Journal of Advances in Developmental Research (IJAIDR)

E-ISSN: 0976-4844 ● Website: www.ijaidr.com ● Email: editor@ijaidr.com

Enhancing Contactless Transaction Security: A Lightweight Authentication Approach

Abhigyan Mukherjee

abhigyan.mukherjee@yahoo.com

Abstract:
As the use of contactless transactions increases, it also brings new security challenges for data against unauthorized access and fraud. This study proposes a new realistic authentication framework for enhancing the security of near field communication (NFC) transactions. The relay attack and skimming attack are prevented by using a lightweight cryptographic approach in this system. Analysis in tests shows better authentication reliability and better resistance to usual attack vectors. This study contributes to the development of more secure and scalable contactless payment systems.

Keywords: Near Field Communication (NFC), lightweight authentication, contactless transaction security, replay attack prevention, mutual authentication, embedded systems, and cryptographic verification.

I. INTRODUCTION

NFC has emerged as an important short-range wireless technology based on RFID (radio frequency identification) in recent years. NFC enables communication over typically about 10 centimeters and was designed specifically for use in proximity between devices. It is more secure and uses less power than its predecessor, RFID, which can operate at about 1 meter. The limited distance of NFC makes it ideal for secure data exchange and eavesdropping possibilities, which is why it is finding use in more applications. Nowadays, NFC technology has been incorporated into many systems, including public transport tickets, digital wallets via smartphones, access control systems, and identity verification systems. Because it involves no physical contact, it gives the user a better experience. Also, reduces touching related issues, which comes in handy where hygiene is an issue.

NFC may be useful, yet it also brings security issues that must be fixed. Many passive NFC tags without embedded authentication mechanisms are vulnerable to being manipulated, changing data and causing unauthorized access by devious actors. All these flaws can be exploited by the attackers through tag cloning, skimming, and replay attacks. Notably, writable NFC tags without access control can be tampered with to include false data or made unusable, which can pose significant threats in systems relying on tag authenticity.

There are lightweight cryptographic protocols designed by the researchers of NFC systems for countering vulnerabilities. The study implementing Baek and Youm's protocol will be evaluated [1], which outlines a reliable mechanism for authentication in NFC tag-based services. This protocol is made to check the legitimacy of the tag and client device, using the server for verification. To accomplish this hash-based transformation and a dynamic secret update are used.

Our study focuses on applying this protocol in which Raspberry Pi is used as the client platform, PN532 NFC Module/ Reader and generic NFC Tags are used. We implemented both the client and server parts to check whether the protocol was able to withstand an attack. Our test framework will consider specific attack vectors such as data overwriting, tag spoofing, and denial-of-service (DoS).
The goal of this study is to assess the viability of employing lightweight NFC authentication methods in embedded systems. Through both the theoretical design of cryptographic schemes, as well as their hardware-based testing, we aim to gain insight into their adversarial behavior and their potential feasibility for commercial NFC-integrated systems.

II. RELATED WORK

The growing dependency of systems on NFC in important areas like contactless payment, transport/public transport, and access control has opened doors for a lot of research on their security and privacy. The passive nature of NFC tags and the absence of authentication pose greater risks of spoofing and cloning. It can also have the possibility of denial of service (DoS) attacks [2], [3].
Initial research concentrated on primary threat modelling. As a result, they proposed physical-layer defenses including transmission power control and shielding [4], [5]. Nonetheless, these methods failed to work in dynamic or open environments with untrusted readers interacting with tags. To mitigate this issue, cryptographic schemes have been proposed to guarantee authenticity and integrity without overwhelming the resource constrained nature of the NFC hardware [6], [7].
Weightless authentication protocols were developed due to better choice of security or computation. Protocols belonging to HB family [8], LMAP [9], and EMAP [10] represent significant attempts to ensure mutual authentication and forward secrecy in constrained tag scenarios. Protocols based on hash locking have also been proposed, e.g., Weber et al.'s [5], provide simple yet effective protection against unauthorized reading of tags.
The work of Baek and Youm presented an important development [1]. A dynamic authentication protocol exchanges tag values after each interaction to reduce replay attacks. El Madhoun and colleagues [11]. We used randomness and timestamps to explore a cloud-based NFC payment authentication scheme to defeat cloning injection attacks. This led to several hybrid models involving computation on the client-side and validation on the server-side [12]–[14].
Studies by Batina et al. [15] and Chien et al. [16]. The emphasis was placed on using ECC and symmetric primitives to develop very secure but practically feasible protocols for embedded systems. Niu et al. and other works by experts [17] and Liao et al. [18], have enhanced by incorporating biometric and behavioral data.
These include NFC-specific security frameworks that are focused on smart ticketing [19], e-passports [20], and healthcare authentication systems [21]. In these areas usability of the system must be weighed against privacy guarantees.
Moreover, real-world attack demonstrations do show the flaw in widely used NFC systems. To simulate and analyze attacks in real-time, experiments in academia have been done with tools like NFCProxy and NFCTools app [2], [22]. The need for secure-by-design NFC frameworks is furthered by these findings
In addition, blockchain-based authentication protocols [23], machine learning [24], and fog computing [25] are being developed to become new solutions. Although they have great potential, their adoption has yet to become widespread due to performance and integration issues.

