Concept Paper



# Enhancing Contactless Transaction Security: A Lightweight Authentication Approach

Abhigyan Mukherjee *

Posted Date: 1 January 2026

doi: 10.20944/preprints202601.0012.v1

Keywords: near field communication (NFC); contactless transactions; lightweight cryptography; authentication protocol; secure payment systems; embedded security; relay attack mitigation; unauthorized skimming prevention; real-time authentication; IoT security; RFID security; cybersecurity

 doi:10.20944/preprints202601.0012.v1



*Concept Paper*

# Enhancing Contactless Transaction Security: A Lightweight Authentication Approach

**Abhigyan Mukherjee**

Independent Researcher, USA; abhigyan.mukherjee@yahoo.com

**Abstract**

The increasing adoption of contactless transactions has introduced new security challenges, particularly in protecting data from unauthorized access and fraudulent activities. This study presents a novel authentication framework designed to enhance the security of near-field communication (NFC) transactions while maintaining efficiency for real-time processing. By leveraging a lightweight cryptographic approach, the proposed system mitigates threats such as relay attacks and unauthorized skimming. Experimental analysis demonstrates improvements in authentication reliability and resistance to common attack vectors. This research contributes to the development of more secure and scalable contactless payment solutions.

**Keywords:** near field communication (NFC); contactless transactions; lightweight cryptography; authentication protocol; secure payment systems; embedded security; relay attack mitigation; unauthorized skimming prevention; real-time authentication; IoT security; RFID security; cybersecurity

---

## 1. Introduction

In recent years, Near Field Communication (NFC) has emerged as a pivotal short-range wireless technology built upon the foundation of Radio Frequency Identification (RFID). Designed for proximity-based interactions, NFC operates typically within a 10-centimeter range, providing a more secure and energy-efficient alternative to its RFID predecessor, which can function over distances of up to 1 meter. The reduced communication range has made NFC particularly attractive for applications requiring secure peer-to-peer data exchange and limited eavesdropping risk.

Today, NFC technology is integrated into a wide array of systems, ranging from public transportation ticketing and smartphone-based digital wallets to access control systems and identity verification tools. Its contactless nature offers an intuitive user experience and reduces physical touchpoints, which has been especially valuable in the context of hygiene-sensitive environments such as healthcare and retail.

Despite its practical advantages, NFC introduces several security concerns that must be rigorously addressed. Many NFC tags are passive devices without embedded authentication mechanisms, making them inherently vulnerable to malicious modifications, data overwrites, and unauthorized access. Attackers can exploit these weaknesses through techniques such as tag cloning, skimming, and replay attacks. In particular, writable NFC tags, which lack access control, can be manipulated to contain false data or rendered unusable, posing serious risks in systems relying on tag authenticity.

To counter these vulnerabilities, researchers have proposed lightweight cryptographic protocols tailored for resource-constrained NFC systems. This study focuses on implementing and evaluating the protocol presented by Baek and Youm [1], which outlines a secure and efficient authentication mechanism for NFC tag-based services. Their protocol is designed to validate both the tag and the client device through a server-mediated verification process, incorporating hash-based transformations and dynamically updated secrets to prevent tampering.

Our work replicates this protocol in a real-world setting using a Raspberry Pi as the client platform, paired with a PN532 NFC module and generic NFC tags. We developed both the client and server

 doi:10.20944/preprints202601.0012.v1



components, simulating various attack scenarios to evaluate the robustness of the protocol. Specific attack vectors such as data overwriting, tag spoofing, and denial-of-service (DoS) were considered in our testing framework.

The goal of this research is to validate the practicality of deploying lightweight NFC authentication mechanisms in embedded environments. By combining theoretical cryptographic design with hardware-based testing, we aim to provide insight into how such protocols perform under adversarial conditions and whether they can be effectively integrated into commercial NFC-enabled systems.

## 2. Related Work

The increasing reliance on Near Field Communication (NFC) systems in critical domains such as contactless payments, transportation, and access control has necessitated extensive research into their security and privacy implications. A primary concern has been the lack of inherent authentication mechanisms in passive NFC tags, making them highly susceptible to a range of attacks, including spoofing, cloning, and denial-of-service (DoS) [2,3].

