QSNFC: Quick and Secured Near Field Communication for the Internet of Things

Thomas Ulz, Thomas Pieber, Christian Steger
Institute for Technical Informatics
Graz University of Technology
Graz, Austria
{thomas.ulz, thomas.pieber, steger}@tugraz.at

Sarah Haas, Rainer Matischek
Design Center Graz
Infineon Technologies Austria AG
Graz, Austria
{sarah.haas, rainer.matischek}@infineon.com

***Abstract*—Near Field Communication (NFC) is used for a wide range of security-critical applications such as payment or access control. Although such applications require secured data transfer, the NFC protocol does not include transport layer security. Other protocols that are built on top of NFC, such as the NFC data exchange format (NDEF), only provide insufficient security measures. Therefore, implemented security solutions are often application specific and do not follow well-established standards. To facilitate NFC usage in the Internet of Things (IoT) where millions of devices need to be secured, an efficient and sufficiently secured NFC-based protocol needs to be developed. In this paper, we present the Quick and Secured NFC (QSNFC) protocol. Our protocol is capable of performing more efficient key agreements for recurring connections, and thus, can be used as an efficient alternative to the Transport Layer Security (TLS) protocol.**

***Index Terms*—Near Field Communication; Internet of Things; Secure Communication; Transport Layer Security.**

I. Introduction

The Internet of Things (IoT) is rapidly growing due to devices being used in a wide range of domains such as smart homes, transportation, healthcare, or industrial scenarios. To assist the rapid growth in the number of application domains as well as in the number of IoT devices, several enabling technologies are required. Al-Fuqaha et al. [1] identify the latest developments in Radio Frequency Identification (RFID), smart sensor technology, and communication technologies and protocols as such enabling technologies. Together with RFID, the authors also mention Near Field Communication (NFC) as a very promising technology for the IoT since many smartphones nowadays are equipped with NFC-enabling technology. In the context of IoT related communication protocols, security is an often neglected aspect as highlighted by the increase in IoT related security breaches [2]. Such security breaches can be fatal if IoT devices are used in domains where malicious functionality could harm human lives such as industrial settings [3], or in healthcare applications [4]. To provide secured communication for IoT devices, protocols that are well known from the traditional Internet cannot be used due to their performance requirements. Especially, if considering NFC, protocols based on the Transmission Control Protocol (TCP) entail a large communication overhead and thus, are infeasible for most devices and scenarios.

Although it is often believed that the limited communication range of NFC obviates the need for dedicated security measures [5], Haselsteiner and Breitfuß demonstrate that eavesdropping data is possible up to $10\,\mathrm{m}$ [6]. If transferred data is protected by weak security measures or even transferred unprotected, attacks are threatening the confidentiality of critical information [7], [8]. Plósz et al. [9] compare the provided security of various wireless communication technologies with NFC. The authors state that although there are several security related mechanisms defined in the NFC standard, many attacks are possible despite these mechanisms. Chen et al. [10] and Chatta et al. [11] list a large number of attacks that are feasible for attackers to perform due to weak or insufficient security in the NFC protocol. To mitigate such problems, many approaches for secured NFC communication have been proposed (see Section III Background and Related Work). However, the drawback with these approaches is that no standardized protocols are used. On the one hand, this fact complicates the use of NFC in IoT applications since the security of each application specific protocol needs to be proven separately. On the other hand, applying protocols with proven security features that where designed for the Internet to NFC communication entails a large overhead.

Contributions. We make the following contributions in this paper. We demonstrate Quick and Secured NFC (QSNFC), a protocol that is suited for NFC-equipped IoT devices. The provided security features are similar to TLS, while the protocol will require fewer messages to be exchanged during key agreement. Thus, the presented protocol will be suitable for any NFC-based IoT scenario, while allowing easy security proofs. To the best knowledge of the authors, no comparable protocol for NFC has been presented yet.

Outline. The remainder of this paper is structured as follows. We define our system model and list corresponding assumptions in Section II. In Section III, background information on involved technologies as well as related work for secured communication are given. Our QSNFC approach is presented in Section IV and evaluated regarding its performance and security in Section V. Section VI discusses example use-cases. This paper is then concluded with Section VII where also future work will be discussed.