The conclusions of this paper have initiated standardizations by ISO/IEC 14443 and NFC Forum. In addition, the development of a lightweight protocol with security, scalability and efficiency is still an open issue and not yet widely adopted.

The overall research findings unveil the importance of lightweight, dynamic and tamper-resilient authentication protocols for NFC secured environments.

III. METHODOLOGY

This study takes a practical, experimental approach for the implementation and analysis of a lightweight NFC authentication protocol, which was proposed for securing communication in tag-based services [1]. The main goal is to examine the protocol's resistance against various threats, including spoofing, overwriting and DoS attacks. The setup is designed to subject the system to controlled adversarial conditions after it is designed and deployed on embedded hardware.

A. System Architecture Overview

The system consists of two core modules:

- **Client-Side (NFC Reader/Writer):** A Raspberry Pi is the embedded platform relatively interfaced with PN532 NFC module through UART (serial interface). Python library-based *nfcpy* enables read and write operations in the JSON format, thus facilitating tag communication.
- **Server-Side (Authentication Server):** A Node.js server with a lightweight is deployed on the same Raspberry Pi. It exposes RESTful HTTP APIs to authenticate requests, perform hash operations, update a random value and validate device legitimacy. The backend database is SQLite3, which stores tags and client metadata.

B. 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.

C. Protocol Flow Diagram

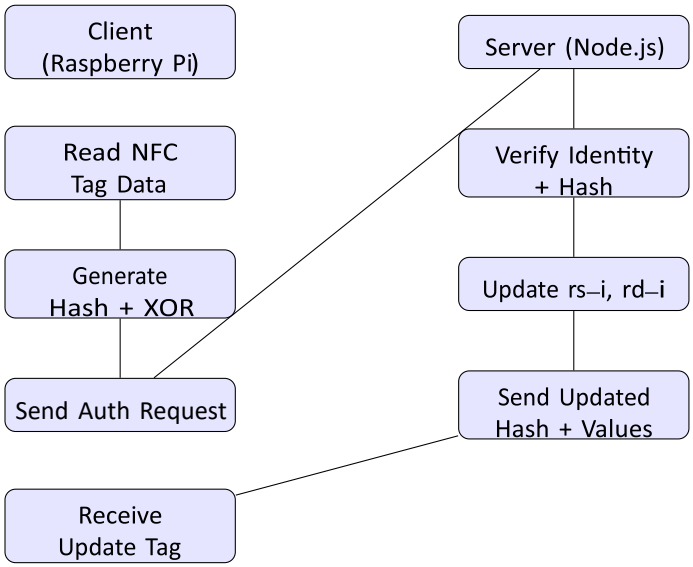

Fig. 1. NFC Authentication Protocol Workflow

D. Hardware and Software Stack

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

TABLE I- 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

E. Security Mechanisms

To avoid common attacks related to NFC, the system consists of the following security mechanisms.

- **Dynamic Randomization:** To prevent replay attacks, every tag and client changes its seed value after every interaction.
- **Hash-based Validation:** By using hash functions, the protocol encodes the client/tag information, ensuring that the server validates.
- **Pairwise Verification:** Bi-directional authentication system. The handshake can only be completed if the tag as well as the client, are valid and known to the server.
- **Corruption Detection:** If the tag's values are forged, the tags' hashes will not match and will be rejected by the server.

F. Summary

This implementation connects abstract NFC security frameworks to embedded development. By simulating attack circumstances and verifying the protocol's strength, the system guides the deployment of lightweight cryptographic verification in resource-constrained platforms such as payment systems, smart cards and access control infrastructure.