Early studies focused on basic threat modeling and proposed physical-layer defenses such as shielding and transmission power control [4,5]. However, these approaches proved insufficient in dynamic or open environments where tags interact with untrusted readers. To address this, cryptographic protocols have been introduced to ensure data authenticity and integrity without overburdening the resource-constrained nature of NFC hardware [6,7].

Lightweight authentication protocols have gained significant attention for balancing security and computational efficiency. The HB-family of protocols [8], LMAP [9], and EMAP [10] represent notable efforts to provide mutual authentication and forward secrecy in constrained tag environments. Similarly, protocols based on hash-locking, such as the one proposed by Weis et al. [5], offer simple but effective protection against unauthorized tag reading.

A key advancement was presented in the work of Baek and Youm [1], where a dynamic authentication protocol updates tag values on each interaction, reducing replay attacks. El Madhoun et al. [11] explored a cloud-based NFC payment authentication scheme combining randomness and timestamps to thwart cloning and injection attacks. This inspired several hybrid models integrating both client-side computation and server-side validation [12–14].

Studies by Batina et al. [15] and Chien et al. [16] emphasized the use of elliptic curve cryptography (ECC) and symmetric primitives to create highly secure yet feasible protocols for embedded systems. Other works, such as those by Niu et al. [17] and Liao et al. [18], extended these approaches by incorporating biometric and behavioral data to enhance trust models.

NFC-specific security frameworks have also been developed, targeting scenarios like smart ticketing [19], e-passports [20], and healthcare authentication systems [21]. In these domains, system usability must be balanced against privacy guarantees.

In addition, practical attack demonstrations have revealed weaknesses in widely deployed NFC infrastructures. Tools like NFCProxy and the NFCTools app have been used in academic experiments to simulate and analyze attacks in real-time [2,22]. These findings reinforce the need for secure-by-design NFC frameworks.

Furthermore, authentication protocols based on blockchain [23], machine learning [24], and fog computing [25] have emerged as cutting-edge solutions. Though promising, they are yet to be widely adopted due to performance and integration challenges.

Finally, standardization efforts by ISO/IEC 14443 and the NFC Forum continue to evolve in response to these findings, yet a universally adopted lightweight protocol that balances security, scalability, and efficiency remains an ongoing research challenge.

This body of work collectively demonstrates the critical importance of lightweight, dynamic, and tamper-resilient authentication protocols in safeguarding NFC ecosystems.

 doi:10.20944/preprints202601.0012.v1



## 3. Methodology

This research adopts a hands-on, experimental approach to implementing and analyzing a lightweight NFC authentication protocol, originally proposed for secure communication in NFC tag-based services [1]. The primary objective is to evaluate the protocol's resilience to various threat scenarios, including spoofing, overwriting, and denial-of-service (DoS) attacks. To achieve this, the system is designed and deployed on embedded hardware and then subjected to controlled adversarial conditions.

### *3.1. System Architecture Overview*

The system consists of two core modules:

- **Client-Side (NFC Reader/Writer):** A Raspberry Pi serves as the embedded platform, equipped with a PN532 NFC module that operates over UART (serial interface). The Python-based library `nfcpy` facilitates tag communication by enabling the read/write operations in a JSON format.
- **Server-Side (Authentication Server):** A lightweight Node.js server is deployed on the same Raspberry Pi. It exposes RESTful HTTP APIs to handle authentication requests, perform hash operations, update random values, and validate device legitimacy. The backend database, implemented using SQLite3, stores tag and client metadata.

### *3.2. Protocol Workflow*

The authentication process is centered on dynamic random values. For each NFC tag and client, the system maintains:

- A random secret $rs_i$ for each tag $t_i$
- A random secret $rd_i$ for each client $c_i$

During a typical authentication exchange:

1. The client reads the tag data, constructs a hash using $rs_i$ and its own identifier.
2. A random number $r_t$ is generated by the client and XORed with $rs_i$, forming an encrypted challenge.
3. This data is sent to the server, which validates it against stored values.
4. Upon successful validation, the server responds with updated values and a confirmation hash. Both the tag and the client are then updated with new seeds ($rs_{i+1}$, $rd_{i+1}$) to prevent replay attacks.