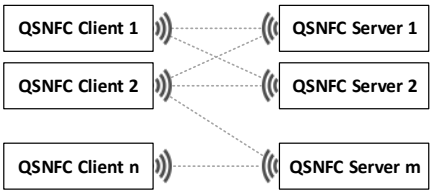


Fig. 1: System model in our proposed QSNFC approach: Over time, clients are capable to connect to an arbitrary number of servers. Servers can manage connections from an arbitrary number of clients (not simultaneous).

II. System Model and Assumptions

The system model we are considering when designing the QSNFC protocol is shown in Fig. 1. As shown in that model, the protocol is supposed to support an arbitrary number of QSNFC *clients* as well as an arbitrary number of QSNFC *servers*. Although the notion of server-client is not common in NFC solutions, we use these terms here to be compatible to other network related protocols such as TLS. Thus, we identify the following four entities in our system model:

QSNFC client: The QSNFC client is the entity that tries to establish a secured communication channel with the QSNFC server. Since this entity is initiating the NFC communication, it can be seen as the *active component* in NFC terms.

QSNFC Server: The QSNFC server is contacted by the QSNFC client in order to establish a secured communication channel. In NFC terms, the QSNFC server would be denoted as the *passive component*.

Communication channel: The communication channel that we assume in our system model is an NFC channel with the potential presence of an undetected adversary.

Adversary: The adversary present in our system model is assumed to be capable of eavesdropping and modifying ongoing NFC data. There is no assumption regarding the range in which these malicious activities are feasible.

III. Background and Related Work

A. Key Agreement and Transport Layer Security (TLS)

If two or more parties need to agree on a shared secret in the potential presence of an adversary, key agreement protocols are used. Usually, key agreement is performed between two parties over an unsecured channel such as the Internet. The final shared secret that is used as a key is composed of influences from all involved parties without revealing the key to any potential adversary that is capable of eavesdropping the key agreement process. One of the most widespread key agreement protocols is the Diffie-Hellman (DH) protocol [12] that is used for key agreement in the TLS protocol [13]. TLS uses TCP as transport protocol and is used to secure connection oriented applications such as web browsing, emails, or instant messaging. Due to its connection oriented nature, it introduces a lot of overhead which might not be suitable for resource constraint devices. Therefore, the Datagram Transport Layer Security (DTLS) protocol [14] was introduced, which uses the User Datagram Protocol (UDP) as its transport protocol. Both protocols have in common that at least one communication round trip time (RTT) is required for the key agreement process when establishing a secured channel.

B. Authenticated Encryption (AE)

AE is capable of providing data confidentiality, integrity, and authenticity by combining symmetric cryptography with Message Authentication Codes (MAC) in a secured way [15]. The widespread Advanced Encryption Standard (AES) provides modes of operation (e.g. AES-CCM, which is used in TLS) that are capable of providing AE [16].

C. Zero Round Trip Time (0-RTT)

The Internet Engineering Task Force (IETF) is currently working on the new TLS 1.3 standard that also includes a 0-RTT requirement [17]. The requirement specifies that the key agreement for recurring connections should not require a traditional handshake and thus, no round trip communication. Recurring connections are specified by the IETF as connections where the two communication partners previously already have established a secured channel, including the 1-RTT handshake required by the key agreement. Protocols that meet the 0-RTT requirement are, for example, OPTLS [18] and Google's Quick UDP Internet Connections (QUIC) protocol [19] that is designed for UDP connections.

D. Quick UDP Internet Connections (QUIC)

QUIC is a protocol presented and developed by Google to enable the transfer of websites via Hypertext Transfer Protocol (HTTP) over UDP instead of TCP [20]. The main goal of QUIC is to improve the perceived performance of web applications, compared to HTTP over TCP. This improvements are achieved by relying on multiplexed UDP connections and by using 0-RTT secured connections that are capable of providing the same level of security as TLS [21]. To make the protocol robust while using unreliable UDP packet transfer, QUIC also includes mechanisms to deal with packet loss, congestions, and error corrections. QUIC is integrated in current versions of Chrome and Chromium and according to Google deployed on thousands of their servers [22]. Also, an IETF working group for QUIC was founded in 2016.

