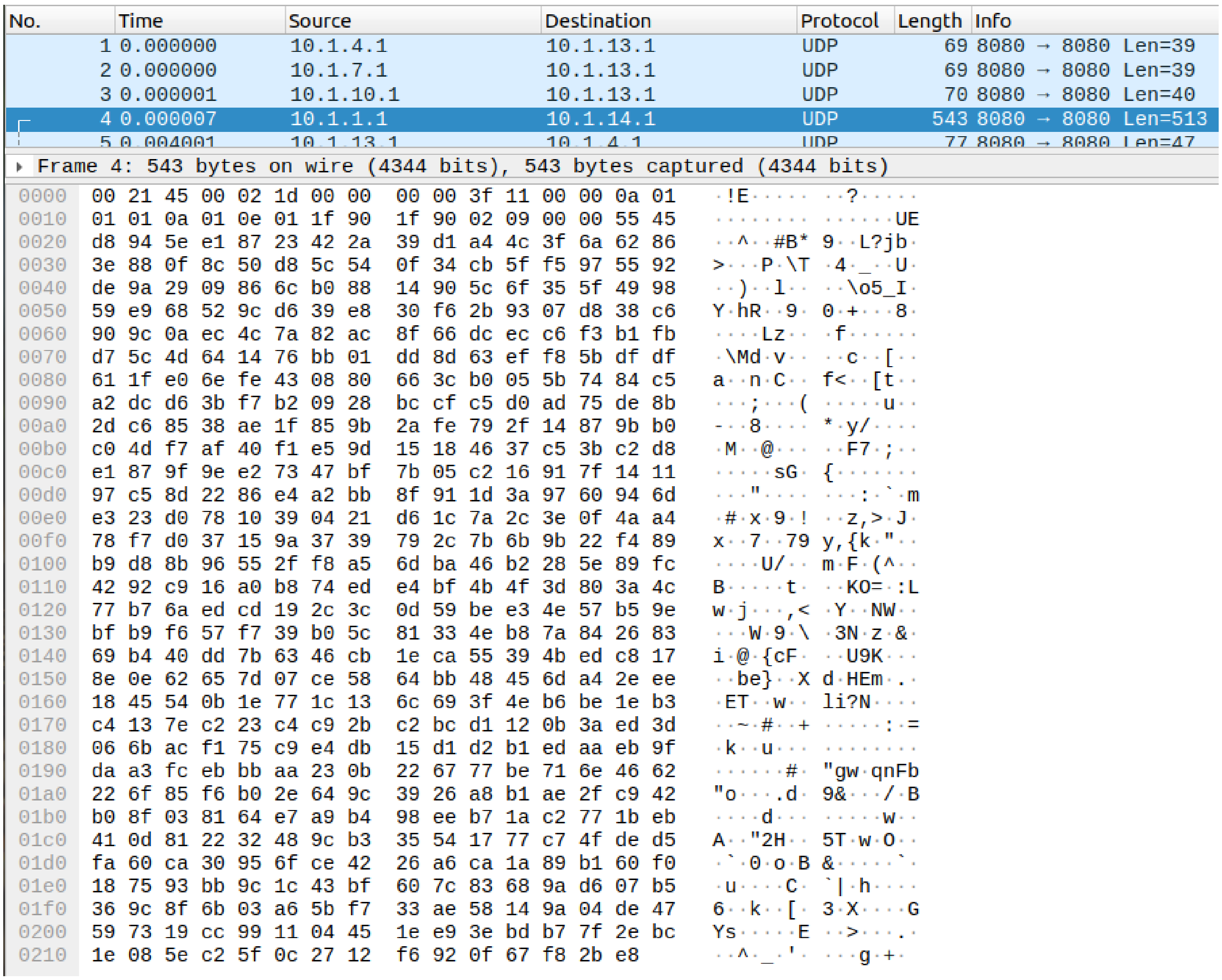

Figure 8: Message for Encrypted Edge Model in Wireshark

"This is a normal msg from client node 1." In return, the edge server will send a message back to the client that says "This is a return msg from 10.1.14.1 to 10.1.1.1." The same will happen for all the client and edge nodes in parallel. The messages can also be viewed using Wireshark if the data capture for the connections is enabled in the code. You can refer to Figures 7 and 8 shown in section 4.