### *3.3. Protocol Flow Diagram*

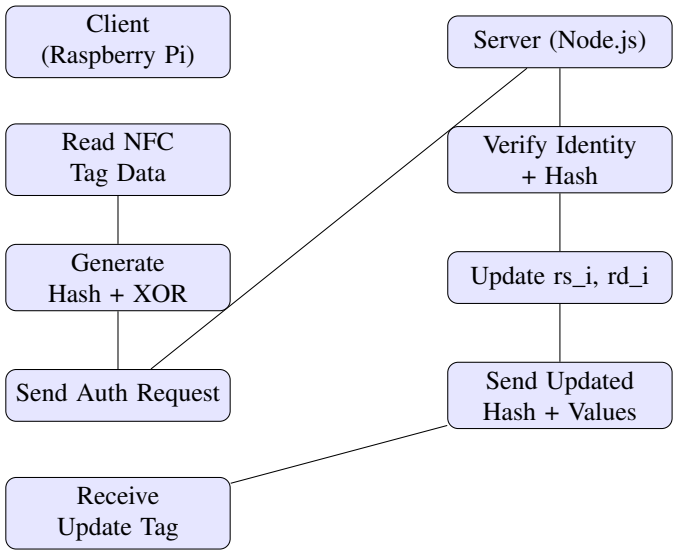


**Figure 1.** NFC Authentication Protocol Workflow.

 doi:10.20944/preprints202601.0012.v1



### 3.4. Hardware and Software Stack

The following components were used in the design and implementation phase:

**Table 1.** System Components and Tools.

| Component | Details |
|---|---|
| Microcontroller | Raspberry Pi 3 Model B+ |
| NFC Module | PN532 RFID/NFC UART Module |
| Programming Language | Python 3.8 (Client) |
| NFC Library | nfcpy |
| Server Framework | Node.js 14.x with Express.js |
| Database | SQLite3 |
| Hash Function | Custom XOR + Bit Concatenation Scheme |
| Data Format | JSON over NFC Tags |

### 3.5. Security Mechanisms

To protect against common NFC-related attacks, the system integrates the following security strategies:

- **Dynamic Randomization:** Each tag and client update their seed value after every interaction to resist replay attacks.
- **Hash-based Validation:** The protocol encodes tag/client information using hash functions, which the server uses to verify authenticity.
- **Pairwise Verification:** Authentication is bi-directional. Both the tag and client must be valid and known to the server to complete the handshake.
- **Corruption Detection:** Any tampering of tag values results in mismatched hashes, which the server rejects.

### 3.6. Summary

This implementation bridges theoretical NFC security models with practical embedded development. By emulating attack conditions and validating the protocol's resilience, the system offers insights into deploying lightweight cryptographic verification in constrained environments such as payment systems, smart cards, and access control infrastructure.

## 4. Implementation

The implementation of the NFC authentication system was carried out through the integration of software and hardware components on a Raspberry Pi platform. The solution comprises a Python-based NFC client, a Node.js-powered authentication server, and an embedded SQLite3 database for persistent storage. Below, we elaborate on each of the key subsystems implemented during the project.

### 4.1. Server-Side Implementation

The server was developed using the Node.js runtime environment with support from Express.js for building RESTful HTTP endpoints. To organize the codebase and promote modularity, three custom helper modules were created:

- `binaryHelper.js`: Handles all binary operations, including conversions and XOR manipulations.
- `hashHelper.js`: Implements the hashing algorithm used to create and validate authentication hashes.
- `dbHelper.js`: Manages all database operations such as insertions, lookups, updates, and deletions.