E. Secured Near Field Communication (NFC)

NFC is a contactless communication technology that is based on several Radio-Frequency Identification (RFID) standards. Similar to High Frequency (HF) RFID, NFC operates at a frequency of $13.56\,\mathrm{MHz}$ and has a relatively short communication range of typically up to $10\,\mathrm{cm}$ with a bit rates of up to $848\,\mathrm{kbps}$. NFC is used in a very diverse range of fields, the most well known and widespread of them being payment applications. The IoT is believed to be a new major field for NFC applications [23] that will post new challenges for NFC technology, such as standardized secured protocols.

Although security is not a major topic in RFID related research, some promising approaches have been presented.

TABLE I: Comparison with related work. We compare the security attributes confidentiality, integrity, and authenticity. In addition, we state if a standardized protocol is used and if the used protocol is efficient in terms of communication overhead.

	Confidentiality	Integrity	Authenticity	Standardized	Efficient
Secret sharing [24]	✗/✓	✗	✗/✓	✗	✗
Secure UHF-RFID Tag [25]	✓	✗	✓	✗	✗
Mobile payment [26]–[28]	✓	✓	✓	✗	✗
Healthcare [29], [30]	✓	✓	✓	✗	✗
Car immobilizer [31]	✓	✓	✓	✗	✗
TLS over NFC [32]	✓	✓	✓	✓	✗
QSNFC [this work]	✓	✓	✓	✓	✓

For instance, Toyoda and Sasase [24] present a secret sharing mechanism that aims at confidentialiy distributing keys. Li et al. [25] present authentication and authorization mechanisms between tag and reader such that a trust relationship between these two devices can be established. However, due to the more powerful communication capabilities of NFC, more complex algorithms can be realized compared to RFID. Especially in the payment sector where security is of utmost importance, many application specific security solutions are presented [26]–[28]. Also in healthcare, where security weaknesses could directly impact the health of users or even threaten their lives, concepts for secured NFC communication are presented [29], [30]. Of course, there are also many other useful application scenarios for secured NFC communication, such as, for example, an NFC-based car immobilizer [31]. All of these approaches have in common that they implement application specific security mechanisms which is critical regarding the use of NFC in IoT devices since no general security assessment for the technology can be made. Urien and Piramuthu [32] try to mitigate this problem by proposing to use TLS over NFC. However, the TLS protocol was designed for TCP-based connections, and thus, entails a large protocol overhead. A comparison is given in Table I.

IV. Quick and Secured NFC

A. Classification in Layer Model

Before specifying our proposed QSNFC protocol in detail, we are going to classify it according to the TCP/IP protocol architecture layer as shown in Fig. 2. In that figure, the similarities to TLS and DTLS are highlighted. Similar to TLS and DTLS our QSNFC protocol resides directly underneath the actual application and provides capabilities for secured data transfer to the upper layer. As a transport protocol, the NFC Data Exchange Format (NDEF) that is based on application protocol data unit (APDU) packets is used. NDEF itself provides limited security measures, such as signature records. However, these security measures are shown to be vulnerable to certain attacks [33]. Therefore, we use NDEF as a transport protocol for QSNFC only, without relying on any of the available security features of NDEF.

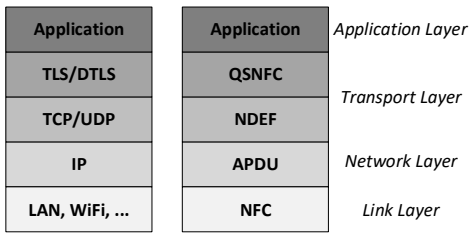


Fig. 2: Protocol stacks for TLS/DTLS and QSNFC respectively, layered according to the TCP/IP model. Both TLS/DTLS and QSNFC reside underneath the application layer and provide their functionality to higher layers, while relying on lower-layer protocols to perform data transfer.