4. RESULTS

As mentioned earlier, the command prompt displays the timestamps for each time a node sends or receives data from another node, and this is important when you want to find out how much latency is added when including RSA encryption and digital signatures. According to Figures 9 and 10, the encryption and digital signatures add a delay of about 0.00003

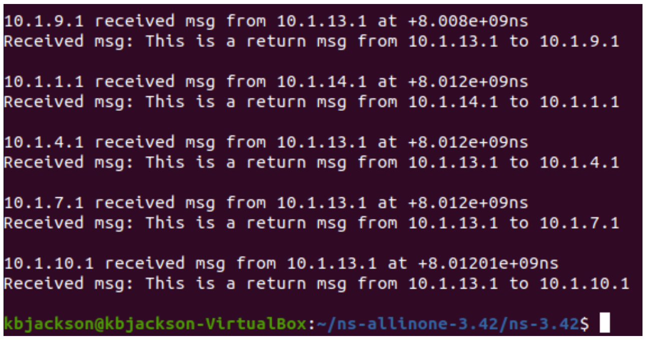

Figure 9: Output section without RSA

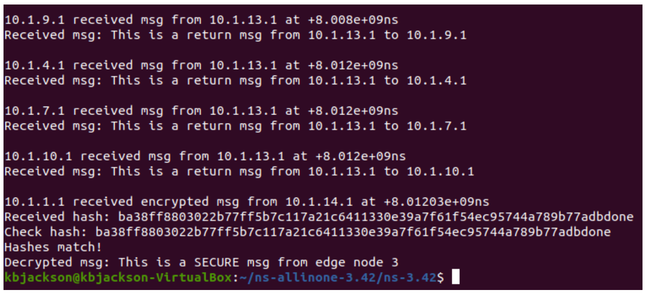

Figure 10: Output section with RSA

nanoseconds compared to the edge model without RSA. In Wireshark, when put side by side, it also shows how large the packet size of the encrypted and signed message is in Figure 7 compared to the packet size of a normal message in Figure 8, and you can see the scrambled text that would be impossible to decipher without the correct keys. With this comparison, it can be assumed that the increased packet size from using the RSA algorithms is a major factor in why there is that small delay difference.

5. CHALLENGES

5.1 Errors and Struggles

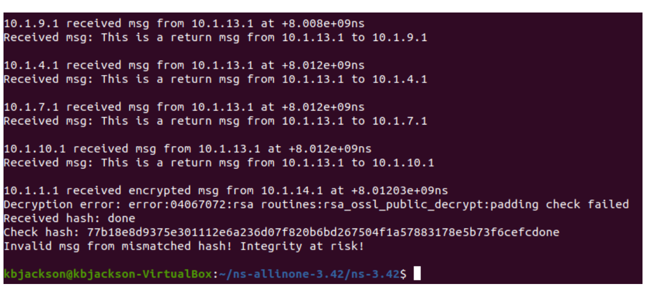

Figure 11: Output section of failed RSA

One major problem that was encountered is performing the proper key-exchange operation. It was attempted by including two separate functions that only send and receive the public keys respectively, and those would be the first functions to execute for the edge and client nodes using RSA. However, when sent, the packet containing the key failed to make it to its destination, so when sending "encrypted" messages to each other for the rest of the run, there is an error when the nodes attempt to decrypt them. This can be seen in the Wireshark screenshot in Figure 12, where we see the top part of the public key data belonging to edge server 12 with the 10.1.14.1 IP address, and the green message with a black background stating that the destination was "unreachable". The work-around for this was to simply swap the public keys when passing them into the node setup function for their sockets, though this would not be most accurate to a real key-exchange.

Another major problem was the handling of digital signatures. Despite Figure 10 showing the successful hash matching, there are also times where the hash decryption and the matching fails. This is because the digital signature was appended to the encrypted message with a colon character in-between them as the separator before it was sent to the receiver, and in the digital signature, similar to the encrypted text, it also contained colon characters, so when the receiving side separates the message and the signature, it does not separate at the correct colon, and both the digital signature and encrypted message would therefore fail at decryption and hash matching. A possible solution could be to put the encrypted message and digital signature into a structure before inserting it to a packet, but this was not attempted in time.

5.2 Potential Issues Moving Forward

Looking forward to the evolution of network security, there is a possibility that RSA key sizes of 2048 bits will no longer be strong enough minimum requirement for security, so we would have to bump up the bits requirement. Depending on how large the key size would have to be by then along with how evolved technology would be overall, this could raise a latency issue and potentially affect task efficiency of many devices using the edge. Another alternative is to use a different kind of asymmetric cryptography algorithm like elliptic curve cryptography (ECC). ECC uses smaller key sizes that offer about the same level of security as RSA, and the algorithm has been gaining popularity, but the main issue is that it is more complicated to implement compared to RSA, and if an error is made, it could hurt the security of the network using the algorithm [1].

6. CONCLUSION

For this project, we saw that using RSA cryptography could benefit the security and integrity of data in transit through an edge network if implemented correctly. With that said, there are a few things for this project that could still be further improved in the future in addition to the issues mentioned earlier in section 5.1, such as:

- programming a key rotation requirement that enforces key pair renewal after a certain amount of time to improve key management practices used here.
- including different kinds of attacks in the simulation to further test out the network security between the targeted communicating parties.
- including WiFi connections instead of only point-to-point connections to better represent today's real-life networks.