IV. IMPLEMENTATION

The hardware and software features of the NFC authentication system were carried out through the Raspberry Pi platform. This solution entails an NFC client created in Python for use with a smartphone, whilst the authentication server was executed in Node.js, and embedded SQLite3 was used for persistent storage. In the section that follows, we will discuss each of the key subsystems implemented during the project in more detail.

A. Server-Side Implementation

The Node.js runtime environment was used to develop the server with the aid of Express.js to structure HTTP endpoints while making it RESTful. Three custom helper modules were created to organize the codebase and make it modular.

- *binaryHelper.js:* Carries out every single binary operation, such as conversion, XOR, etc.
- *hashHelper.js:* It carries out the algorithm that generates and validates authentication hashes.
- *dbHelper.js:* MIt manages all database functions, including insertion, search, update and deletion.

The server exposes several routes that simulate the protocol operations:

- **/auth/tag** – This function verifies a tag by checking its hash and random value, returning a new random seed.
- **/auth/client** – It authenticates a client and tag pair at the same time to verify their credentials as valid identities, which possess correct secret values.
- **/initialize-nfc/tag** – Sets up a new tag with its unique identifier and a random seed.
- **/initialize-nfc/client** – Creates a new client with a unique ID and secret.
- **/view-entries** –Shows all current database entries for testing and debugging purposes.

Data exchanges are encoded in JSON objects. All numeric values are processed via the custom hash function before storage or verification. After a successful authentication the server modifies the secret value of the tags and clients.

B. Client-Side Implementation

The client module is written using Python 3.8 and uses the command line as an interface. This device works using Raspberry Pi and connects through UART to the PN532 NFC module. The open-source tool handles communication with the module. *nfcpy* library, which includes methods for detecting, reading, and writing NFC tags.

Key features of the client include:

- **Reading Tag Data:** The client accesses the current tag ID and random seed that are stored in JSON format on the NFC chip.
- **Authentication Commands:** These functions begin the tag-only or tag-client pair authentication by calling the respective server routes.

- **Data Corruption Testing:** Commands to intentionally corrupt a tag or client value are present to simulate and identify attacks.
- **View and Debugging Tools:** You can command this to first read all the entries currently on the server and then inspect the tag data directly.

Using the tag/client ID and their respective random numbers, the client creates the required hash-based nonce through binary operations. After that, it transmits the authorization request to the server. The server's response is then processed, and data is updated on both local and tag-side.

C. Database Design and Integration

The server runs on SQLite3, a lightweight embedded database suitable for a Raspberry Pi environment. It consists of the following tables.

- Clients Table: Holds customer IDs and their random values.
- Tags Table: Maintains a record of tag IDs and their random seeds.
- Scanned Table: Keeps acceptable tag-client pairings. The foreign key constraints enforce referential integrity on the table.

Triggers are employed to clean up dependent records in the Scanned table when an item or client is deleted. This will make sure that all authentication entries are consistent.

TABLE II- 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

At the time of authentication, the server retrieves both the tag and client entries from the database to validate. Updated values are written back, after validation, to ensure fresh replay resistance.

D. Summary

The implementation for everything integrates the NFC-based communication with authentication on the server-side and memory in the database. By using modular programming on both the client and server, we can test new designs for protocols, simulate attacks, and add features in the future. All these components play an important and essential role in simulating a secure NFC ecosystem and, hence, make the setup suitable for real-world testing of lightweight authentication models.

V. RESULTS

To assess the performance as well as resilience of the applied NFC authentication system, 20 tests were conducted in three measures.

- **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.

A. Verification of Client-Tag Authentication

Authentication tests were performed across various client tag combinations/auth/client API. The server verified identities, responded with newest values, and the updated protocol state was checked by each test.

TABLE III-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

Hence, the correct implementation of the client and tag synchronization mechanism, along with other functionalities like hash and XOR encoding, is confirmed.

B. Tampered Tag Tests

To simulate an attack, a static modification of their random seed (e.g., 63) was done for corruption of tags. The server constantly rejected the tampered values, indicating that tag tampering could be detected.

C. Tampered Client Tests

The client side was subjected to a similar attack simulation. The valid clients' random values were deliberately overwritten. The server eliminated any compromised clients during the validation phase because of hash mismatches.