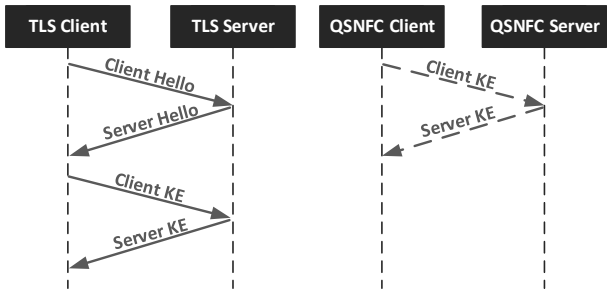


Fig. 3: Round trips required for TLS and QNFC. The TLS connection setup requires two round trips, while the QSNFC connection setup requires one round trip the first time two devices establish a connection and zero rount trips after that.

B. Connection Establishment

Since data transfers using QSNFC rely on secured data channels, key agreement needs to be performed. To meet the 0-RTT requirement for recurring connections, the client needs to cache information about the server if a successful *initial handshake* is performed. The performance of *subsequent handshakes* can then be improved by using this cached information. Fig. 3 demonstrate the handshake process in comparison to TLS. To identify cached information, QUIC uses a set of *URI*, *hostname*, and *port number*. Since this information is not available in NFC, unique identifiers will be used to identify entities. However, the handshake process itself that comprises of initial and subsequent handshake is *not* modified.

Initial handshake. Since on the first connection attempt the client has no cached information about the respective server, an initial handshake needs to be performed. To initiate this handshake, the client sends a so-called *inchoate client hello (CH)* message to the server, which recognizes the inchoate information and replies with a *reject (RJ)* message. This RJ message contains the following information:

(i) The server's long-term DH public value. This public key is used for the generation of subsequent keys and thus, needs to be cached by the client.

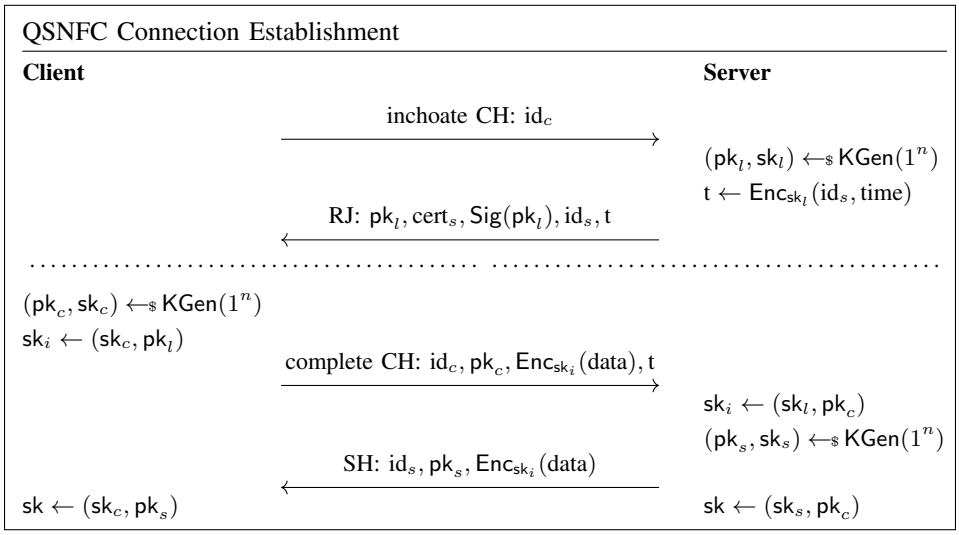


Fig. 4: Connection establishment in QSNFC. All message types are inherited from QUIC [22]; only the content of these messages is adapted to be better suited for NFC. The messages above the dotted line represent the *initial handshake*. The messages underneath the dotted line represent the *subsequent handshake* only. The parameters are: the client's and server's id ($\mathrm{id}_{c|s}$), the long-term DH public and secret value (pk_l, sk_l), the server's and client's ephemeral DH public and secret value ($\mathrm{pk}_{s|c}$, $\mathrm{sk}_{s|c}$), the server's certificate (cert_s), the initial key (sk_i) and the final shared key (sk).