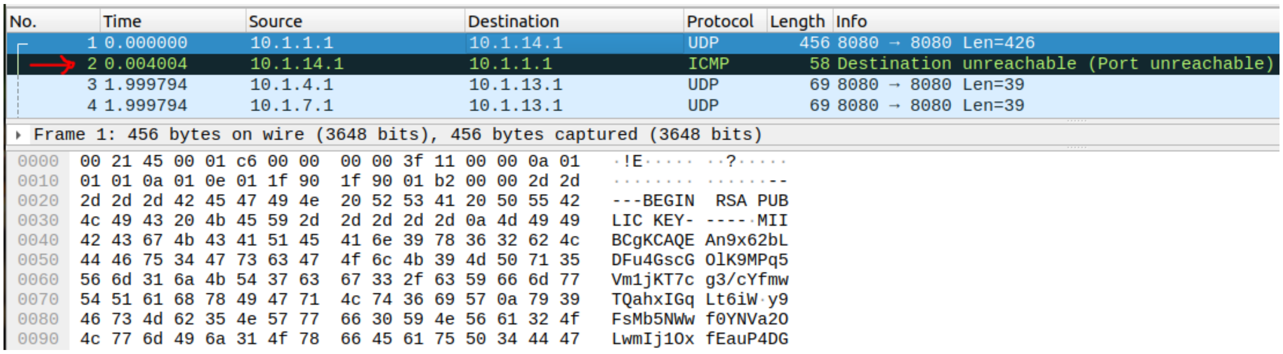

Figure 12: Failed public key exchange attempt

Computers and technology have come a long way since the very first electronic computer from 1946, and as technology evolved, resource demand becomes greater. With edge computing implemented, devices connected to the edge have greatly improve in performance and efficiency even today, but the security and integrity of potentially sensitive data that is transmitted throughout the edge network will always be a critical issue for the world to address as the edge is vulnerable to different kinds of cyberattacks without the use of some kind of cryptographic algorithm like RSA to secure communicating parties. As technology continues to evolve, so will security risks, and we must keep up to maintain strong network security.

REFERENCES

[1] “What are the differences between rsa, dsa, and ecc encryption algorithms?” *Sectigo*, 2021. [Online]. Available: https://www.sectigo.com/resource-library/rsa-vs-dsa-vs-ecc-encryption

[2] “Edge computing requires edge security: Best practices for protecting sensitive data at the edge,” *Cigent*, 2024. [Online]. Available: https://www.cigent.com/federal/blog/edge-computing-requires-edge-security-best-practices-for-protecting-sensitive-data-at-the-edge

[3] “Edge network,” *Heavy.AI*, 2024. [Online]. Available: https://www.heavy.ai/technical-glossary/edge-network

[4] “What is rsa? how does an rsa work?” *Encryption Consulting*, 2025. [Online]. Available: https://www.encryptionconsulting.com/education-center/what-is-rsa/

[5] “What is sha-256?” *Encryption Consulting*, 2025. [Online]. Available: https://www.encryptionconsulting.com/education-center/sha-256/

[6] E. K. Brockmeier, “The world’s first general purpose computer turns 75,” *Penn Today*, 2021. [Online]. Available: https://penntoday.upenn.edu/news/worlds-first-general-purpose-computer-turns-75

[7] S. Insights, “Edge computing examples,” *Scale Computing*, 2023. [Online]. Available: https://www.scalecomputing.com/resources/edge-computing-examples

[8] A. Kanungo, P. Mohanty, S. Mishra, R. Rathore, and B. Sh. Z. Abood, *Multi-level Security Threats Projection in Edge Computing*, 10 2024, pp. 173–185. [Online]. Available: https://www.researchgate.net/publication/384611054_Multi-level_Security_Threats_Projection_in_Edge_Computing

[9] P. Liu, “Public-key encryption secure against related randomness attacks for improved end-to-end security of cloud/edge computing,” *IEEE Access*, vol. 8, pp. 16 750–16 759, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8961999

[10] B. D. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn, and K. R. Walker, “Agile application-aware adaptation for mobility,” *SIGOPS Oper. Syst. Rev.*, vol. 31, no. 5, p. 276–287, Oct. 1997. [Online]. Available: https://doi.org/10.1145/269005.266708

[11] T. Shaikh, “From the fringe to the forefront: The evolution of edge cloud technology,” *World Business Outlook*, 2024. [Online]. Available: https://worldbusinessoutlook.com/from-the-fringe-to-the-forefront-the-evolution-of-edge-cloud-technology/

[12] S. Shawkat, “Enhancing steganography techniques in digital images,” Ph.D. dissertation, University of Samarra, 11 2016. [Online]. Available: https://www.researchgate.net/publication/328828460_Enhancing_Steganography_Techniques_in_Digital_Images

[13] B.-N. Tan, “Understanding rsa digital signatures,” *MalwarebytesLabs*, 2020. [Online]. Available: https://bntan.medium.com/understanding-rsa-digital-signatures-cfba3bc67428