Amirkabir University of Technology
(Tehran Polytechnic)
Faculty of Mathematics and Computer Science

Security Analysis in Distributed Information Systems and Implementation of an Encryption Algorithm for Data Protection

Narges Golmohamadi

Kiarash Esmaeilbeigi

Supervisor: Dr. mahsa saadat

july 2025

Table of Contents

Abstract

This paper analyzes security in distributed information systems (DIS). It outlines core security principles such as confidentiality, integrity, and availability, and identifies key vulnerabilities and threats specific to distributed architectures, including software flaws, configuration weaknesses, and attacks like Man-in-the-Middle and DDoS. To address these challenges, a layered security approach is proposed. The paper demonstrates these concepts through the implementation and testing of a custom RSA-based end-to-end encryption system. Experiments confirm that a message can be securely transmitted between nodes via a "blind router," proving that the encryption algorithm maintains confidentiality and prevents the intermediary from accessing sensitive data. The results validate the effectiveness of end-to-end encryption as a foundational security measure for distributed systems.

1 Chapter 1: Introduction

1.1 What Is a Distributed Information System?

In today's interconnected world, information systems are no longer confined to a single physical location but are instead distributed across various devices, networks, and platforms. A Distributed Information System (DIS) is a computing architecture where components such as data storage, processing units, applications, and user interfaces are located in different physical places but function together as a unified whole. These systems are designed to efficiently and securely share data, resources, and processes across geographically dispersed nodes. DIS forms the foundation of modern services like cloud computing, enterprise resource planning, distributed databases, and the Internet of Things.

At the core of a DIS lies a network of computers, or nodes, that communicate and collaborate to achieve a common objective. These nodes are connected through communication networks such as the internet or local area networks (LANs), with each node performing specific roles like data processing or storage. The architecture of the system dictates how responsibilities are divided among nodes. The primary motivation for distributing system components is to enhance scalability, availability, fault tolerance, and performance—especially in situations where centralized systems could become bottlenecks or points of failure.

The key architectural difference between centralized and distributed systems lies in how resources are managed. Centralized systems are generally easier to manage and secure due to their single control point but are vulnerable to complete failure if the central node goes down. In contrast, DIS is inherently more fault-tolerant and scalable; if one node fails, others can maintain system functionality. Distributed systems allow horizontal scaling by adding more nodes, unlike centralized systems that require vertical scaling, which is often costlier and hardware-limited. However, distribution introduces complexities such as synchronization, data consistency, and secure inter-node communication. Thus, while centralized systems are suited for simpler use cases, DIS is essential for large-scale, high-demand environments.

A defining feature of distributed information systems is transparency—users and applications should not be aware of the system's distributed nature. This includes location transparency (users don't need to know where resources are located), replication transparency (multiple data copies are hidden), and concurrency transparency (simultaneous access by multiple users without conflict). Achieving this requires careful system design and use of technologies such as synchronization protocols and consensus algorithms like Paxos or Raft. DIS architectures vary, including client-server, peer-to-peer, and multi-tier models, each with trade-offs in complexity, efficiency, and fault tolerance. As society relies more on digital infrastructure—from smart cities to cloud platforms—understanding the structure and challenges of DIS is increasingly vital for developers, researchers, and cybersecurity professionals.

1.2 Core Security Principles in Distributed Systems

Securing distributed systems hinges on upholding a set of fundamental principles that, while familiar from traditional security contexts, take on unique complexities in a distributed environment. At their core, these principles aim to protect the system and its data from a myriad of threats:
Confidentiality:
This is about keeping secrets safe. It means ensuring that sensitive information, whether it's data being transmitted between nodes or stored across various locations, is accessible only to authorized entities. Think of it as an encrypted message where only the intended recipient holds the key.
Integrity:
Integrity ensures the trustworthiness and accuracy of data. It's about guaranteeing that information remains unaltered and untampered with, both during its journey across the network and when it's at rest in storage. Any modification, whether accidental or malicious, should be detectable and, ideally, preventable.
Availability:
In distributed systems, availability is paramount. It means ensuring that services and resources are continuously accessible and operational for legitimate users and processes, especially for mission-critical or real-time functions. A distributed system that can't be accessed when needed is, by definition, failing its purpose.
Authentication:
Before any interaction, we need to verify who or what we're dealing with. Authentication is the process of confirming the identity of users, services, or other systems attempting to access resources within the distributed environment. This is often the first line of defense against unauthorized access.
Non-repudiation:
This principle ensures accountability. Non-repudiation guarantees that an entity cannot legitimately deny having performed a specific action or having made a particular commitment. It's crucial for auditing, legal compliance, and establishing trust within a distributed network where actions can originate from many different points.
Upholding these principles becomes significantly more challenging when dealing with diverse networks, multiple interconnected nodes, and often, components managed by different administrative authorities. The distributed nature introduces complexities in trust management, synchronized policy enforcement, and comprehensive threat monitoring that are less prevalent in centralized systems.