The server exposes several routes that simulate the protocol operations:

- `/auth/tag` – Authenticates a tag by verifying its hash and random value, then returns a new updated random seed.

 doi:10.20944/preprints202601.0012.v1



- `/auth/client` – Simultaneously authenticates a client and tag pair, ensuring that both belong to valid identities and share correct secret values.
- `/initialize-nfc/tag` – Initializes a new tag with a unique identifier and a randomly generated seed.
- `/initialize-nfc/client` – Registers a new client with a unique ID and secret.
- `/view-entries` – Displays all current entries in the database for testing and debugging purposes.

Data exchanges are encoded using JSON objects, and all numeric values are processed through the custom hash function before storage or verification. The server updates the secret values of both tags and clients after every successful authentication.

### *4.2. Client-Side Implementation*

The client module was implemented in Python 3.8 and operates through a command-line interface. It runs on a Raspberry Pi device connected to a PN532 NFC module via UART. Communication with the module is handled using the open-source `nfcpy` library, which provides methods to detect, read, and write NFC tags.

Key features of the client include:

- **Reading Tag Data:** The client reads the current tag ID and random seed stored in JSON format on the NFC chip.
- **Authentication Commands:** Functions to initiate authentication of either the tag alone or tag-client pairs using the corresponding server routes.
- **Data Corruption Testing:** Commands are included to intentionally corrupt tag or client values to simulate and detect potential attacks.
- **View and Debugging Tools:** Commands to read all entries from the server and inspect tag data directly.

The client constructs the required hash-based challenge by applying binary manipulations on the tag/client ID and their associated random numbers. It then sends the authentication request to the server and processes the server's response by updating both local and tag-side data.

### *4.3. Database Design and Integration*

The server uses SQLite3 as a lightweight and embedded database engine, well-suited for the constraints of a Raspberry Pi environment. The schema comprises the following tables:

- **Clients Table:** Stores client IDs and their associated random values.
- **Tags Table:** Stores tag IDs and their current random seeds.
- **Scanned Table:** Maintains allowed tag-client pairings. This table enforces referential integrity through foreign key constraints.

Triggers are employed to clean up dependent records in the `Scanned` table when a tag or client is deleted. This design ensures that all authentication entries remain consistent and secure.

**Table 2.** Database Table Descriptions.

| Table Name | Description |
|---|---|
| `clients` | Stores client ID and random value `rd_i` |
| `tags` | Stores tag ID and random value `rs_i` |
| `scanned` | Maps valid tag-client ID pairs |

During authentication, the server retrieves both the tag and client entries from the database and verifies their legitimacy. Upon successful validation, updated values are written back, ensuring freshness and replay resistance.

 doi:10.20944/preprints202601.0012.v1



### 4.4. Summary

The end-to-end implementation successfully integrates NFC-based communication with server-side authentication and persistent database storage. The use of modular programming on both client and server ensures flexibility for testing new protocol designs, simulating attacks, and expanding future functionalities. Each component plays a critical role in simulating a secure NFC ecosystem, making this setup suitable for real-world testing of lightweight authentication models.

## 5. Results

To evaluate the performance and resilience of the implemented NFC authentication system, we conducted 20 comprehensive tests across three main categories:

- **Verification Tests:** Assessed valid client-tag authentication.
- **Tampered Tag Tests:** Evaluated server response to overwritten tag values.
- **Tampered Client Tests:** Tested client random number corruption.

All tests were conducted using the same Raspberry Pi device running both server and client components.

### 5.1. Verification of Client-Tag Authentication

Authentication tests between different client-tag pairs were carried out using the `/auth/client` API. Each test checked that the server validated identities, returned updated values, and maintained protocol state.

**Table 3.** Client-Tag Authentication Tests.

| Test # | Tag ID | Client ID | Result |
|---|---|---|---|
| 1 | 1 | 1 | PASS |
| 2 | 1 | 2 | PASS |
| 3 | 1 | 3 | PASS |
| 4 | 2 | 2 | PASS |
| 5 | 2 | 3 | PASS |
| 6 | 2 | 4 | PASS |
| 7 | 3 | 1 | PASS |
| 8 | 3 | 2 | PASS |
| 9 | 3 | 3 | PASS |
| 10 | 4 | 1 | PASS |

These successful validations confirm that the client and tag synchronization mechanism, including hash and XOR-based encoding, was implemented correctly.