(ii) A certificate chain that authenticates the server and that needs to be verified during the initial handshake.
(iii) A signature of the long-term DH public value that is signed using the private key from the provided certificate chain's leaf certificate.
(iv) A source address token that contains the server's unique ID and a nonce from the server. This information is protected using AE. The client needs to send this token back to the server in subsequent handshakes to demonstrate ownership of the server's identity.

After the client has received this information, it can authenticate the server's long-term DH public value using the provided certificate chain and signature. In addition, the certificate chain is validated using a higher-ranking certificate. After that, the client sends a *complete CH* that contains the client's ephemeral DH public value as well as an optional payload that can already be encrypted using a key generated from the server's long-term DH public value and the client's ephemeral DH public value.

Subsequent handshake. Since the client already is in possession of the server's long-term DH public value, it can calculate a shared key using its own ephemeral DH public value. The client can then send a *complete CH*, without first sending a *inchoate CH* message as is done in the initial handshake. Thus, the first RTT from the initial handshake is not required and encrypted data can be sent to a known server instantly.

If the handshake is successful, a *server hello (SH)* message is sent by the server as response to the complete CH. The SH is encrypted using a key generated from the server's long-term public DH public value and the client's ephemeral DH public value. The SH message also contains the server's ephemeral DH public value. The complete connection establishment is shown in Fig. 4. After both involved entities are in possession of each others' ephemeral DH public value, a *forward-secure key* can be calculated for the connection. Thus, after the SH message is sent and received, both communicating entities switch to encrypting packets with the forward-secure keys.

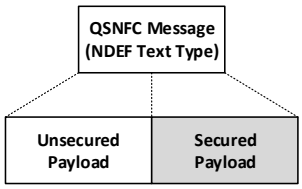


Fig. 5: Basic structure of QSNFC messages. A text record type NDEF message comprises unsecured and secured payload.

C. Connection Tear Down

Although there is no *connection* concept in NFC, we previously discussed connection establishment. During this connection establishment, keys between server and client are exchanged and stored at the communication partners. In addition, also information regarding the other communication partner, such as a source address token need to be cached. However, since there are no concepts such as out-of-order packet reception or multiple streams that are known from TCP-based connection, also no connection tear down is needed in our QSNFC protocol. Methods regarding the replacement of cached keys and information will be discussed in Section IV-E.

D. Packet Structure

The basic packet structure of each QSNFC message is shown in Fig. 5. As can be seen there, any QSNFC message comprises unsecured as well as secured payload inside a *text record type* NDEF message. Similar to TLS and DTLS that use TCP and UDP as their transport protocols, we build QSNFC on top of NDEF messages for the following two reasons: (i) NDEF is a standardized data exchange format for NFC. Similar to TLS over TCP and DTLS over UDP we can utilize it as transport protocol without any modification to the underlying protocol. (ii) The security aspect of data transfer is cleanly separated from the data transfer aspect. That is, limitations such as maximum APDU size do not need to be considered in our proposed QSNFC protocol.

The handshake protocol shown in Fig. 4, comprises four different message types: CH messages, RJ messages, SH messages, and standard data messages (SD) that are shown in Fig. 6. The specified field lengths are calculated for $128\,\mathrm{bit}$ keys. All of these four message types contain three unsecured message attributes that are common to all of them.

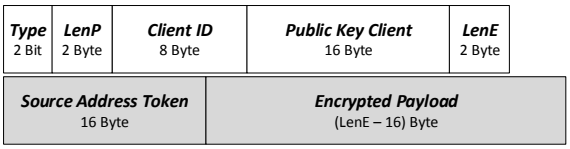
- *Type:* The message's type with allowed options `0b00` CH, `0b01` RJ, `0b10` SH, and `0b11` SD.
- *LenP:* Specifies the length of unsecured payload contained in this message. The length of this field is $2\,\mathrm{Byte}$.
- *LenE:* Specifies the length of secured payload contained in this message. The length of this field is $2\,\mathrm{Byte}$.

CH messages can be either an inchoate CH message or a complete CH message. The packet structure for both of these two types is shown in Fig. 6a. If a client initiates the initial handshake by sending an inchoate CH message, the *Client ID* is set accordingly with each other attribute being empty.