2 Chapter 2: Security in Distributed Information Systems

2.1 Security Vulnerabilities and threats in Distributed Systems

Distributed systems, while offering scalability and resilience, face a wide range of security vulnerabilities stemming from their interconnected, decentralized nature. These vulnerabilities can be categorized into four main types. Software-based vulnerabilities include flaws in code like buffer overflows, input validation issues (e.g., SQL injection), and logic errors, which can compromise multiple nodes once exploited. Configuration-based vulnerabilities arise from poor system setup, such as open ports, weak firewall rules, or missing encryption. These misconfigurations can leave entire systems exposed. Network-related vulnerabilities target the communication between nodes and include unencrypted data transfer, insecure protocols, and weak session management, all of which increase the risk of interception or manipulation. Lastly, human-centered vulnerabilities—including poor password practices, lack of awareness, and susceptibility to social engineering—can bypass even strong technical defenses.

The threats facing distributed systems can also be categorized by their sources and methods. External threats include hackers, cybercriminals, or state-sponsored attackers seeking financial gain, sabotage, or intelligence. Internal threats often come from disgruntled or careless employees who misuse access or unintentionally introduce vulnerabilities. In terms of method, passive threats like eavesdropping and traffic analysis aim to gather intelligence without altering system data, while active threats—including data tampering, message forgery, and Denial-of-Service attacks—disrupt or manipulate system operations directly.

Distributed architectures are particularly vulnerable to specific attacks that exploit their complexity and openness. For instance, Man-in-the-Middle (MitM) attacks allow adversaries to intercept or modify communication between nodes without detection, undermining confidentiality and integrity. Session hijacking involves taking control of legitimate user sessions, potentially enabling wide system access across distributed services. Similarly, replay attacks trick the system by resending legitimate data transmissions like authentication tokens, bypassing security checks. Distributed Denial-of-Service (DDoS) attacks flood systems with traffic from many sources, overwhelming resources and making services inaccessible.

Common web-based vulnerabilities become even more dangerous in distributed environments. SQL and command injection attacks target insecure input fields to gain unauthorized access or control over databases and infrastructure. Cross-Site Scripting (XSS) exploits client-side vulnerabilities to inject malicious scripts, which then affect other users. These attacks compromise data integrity and user trust, especially in distributed web applications. Meanwhile, phishing attacks continue to be effective by targeting users through deceptive messages to steal credentials or install malware, often serving as an entry point into the system.

Once inside the system, attackers often aim for privilege escalation and lateral movement. From an initially compromised node, they attempt to gain higher access rights and move across interconnected nodes, expanding control and reaching critical components. The interconnected design of distributed systems, while enabling efficiency and redundancy, also allows attackers to propagate their access quickly if not properly segmented and monitored. Defending against such threats requires a combination of strong technical safeguards, user education, secure coding practices, and proactive monitoring across all components of the system.

2.2 A Layered Approach to Security in Distributed Systems

Securing distributed information systems is not a one-time task but an ongoing, multi-faceted endeavor. A truly robust defense requires a layered security approach, where multiple security controls are implemented at different levels of the system, creating a formidable barrier against diverse threats. If one layer is breached, others are in place to detect and mitigate the attack.

Here are the key layers of this essential security framework:

- **System Hardening**
  - *Removing Unnecessary Services:* Disabling or uninstalling any unneeded software, applications, or network services that could serve as potential entry points for attackers.
  - *Applying Patches Regularly:* Promptly applying security patches and updates to operating systems, applications, and firmware to fix known vulnerabilities.
  - *Configuring Firewalls:* Implementing strict firewall rules at the host, network, and application levels to control inbound and outbound traffic.
  - *Enforcing Strict Access Controls:* Setting granular permissions on files, directories, and resources to ensure least privilege access.
- **Secure Communications**
  - *Authentication:* Using digital certificates, security tokens, and secure key exchanges to establish trusted identities.
  - *Encryption:*
    - *Data at Rest:* Encrypting databases, file systems, and storage volumes.
    - *Data in Transit:* Utilizing secure protocols such as TLS (e.g., HTTPS) and IPsec.
  - *Secure Routing Protocols and VPNs:* Employing secure routing and encrypted tunnels to protect data across public networks.
- **Intrusion Detection and Prevention Systems (IDPS)**
  - *Monitoring Traffic:* Continuously analyzing traffic and logs for threats.