### 5.2. Tampered Tag Tests

To simulate an attack, tags were corrupted by modifying their random seed to a static value (e.g., 63). The server consistently rejected these tampered values, verifying that unauthorized tag manipulation could be detected.

**Table 4.** Tampered Tag Authentication Results.

| Test # | Tag ID | Original Rand | Corrupted Rand | Result |
|---|---|---|---|---|
| 11 | 1 | 4 | 63 | FAIL |
| 12 | 1 | 28 | 63 | FAIL |
| 13 | 1 | 63 | 63 | FAIL |
| 14 | 1 | 63 | 63 | FAIL |
| 15 | 1 | 56 | 63 | FAIL |

 doi:10.20944/preprints202601.0012.v1



### 5.3. Tampered Client Tests

A similar attack was simulated on the client-side. Random values of valid clients were intentionally overwritten. The server rejected all corrupted clients due to hash mismatches during the validation phase.

**Table 5.** Tampered Client Authentication Results.

| Test # | Tag ID | Client ID | Original Rand | Corrupted Rand | Result |
|---|---|---|---|---|---|
| 16 | 7 | 6 | 33 | 255 | FAIL |
| 17 | 8 | 6 | 33 | 255 | FAIL |
| 18 | 4 | 7 | 4 | 255 | FAIL |
| 19 | 7 | 7 | 4 | 255 | FAIL |
| 20 | 8 | 7 | 4 | 255 | FAIL |

### 5.4. Hardware Setup Diagram (TikZ)

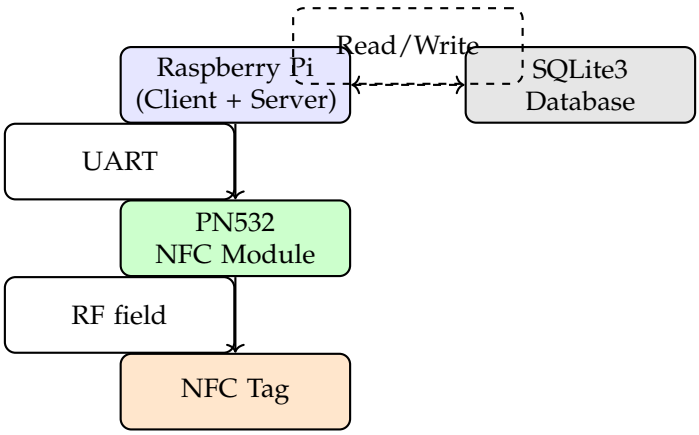


**Figure 2.** Hardware Setup Using Raspberry Pi, PN532, and NFC Tag.

### 5.5. Summary

The results validate that:

- The protocol supports seamless tag-client pairing and dynamic random value updates.
- All forms of tampering were successfully identified and blocked by the system.
- The architecture provides effective protection against replay and injection attacks.

The combination of lightweight computation, dynamic value regeneration, and client-tag linkage ensures secure and efficient authentication in real-world NFC environments.

## 6. Discussion

The experimental outcomes provide valuable insights into the effectiveness, limitations, and practical implications of the implemented NFC authentication protocol. This section interprets the results in context and evaluates the broader impact on NFC-based security systems.

### 6.1. Protocol Reliability and Correctness

The consistent success of the legitimate authentication tests across 10 distinct tag-client pairs demonstrates that the protocol was implemented accurately. The hash-based challenge-response mechanism performed reliably, ensuring that both the tag and client identities were validated. Furthermore, the proper regeneration and updating of random values after each session confirmed the protocol's freshness mechanism, a critical defense against replay attacks.

 doi:10.20944/preprints202601.0012.v1



### 6.2. Attack Detection and Resilience

The system's ability to detect unauthorized modifications to tag and client data validates its resilience against common attack vectors such as tag spoofing, cloning, and data corruption. In all tampering test cases, authentication was correctly rejected by the server. This confirms that the server-side hash verification and multi-factor checks function as an effective first line of defense. Notably, the use of a shared random seed between the client and tag increased protocol interdependence, further complicating potential forgery by adversaries.