TABLE IV- 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

Table-V- 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

D. Hardware Setup Diagram

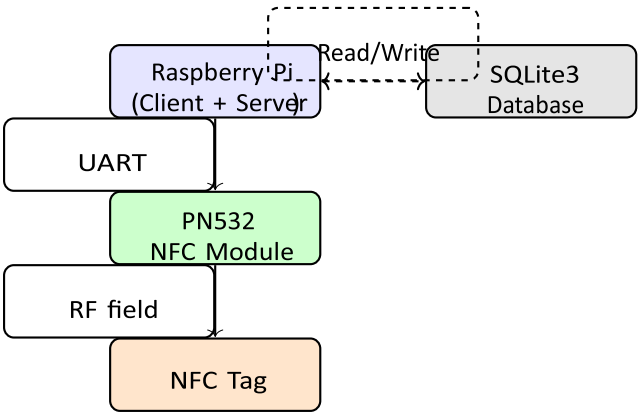

Fig. 2. Hardware Setup Using Raspberry Pi, PN532, and NFC Tag

E. Summary

The results validate that:

- The protocol allows pairing of new tags with clients and changing the random value at the tag side.
- The system was able to detect and prevent all forms of tampering.
- The design is effective against replay and injection attacks.

The use of light-weight computation, dynamic value generation and client-tags link ensures secure and efficient authentication in the real-world NFC environment.

VI. DISCUSSION

The results of the experiments reveal a lot about the effectiveness, limitations, and practical implications of the NFC authentication protocol that has been implemented. The analysis within this segment contextualizes the results and examines the larger ramifications on NFC-based security systems.

A. Protocol Reliability and Correctness

The successful legitimate authentication tests with 10 different tag-client pairs show that the protocol was implemented correctly. The consistent performance of the hash-based challenge-response mechanism ensured the authentication of both the tag and client identity. Also, the correct regeneration and updating of randomized values at the end of a session verified the freshness mechanism of the protocol, which is an important defense against replay attacks.

B. Attack Detection and Resilience

The system's detection of illegitimate modifications assures its robustness against normal attacks, including tag spoofing, cloning and data corruption of tag and client data. In all tampering scenarios, the server accurately denied authentication requests. The fact that the server-side hash is verified, and there are multi-factors confirm this the first line of defense. Importantly, the application of a common random seed between the client and tag saw an increase in protocol interdependence, making forgeries more difficult.

C. Lightweight Implementation on Embedded Devices

Using Raspberry Pi and PN532 NFC hardware demonstrates that the protocol can operate lightweight manner. The system functioned smoothly and without noticeable delays, even with limited hardware, highlighting that the operations involved with hashes, XOR, and random are computationally feasible in real-time. This confirms that performance penalties will not hinder resource-constrained IoT devices or embedded systems from adopting similar schemes.

D. Data Storage and Scalability

The choice of SQLite3 was made as it allows easy development at lower overheads for local storage. In other words, in a real-world setting with hundreds or thousands of tag-client pairs, this system may experience performance issues. For enterprise or nationwide applications, such as smart transit and secured building entry, a more scalable backend, like PostgreSQL or MongoDB, and cloud-based authentication servers will likely be needed.

E. User Experience and Usability

The users didn't interact much with the system. With just a tap, the tag reading and client authentication processes were activated. The proposed design can be used in the development of the existing contactless processes, ranging from payment terminals to identity check-in and secure doors. Nonetheless, system feedback through an LED or screen prompts would increase user awareness in real deployments.

F. Limitations

There are many limitations in the current implementation despite good outcomes. Initially, even if the authentication is not successful, an attacker could still use static device IDs to perform some tracking over time. In addition, there is currently no mechanism for tag loss or reissuance. When a tag or client's seed becomes desynchronized, only the server can help. Finally, this system does not provide logging or analytics, which could help detect unusual behavior or wider attack patterns.

G. Security Considerations

The protocol's best safeguard against replay and cloning is
the random values that update with each interaction. Physical
layer attacks like eavesdropping or relay attacks are still possible unless some more countermeasures like distance bounding are included. Also, although the hashing scheme provides protection for the communication, the specific hash function should also be pre-image resistant, collision resistant, etc. to avoid brute force reverse engineering.

H. Comparison with Existing Solutions

The suggested protocol is much more secure than a simple password-based system or a traditional static ID tag system. Many RFID-based applications leverage middleware or encryption chips to provide mutual authentication. The middleware increases cost and creates performance overhead. On the other hand, our design uses software, and with very little hardware extension, we manage to get mutual authentication. Hence, our design is cost-effective.