  - *Detecting Anomalies:* Using behavioral analysis and machine learning to detect unknown threats.
  - *Responding to Threats in Real-Time:* Blocking, quarantining, or alerting administrators to limit damage.

- **Access Management**
  - *Multi-Factor Authentication (MFA):* Combining multiple verification factors to secure access.
  - *Access Control Models:*
    - *Role-Based Access Control (RBAC):* Permissions based on user roles.
    - *Attribute-Based Access Control (ABAC):* Permissions based on user/resource/environment attributes.

- **Monitoring and Auditing**
  - *Implementing Audit Logs:* Recording all significant system and user events.
  - *Monitoring Systems:* Using SIEMs for centralized, real-time log analysis.
  - *Forensic Investigations:* Providing evidence to reconstruct and analyze security incidents.

By meticulously implementing and continuously refining these layered security measures, organizations can significantly enhance the resilience and trustworthiness of their distributed information systems against the ever-evolving threat landscape.

3 Chapter 3: Implementation of an End-to-End Encryption system

To demonstrate the principles of secure communication in a distributed system, a custom implementation of an RSA-based encryption algorithm was developed. This model simulates a distributed network with two communicating nodes ("alpha" and "beta") and a central "blind router" or observer. The core of this implementation is the use of public-key cryptography to ensure end-to-end encryption, meaning that only the intended recipient can decrypt and read a message.

3.1 RSA Key Generation

Each node in the system is responsible for generating its own public and private key pair. The process is as follows:

- **Prime Number Generation**: Each node first generates two large, distinct prime numbers, p and q, using a primality test. The `generate_prime` function is used for this purpose, which in turn relies on the `is_prime` function (a Miller-Rabin primality test) to ensure the numbers are indeed prime. The key size is set to 256 bits, which is a significant factor in the security of the encryption.
- **Modulus Calculation**: The modulus n is calculated by multiplying these two prime numbers: $n = p * q$.
- **Totient Calculation**: Euler's totient function, $\phi(n)$, is calculated as $\phi(n) = (p - 1) * (q - 1)$.
- **Public and Private Keys**: A public exponent e is chosen (in this case, the fixed value 65537). The private key exponent d is then computed as the modular multiplicative inverse of e with respect to $\phi(n)$, a process handled by the `mod_inverse` and `extended_gcd` functions. The public key is the pair (n, e), and the private key is the pair (n, d).

3.2 The Communication Flow

The system's architecture includes `Node` objects (e.g., "alpha" and "beta") and an `observer` object that acts as a central registry and router. The communication process, secured by the RSA algorithm, unfolds in these steps:

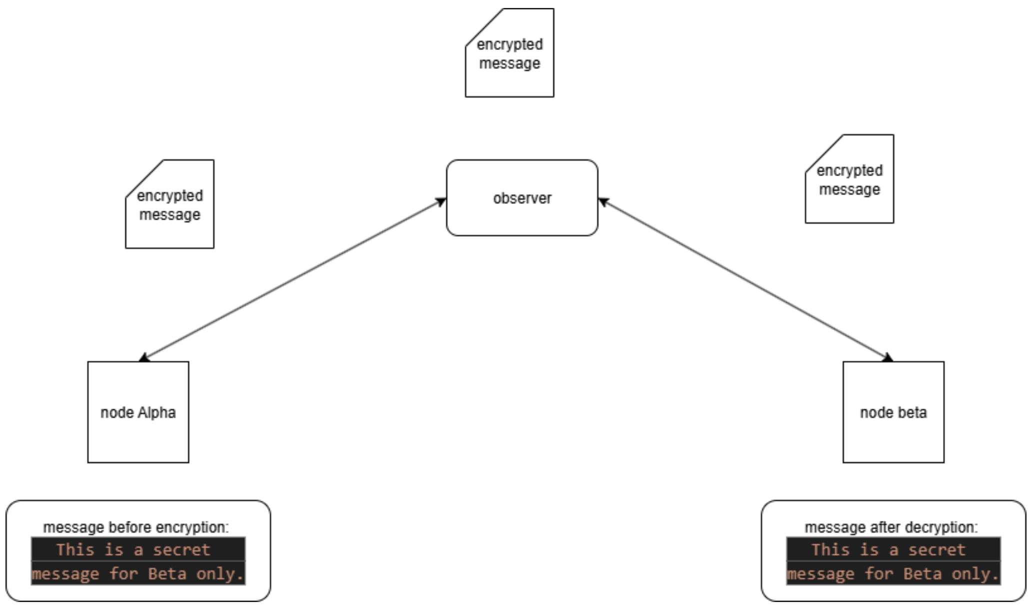
1. **Registration**: When a `Node` is initialized, it registers itself with the `observer`, providing its unique `node_id` and its public key. The observer stores this information in a directory but has no access to the node's private key.
2. **Message Sending (Encryption)**: To send a message, a node first requests the recipient's public key from the observer. Once it has the recipient's public key (n, e), the sender converts the plaintext message into an integer. It then encrypts this integer using the recipient's public key with the following operation:
$$encrypted\_message = (message\_int)^e \mod n$$