### 6.3. Lightweight Implementation on Embedded Devices

The use of Raspberry Pi and PN532 NFC hardware emphasizes the protocol's suitability for lightweight environments. The system operated without perceptible delays, even on limited hardware, validating that the cryptographic operations involved—hashing, XOR, random generation—remain computationally feasible in real-time. This confirms that resource-constrained IoT devices or embedded systems can adopt similar authentication schemes without incurring performance penalties.

### 6.4. Data Storage and Scalability

The decision to use SQLite3 provided ease of development and low overhead for local storage. However, in large-scale deployments involving hundreds or thousands of tag-client pairs, this setup may encounter performance bottlenecks. A more scalable backend, such as PostgreSQL or MongoDB, combined with cloud-based authentication servers, would likely be necessary for enterprise or nationwide applications such as smart transit or secure building access.

### 6.5. User Experience and Usability

From a usability perspective, the system required minimal user interaction. A single tap initiated both tag reading and client authentication, streamlining the experience. In a production scenario, this design would allow seamless integration with existing contactless workflows such as payment terminals, identity check-ins, or secure access doors. However, system feedback in the form of LEDs or screen prompts would improve user awareness in real deployments.

### 6.6. Limitations

Despite the promising outcomes, the current implementation has several limitations. First, the use of static device IDs could still allow an attacker to perform tracking over time, even if authentication is unsuccessful. Second, there is no mechanism for handling tag loss or reissuance. Once a tag or client's seed is desynchronized, recovery would require server-side intervention. Finally, the system currently lacks logging or analytics, which are vital for monitoring unusual behavior or detecting broader attack patterns.

### 6.7. Security Considerations

The dynamic update of random values on every interaction is the protocol's strongest defense against replay and cloning. Nevertheless, physical layer attacks such as eavesdropping or relay attacks could still pose a risk unless additional countermeasures like distance bounding are incorporated. Moreover, while the hashing mechanism secures communication, the specific hash function used should be robust against pre-image and collision attacks to prevent brute-force reverse engineering.

### 6.8. Comparison with Existing Solutions

Compared to traditional static-ID tag systems or basic password-based approaches, the proposed protocol offers a significantly higher level of security. Unlike many RFID-based methods that rely on external middleware or encryption chips, our approach provides mutual authentication using only software and minimal hardware extensions, thereby reducing cost while maintaining security.

 doi:10.20944/preprints202601.0012.v1



## 7. Conclusion and Future Work

### *7.1. Conclusion*

The research presented in this paper demonstrates the successful design and deployment of a lightweight yet secure NFC-based authentication framework using embedded hardware. By leveraging a Raspberry Pi in conjunction with a PN532 NFC module, a complete end-to-end authentication system was realized, capable of securely validating NFC tag-client pairs in real time.

The implementation focused on the reproduction of a previously proposed lightweight protocol that combines hash-based message verification with random seed updates to ensure session freshness and protection against replay attacks. Testing across multiple scenarios—including valid authentications, tag tampering, and client-side corruption—confirmed that the system consistently enforced access control and preserved the integrity of communications between devices.

A key takeaway from this work is that secure authentication protocols can be deployed efficiently even on resource-constrained platforms. The use of Python-based libraries for NFC operations and a Node.js server enabled modular development and clear separation between hardware and backend logic. Furthermore, the reliance on dynamically generated secrets and hash transformations mitigated risks posed by unauthorized access, spoofing, or cloning of tags.

Our experiments confirmed the protocol's robustness in various adversarial conditions. Authentication failed when expected in all manipulated scenarios, while all valid client-tag interactions were seamlessly processed. These findings establish the viability of implementing lightweight NFC authentication systems for real-world applications such as access control in buildings, transportation systems, healthcare verification, or identity-driven IoT services.