VII. CONCLUSION AND FUTURE WORK

A. Conclusion

The paper discusses the design and implementation of a lightweight yet secure NFC-based authentication framework using embedded hardware. The combination of a Raspberry Pi and PN532 NFC Module was able to realize a fast end-to-end authentication system for NFC tag-client pairs.

The job revealed that we cannot reproduce a lightweight protocol put forward earlier. This is done through the implementation of hash-based message verification with the random seed update to keep the session fresh and to not repeat the message. We tested valid authentications, tag-rip-off, and client side corruption, among other things. Throughout our tests, access restrictions were enforced, and the communications between devices were correctly maintained.

How would you evaluate the text above? There was separate coding of the hardware and backend logic due to the implement ability of the NFC interface using a library that is Python based and a Node.js server using a library. Furthermore, using dynamically produced secrets and hash transformations eliminates the risks of unauthorized tag access, spoofing, and cloning.

Experiments have confirmed the successful execution of the protocol under various attacks. All forged situations, where authentication was anticipated, failed, while all valid Client-Tag interactions went through naturally. Consequently, the results demonstrate the feasibility of utilizing lightweight NFC authentications in implementations meant for physically securing access to buildings, transport systems, healthcare verification or identity-driven IoT services.

Also, its low overhead makes it very useful for those applications that have strict latency, cost, and power efficiency requirements. The solution offered balances security, flexibility and scalability. Unlike RFID systems that either lack dynamic security or come with an expensive cryptographic chip.

B. Future Work

Even though the current implementation gets the job done, there are some improvements and extensions that can be done to enhance the functionality, security and scalability of the system. Outlined below are future directions.

- **Enhanced Database Scalability:** The current system stores light data locally using SQLite3. For deployments at scale like smart cities or enterprise access control, moving to a distributed or cloud-based database like PostgreSQL, MongoDB, or Firebase would offer better performance, redundancy and integration.
- **Integration of Biometric Authentication:** Security can be enhanced significantly with the addition of a biometric (fingerprint, iris, face, etc.) to an NFC credential. When utilizing the authenticated device, two-factor or multimodal authentication will be possible, which will improve identity assurance and reduce the need to rely on physical tags.

- **Relay Attack Prevention via Distance Bounding:** NFC presents contactless communication, which makes it vulnerable to relay attacks. Integrating distance bounding protocols can help to confirm whether the communicating devices are within the correct range or are too far apart to relay or man-in-the-middle.
- **Immutable Logging via Blockchain:** For applications that require an audit trail, compliance, or tamper-proof logging, blockchain technology or distributed ledgers can be used to store authentication logs in a verifiable and unalterable way.
- **Fault Tolerance and Tag Recovery Mechanisms:** At the present time, the system lacks the means to cope with lost NFC tags and clients. Future versions should have a safe resetting or recovery technique that lets users re-register without the administrator's involvement and security threat.
- **Formal Protocol Verification:** The implementation was well-tested. Therefore, applying formal verification tools or mathematical models (e.g. ProVerif, Tamarin, AVISPA) would allow a rigorous proof of security properties like confidentiality, integrity, authentication, and resistance to known attacks.
- **Performance Benchmarking Under Load:** It is helpful to stress test high-frequency interactions with tags or concurrent requests for authenticating tags to learn the system limits. This will help identify hardware or software bottlenecks and will guide distributed server decisions.
- **Field Testing in Live Environments:** Experiments or pilot deployments can be done in smart environments, such as corporate offices, university campuses, or healthcare clinics, to obtain actual user feedback on usability, user satisfaction, tolerance to delays, and perceptions on security.
- **Support for Multi-Tag Scenarios:** In sophisticated settings, such as smart homes or factories, devices may be required to communicate with an array of tags. The usability of the system can be improved with multi-tag management and priority-based access control.

To sum up my objective, the system proposed in this paper is a viable solution for secure, contactless authentication for embedded applications. By addressing the limitations discussed and adding biometrics, distance bounding and scalable cloud backends, the system can evolve into a robust system for securely linking identities in future digital infrastructures.