3. **Message Routing**: The sending node forwards the encrypted message to the observer, along with the recipient's `node_id`. The observer, acting as a "blind router," receives this opaque packet. Critically, because the observer only has public keys and the message is encrypted with the recipient's public key, it cannot decrypt or read the content of the message. Its sole function is to look up the recipient's node object and forward the encrypted data.

4. **Message Reception (Decryption)**: The recipient node receives the encrypted message from the observer. It then uses its own unique private key (n, d) to decrypt the message:

$$decrypted\_message = (encrypted\_message)^d \mod n$$

The decrypted integer is then converted back into the original plaintext message. This ensures that the message remains confidential throughout its journey across the network and is only accessible to the intended recipient. This implementation effectively demonstrates how end-to-end encryption, using a manually created RSA scheme, can secure data in a distributed environment, preventing a central intermediary (the observer) from accessing sensitive information while still allowing it to perform its routing function.



```
This is a secret
message for Beta only.
```

```
This is a secret
message for Beta only.
```

4 Results and Experiments

4.1 Objective of the Experiment

The primary objective of the experimental phase was to validate the core functionality and security of the custom RSA-based encryption algorithm within a simulated distributed system. The experiments were designed to demonstrate the successful and secure flow of a message from a sender to a recipient. Specifically, the experiments aimed to:

- Verify that a plaintext message could be successfully converted into an encrypted format using the recipient's public key.
- Confirm that the intermediary observer, acting as a blind router, could not access the original plaintext of the message, thereby demonstrating the principle of confidentiality.
- Ensure that the intended recipient could successfully decrypt the message using their private key to recover the original plaintext.
- Demonstrate that the system could handle a standard, non-trivial message and correctly decode it without corruption.

4.2 Experimental Setup

The experiments were conducted using the Python code from `RSA.py` in a controlled simulation environment. The setup consisted of:

- A central `observer` instance responsible for routing messages.
- Two `Node` instances, "alpha" and "beta", each with a unique public/private key pair.
- A message, "This is a secret message for Beta only.", was chosen to represent a typical communication payload.

The code's output logs were used as the primary source of data to trace the message's journey and confirm its state (plaintext or encrypted) at each point in the system.

4.3 Experimental Results

4.3.1 Successful End-to-End Communication

The experiment successfully simulated the entire end-to-end communication flow, from message creation to final reception and decryption.

1. **Message Encoding and Encryption**: The sending node, "alpha", successfully encoded the message string into an integer format. It then encrypted this integer using the public key of the recipient, "beta". The output log from the sender confirmed that the message was prepared for transmission in an encrypted state. The 'Node' class handles the conversion of the message string to a large integer and then performs the modular exponentiation for encryption.

```
[Node alpha] preparing a message for 'beta'.
[Node alpha] sending:
'This is a secret message for Beta only.'
[Node alpha] recipient public key:
(10591794858363226607843203309526З
1553446343225091502053414342150671
7717800423645529469792739504171865
6173371110823458853890321039333068
7948244355531672568З, 65537)

[Node alpha] private key:

(8148399447800139119107827844548 71
8661868010506218509825909075200323
1689264280689682800694736040087980
1128750207211494808248219385023372
0463009009177831107,15531654775458
0371203892435470196367737392903617
3178887426280839868120698673072240
9423970425234581311023952150599020
6729229320965441702665212353568016
4993)

[Node alpha] encrypted message using 'beta's public key.
```

2. **Blind Routing via observer**: The encrypted integer was sent to the `observer`. The observer's log output showed that it received an "opaque packet" with an integer value, which it could not read. This is a critical result, as it confirms that the observer performs its routing function without compromising the confidentiality of the message content. This demonstrates a key security principle for distributed systems.

```
[Observer] received an encrypted packet
from 'alpha' for 'beta'.
[Observer] content of message:
87461553353432103638948761327 33
9537474134546556465839284128878
33814277564845209077317334867 4
7477357471047324549325979256 93
39446084607638639115770297940132
```