Moreover, the ability of the system to operate efficiently with minimal overhead makes it an attractive solution for environments where latency, cost, and power efficiency are critical. Compared to traditional RFID systems that either lack dynamic security or require expensive cryptographic chips, the presented solution offers a balanced trade-off between security, flexibility, and scalability.

### *7.2. Future Work*

While the current implementation achieves its intended objectives, several improvements and extensions can be pursued to enhance the system's functionality, security, and scalability. These future directions are outlined below:

- **Enhanced Database Scalability:** The present system utilizes SQLite3 for lightweight local data storage. For deployments at scale—such as smart cities or enterprise access control—migration to distributed or cloud-based databases like PostgreSQL, MongoDB, or Firebase would offer better performance, redundancy, and integration capabilities.
- **Integration of Biometric Authentication:** Security can be significantly bolstered by incorporating biometric data (e.g., fingerprint, iris, or facial recognition) alongside NFC-based credentials. This would enable two-factor or multi-modal authentication, greatly improving identity assurance and reducing reliance on physical tags.
- **Relay Attack Prevention via Distance Bounding:** NFC is inherently vulnerable to relay attacks due to its contactless nature. Integrating distance bounding protocols could help establish whether the communicating devices are within a legitimate proximity range, thereby limiting the possibility of relay or man-in-the-middle attacks.
- **Immutable Logging via Blockchain:** For applications requiring audit trails, compliance, or tamper-proof logging, blockchain technology or distributed ledgers can be employed to store authentication logs in a verifiable and immutable manner, thereby increasing transparency and trust.
- **Fault Tolerance and Tag Recovery Mechanisms:** Currently, the system lacks built-in mechanisms for handling lost or desynchronized NFC tags and clients. Future versions should include a secure reset or recovery protocol, allowing users to re-register without administrator intervention or security compromise.

 doi:10.20944/preprints202601.0012.v1



- **Formal Protocol Verification:** While the current implementation was tested extensively, applying formal verification tools or mathematical models (such as ProVerif, Tamarin, or AVISPA) would enable a rigorous proof of security properties like confidentiality, integrity, authentication, and resistance to known attacks.
- **Performance Benchmarking Under Load:** To better understand system limitations, stress tests under high-frequency tag interactions or concurrent authentication requests should be conducted. This will help identify hardware or software bottlenecks and guide decisions about distributed server deployment.
- **Field Testing in Live Environments:** Pilot deployments in smart environments—such as corporate offices, university campuses, or healthcare clinics—can provide empirical feedback on usability, user satisfaction, latency tolerance, and security perception among non-technical users.
- **Support for Multi-Tag Scenarios:** In complex environments like smart homes or factories, devices may need to interact with multiple tags. Extending the system to support multi-tag management and priority-based access control would open new use cases.

In summary, the proposed NFC authentication system represents a viable solution for contactless, secure authentication in embedded settings. By addressing current limitations and integrating advanced features like biometrics, distance bounding, and scalable cloud backends, the system can evolve into a robust platform for secure identity verification in future digital infrastructures.