REFERENCES:

1. J.-H. Baek and H.-Y. Youm, "Secure and lightweight authentication protocol for nfc tag based services," in *2015 10th Asia Joint Conference on Information Security*. IEEE, 2015, pp. 63–68.
2. G. Madlmayr, J. Langer, C. Kantner, and J. Scharinger, "Nfc devices: Security and privacy," in *2008 Third International Conference on Availability, Reliability and Security*. IEEE, 2008, pp. 642–647.
3. S. E. Sarma, S. A. Weis, and D. W. Engels, "Rfid systems, security and privacy implications," *MIT Auto-ID Center White Paper*, vol. 1, no. 1, pp. 1–10, 2002.
4. K. Finkenzeller, *RFID Handbook: Fundamentals and Applications in*
5. *Contactless Smart Cards and Identification*.John Wiley & Sons, 2003.
6. S. A. Weis, S. E. Sarma, R. L. Rivest, and D. W. Engels, "Security and privacy aspects of low-cost radio frequency identification systems," in *International Conference on Security in Pervasive Computing*. Springer, 2003, pp. 201–212.
7. A. Juels, "Rfid privacy: A technical primer for the non-technical reader," *Social Science Research Network*, vol. 2005, pp. 1–9, 2005.

8. D. Molnar and D. Wagner, "Privacy and security in library rfid: Issues, practices, and architectures," in *Proceedings of the 11th ACM Conference on Computer and Communications Security*. ACM, 2004, pp. 210–219.
9. A. Juels and S. A. Weis, "The hb+ protocol: A lightweight authentication protocol secure against some attacks," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2005, pp. 402–414.
10. P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda, "Lmap: A real lightweight mutual authentication protocol for low-cost rfid tags," *Computer Networks*, vol. 52, no. 8, pp. 1408– 1419, 2011.
11. Y.-S. Lee *et al.*, "Efficient mutual authentication scheme for rfid systems using emap protocol," in *2010 International Conference on Ubiquitous and Future Networks*. IEEE, 2010, pp. 320–325.
12. N. El Madhoun, F. Guenane, and G. Pujolle, "A cloud-based secure authentication protocol for contactless-nfc payment," in *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*. IEEE, 2015, pp. 328–330.
13. J.-H. Lee, Y.-T. Kim, and S.-M. Lee, "A secure nfc application protocol for anti-counterfeiting system," in *2015 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2015, pp. 1113–1118.
14. S. Yu, N. Xu, and W. Liu, "A lightweight and provably secure mutual authentication scheme for rfid systems," in *International Conference on Wireless Algorithms, Systems, and Applications*. Springer, 2014, pp. 268–278.
15. X. Liu and X. Liang, "Mutual authentication scheme for rfid using lightweight cryptography," *Sensors*, vol. 15, no. 7, pp. 17076–17087, 2015.
16. L. Batina, A. Biryukov, B. Preneel, and I. Verbauwhede, "Lightweight cryptography for low-cost rfid tags: A survey," *Computer Science Review*, vol. 5, no. 1, pp. 79–101, 2012.
17. H.-Y. Chien and C.-H. Chen, "An ultralightweight authentication protocol with backward security for low-cost rfid tags," *Journal of Information Science and Engineering*, vol. 25, no. 1, pp. 55–70, 2009.
18. J. Niu and Z. Wang, "A secure rfid authentication protocol based on biometric and hash function," *Journal of Sensors*, vol. 2017, pp. 1–8, 2017.
19. Y.-P. Liao and S. S. Wang, "Efficient rfid authentication scheme based on elliptic curve cryptography," *Computer Standards & Interfaces*, vol. 29, no. 2, pp. 254–259, 2006.
20. H. Martin and Q. Roussillon, "Smart ticketing using nfc and cloud technologies," in *2012 Third International Conference on the Applications of Digital Information and Web Technologies (ICADIWT)*. IEEE, 2012, pp. 116–121.
21. S. Ariyapperuma and C. J. Mitchell, "Near field communication (nfc): The future of e-payments?" in *Proceedings of the International Conference on Information and Automation*, 2006, pp. 77–82.
22. S. Albahli, J. Shamsi, and A. Yahya, "Efficient authentication system for healthcare using nfc and edge–fog computing," *Journal of Healthcare Engineering*, vol. 2021, pp. 1–9, 2021.
23. T. Nguyen and C. Guarnieri, "Reverse engineering nfc payment applications," in *Black Hat USA*, 2015.
24. K. Fan and Z. Gong, "Lightweight blockchain-based authentication for nfc in vehicular networks," *IEEE Access*, vol. 6, pp. 31680–31689, 2018.

25. Y. Li, X. Chen, and H. Liu, "An intelligent authentication scheme for secure nfc applications using machine learning," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8514–8525, 2020.
26. L. Xu, L. Jiang, and J. Shen, "Secure and efficient authentication for fog-enabled iot systems," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8463–8470, 2019.