3. **Message Decryption and Decoding**: The recipient node, "beta", received the encrypted integer. Using its own private key, it successfully performed the decryption operation, converting the encrypted integer back into its original integer form. Finally, the system decoded this integer back into the original plaintext message string. The output log from the receiver, "beta", explicitly showed the successful recovery of the original message.

```
[Node beta] received an encrypted packet from ’alpha’.
[Node beta] private key:
(1059179485836322660784320330952631553446343225091502
0
53414342150671771780042364552946979273950417186561733
7
111082345885389032103933306879482443555316725683,482357
6137795579006327543921389786127555359552021143825046074
0999663854840996379505365916899795374450438807931710533
95762228382810481394703823666678584497)
[Node beta] decrypted message:
’This is a secret message for Beta only.’
```

5 Conclusions

5.1 Overview

This paper has explored the critical aspects of security in distributed information systems, highlighting their unique vulnerabilities and the threats they face. The distributed nature of these systems, while offering advantages in scalability and fault tolerance, introduces complexities such as securing multiple nodes, managing trust, and ensuring consistent policy enforcement. We have established that a robust security posture for a distributed system relies on upholding core security principles like confidentiality, integrity, and availability. To achieve this, a layered security approach is essential, encompassing system hardening, secure communication protocols, intrusion detection, access management, and continuous monitoring.

The practical application of these principles was demonstrated through the implementation and experimental validation of a custom RSA-based end-to-end encryption system. The experiments successfully verified that a message could be securely transmitted between two nodes (”alpha” and ”beta”) via a central ”blind router” or proxy. The results confirmed that the message remained confidential and unintelligible to the proxy, proving the effectiveness of the encryption algorithm in a simulated distributed environment. This successful implementation provides a tangible example of how cryptographic techniques can form a fundamental layer of data protection, ensuring that only the intended recipient can access sensitive information.

5.2 Future perspectives

The work presented here serves as a foundation for further research and development. One area for future exploration is the optimization of the RSA key generation process for even greater efficiency and security. While the current implementation uses a 256-bit key size, which is a significant factor in security, future work could explore larger key sizes and the performance trade-offs involved. Another key area is the integration of more sophisticated security measures beyond basic encryption. This could include the development of more complex and scalable access control models suitable for large-scale distributed systems, such as advanced Attribute-Based Access Control (ABAC) frameworks. Furthermore, implementing and testing the system with various consensus algorithms, like Paxos or Raft, would provide insights into maintaining data integrity and coordination in the presence of node failures, a crucial challenge in distributed systems. Finally, exploring the use of alternative, post-quantum cryptographic algorithms could prepare the system for future security challenges, ensuring long-term data protection in the face of evolving threats.

6 bibliography

Tanenbaum, A. S., Van Steen, M. (2016). Distributed Systems: Principles and Paradigms. Pearson.

Coulouris, G., Dollimore, J., Kindberg, T., Blair, G. (2011). Distributed Systems: Concepts and Design. Pearson.

Stallings, W. (2017). Cryptography and Network Security: Principles and Practice. Pearson.

Bishop, M. (2003). Computer Security: Art and Science. Addison-Wesley.

Lamport, L. (1998). The Part-Time Parliament. ACM Transactions on Computer Systems, 16(2), 133–169.

Ongaro, D., Ousterhout, J. (2014). In Search of an Understandable Consensus Algorithm. USENIX Annual Technical Conference.

NIST Special Publication, Guide to General Server Security
https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-123.pdf

OWASP Developer Guide, covering secure coding practices and vulnerabilities
https://owasp.org/www-project-developer-guide/assets/exports/OWASP$_{D}eveloper_{G}uide.pdf$

Cisco White Paper on Layered Security for Service Provider Networks
https://www.cisco.com/c/dam/en/us/solutions/service-provider/pdfs/rural-broadband/white-paper-sp-security.pdf

Cisco Industrial Cybersecurity for Distributed Field Networks Solution Brief
https://www.cisco.com/c/en/us/products/collateral/security/industrial-security/industrial-cybersecurity-field-networks-sb.html

GeeksforGeeks - RSA Algorithm in Cryptography
https://www.geeksforgeeks.org/computer-networks/rsa-algorithm-cryptography/

GeeksforGeeks - Encryption in Distributed Systems
https://www.geeksforgeeks.org/system-design/encryption-in-distributed-systems/

Understanding End-to-End Encryption Guide by BlueGoatCyber
https://bluegoatcyber.com/blog/understanding-end-to-end-encryption-a-beginners-guide/