## References


1. Baek, J.H.; Youm, H.Y. Secure and lightweight authentication protocol for NFC tag based services. In Proceedings of the 2015 10th Asia Joint Conference on Information Security. IEEE, 2015, pp. 63–68.
2. Madlmayr, G.; Langer, J.; Kantner, C.; Scharinger, J. NFC devices: Security and privacy. In Proceedings of the 2008 Third International Conference on Availability, Reliability and Security. IEEE, 2008, pp. 642–647.
3. Sarma, S.E.; Weis, S.A.; Engels, D.W. RFID systems, security and privacy implications. *MIT Auto-ID Center White Paper* **2002**, *1*, 1–10.
4. Finkenzeller, K. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*; John Wiley & Sons, 2003.
5. Weis, S.A.; Sarma, S.E.; Rivest, R.L.; Engels, D.W. Security and privacy aspects of low-cost radio frequency identification systems. In Proceedings of the International Conference on Security in Pervasive Computing. Springer, 2003, pp. 201–212.
6. Juels, A. RFID privacy: A technical primer for the non-technical reader. *Social Science Research Network* **2005**, *2005*, 1–9.
7. Molnar, D.; Wagner, D. Privacy and security in library RFID: Issues, practices, and architectures. In Proceedings of the Proceedings of the 11th ACM Conference on Computer and Communications Security. ACM, 2004, pp. 210–219.
8. Juels, A.; Weis, S.A. The HB+ protocol: A lightweight authentication protocol secure against some attacks. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security. Springer, 2005, pp. 402–414.
9. Peris-Lopez, P.; Hernandez-Castro, J.C.; Estevez-Tapiador, J.M.; Ribagorda, A. LMAP: A real lightweight mutual authentication protocol for low-cost RFID tags. *Computer Networks* **2011**, *52*, 1408–1419.
10. Lee, Y.S.; et al. Efficient mutual authentication scheme for RFID systems using EMAP protocol. In Proceedings of the 2010 International Conference on Ubiquitous and Future Networks. IEEE, 2010, pp. 320–325.
11. El Madhoun, N.; Guenane, F.; Pujolle, G. A cloud-based secure authentication protocol for contactless-NFC payment. In Proceedings of the 2015 IEEE 4th International Conference on Cloud Networking (CloudNet). IEEE, 2015, pp. 328–330.
12. Lee, J.H.; Kim, Y.T.; Lee, S.M. A secure NFC application protocol for anti-counterfeiting system. In Proceedings of the 2015 International Conference on Information and Communication Technology Convergence (ICTC). IEEE, 2015, pp. 1113–1118.
13. Yu, S.; Xu, N.; Liu, W. A lightweight and provably secure mutual authentication scheme for RFID systems. In Proceedings of the International Conference on Wireless Algorithms, Systems, and Applications. Springer, 2014, pp. 268–278.

 doi:10.20944/preprints202601.0012.v1




14. Liu, X.; Liang, X. Mutual authentication scheme for RFID using lightweight cryptography. *Sensors* **2015**, *15*, 17076–17087.
15. Batina, L.; Biryukov, A.; Preneel, B.; Verbauwhede, I. Lightweight cryptography for low-cost RFID tags: A survey. *Computer Science Review* **2012**, *5*, 79–101.
16. Chien, H.Y.; Chen, C.H. An ultralightweight authentication protocol with backward security for low-cost RFID tags. *Journal of Information Science and Engineering* **2009**, *25*, 55–70.
17. Niu, J.; Wang, Z. A secure RFID authentication protocol based on biometric and hash function. *Journal of Sensors* **2017**, *2017*, 1–8.
18. Liao, Y.P.; Wang, S.S. Efficient RFID authentication scheme based on elliptic curve cryptography. *Computer Standards & Interfaces* **2006**, *29*, 254–259.
19. Martin, H.; Roussillon, Q. Smart ticketing using NFC and cloud technologies. In Proceedings of the 2012 Third International Conference on the Applications of Digital Information and Web Technologies (ICADIWT). IEEE, 2012, pp. 116–121.
20. Ariyapperuma, S.; Mitchell, C.J. Near Field Communication (NFC): The future of e-payments? In Proceedings of the Proceedings of the International Conference on Information and Automation, 2006, pp. 77–82.
21. Albahli, S.; Shamsi, J.; Yahya, A. Efficient authentication system for healthcare using NFC and edge–fog computing. *Journal of Healthcare Engineering* **2021**, *2021*, 1–9.
22. Nguyen, T.; Guarnieri, C. Reverse engineering NFC payment applications. In Proceedings of the Black Hat USA, 2015.
23. Fan, K.; Gong, Z. Lightweight blockchain-based authentication for NFC in vehicular networks. *IEEE Access* **2018**, *6*, 31680–31689.
24. Li, Y.; Chen, X.; Liu, H. An intelligent authentication scheme for secure NFC applications using machine learning. *IEEE Internet of Things Journal* **2020**, *7*, 8514–8525.
25. Xu, L.; Jiang, L.; Shen, J. Secure and efficient authentication for fog-enabled IoT systems. *IEEE Internet of Things Journal* **2019**, *6*, 8463–8470.