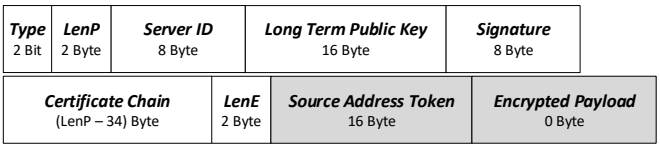
- *Client ID:* A unique ID identifying the client. The Client ID is contained in inchoate CH as well as in complete CH messages. The size of this attribute is $8\,\mathrm{Byte}$.
- *Public Key Client:* The client's ephemeral DH public value according to the handshake protocol shown in Fig. 4. This attribute is only set for complete CH messages and is $16\,\mathrm{Byte}$ in size.
- *Source Address Token:* Only sent by the client for complete CH messages. The value is obtained in the server's RJ message during initial handshake and stored at the client, for instance, in an SE. This attribute's size $16\,\mathrm{Byte}$.
- *Encrypted Payload:* In case of complete CH messages, also an encrypted payload is contained (see Fig. 4). The size of this field is determined by the *LenE* attribute.

RJ messages are sent as a response to inchoate CH messages. The message contains information from the server that is required to perform connection establishment and key agreement. In contrast to other message types, no additional arbitrary payload can be included in RJ messages. The structure of RJ messages is shown in Fig. 6b.

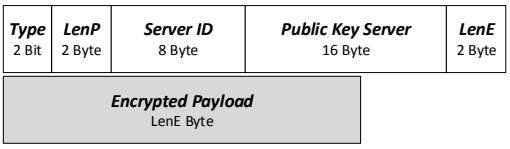
- *Server ID:* A unique ID identifying the server. The Server ID is used by the client to match cached information to the correct server. This attribute has a size of $8\,\mathrm{Byte}$.



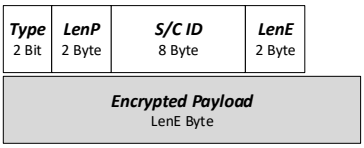
(a) CH message structure.



(b) RJ message structure.



(c) SH message structure.



(d) SD message structure.

Fig. 6: Message structures for CH, RJ, SH, and SD messages. The specified field lengths are are calculated for $128\,\mathrm{bit}$ keys.

- *Long Term Public Key:* The long-term DH public value that is used for key agreement and to calculate initial keys. Using this key, the client is capable of sending encrypted payload inside complete CH messages. The Long Term Public Key has a length of $16\,\mathrm{Byte}$.
- *Signature:* A signature of the Long Term Public Key that is used by the client to validate the integrity of that key. The signature has a size of $8\,\mathrm{Byte}$.
- *Certificate Chain:* The certificate chain that is used by the client to authenticate the server. Also, the private key of the certificate chain's leaf certificate is used to create the previously mentioned Signature. The Certificate Chain has a variable length, depending on the issued certificates.
- *Source Address Token:* This token is used by the client in subsequent handshakes. It includes the server's identifier and a nonce, and is protected by AE using the server's private key. The attribute has a size of $16\,\mathrm{Byte}$.
- *Encrypted Payload:* In RJ messages, only the Source Address Token is contained in the encrypted payload.

SH messages are sent by the server in response to a successful connection establishment. The structure of this message type is shown in Fig. 6c.

- *Server ID:* A unique ID identifying the server. The Server ID is used by the client to match cached information to the correct server. This attribute has a size of $8\,\mathrm{Byte}$.
- *Public Key Server:* The servers's ephemeral DH pub-

lic value according to the handshake protocol shown in Fig. 4. After reception of this attribute, server and client can calculate a forward-secure key to protect data transferred with subsequent SD messages. The size of this attribute is $16\,\mathrm{Byte}$.
- *Encrypted Payload:* SH messages can contain arbitrary payload that needs to be transferred from the server to the client. This information is protected by AE using the previously established initial keys.

SD messages are exchanged between server and client after successful handshakes. Thus, after a forward-secure key was established between these two entities, SD messages with minimal protocol overhead can be used to transfer arbitrary payload in an efficient but secured way.

- *S/C ID:* The respective ID of either server or client is included to identify the sender of an SD message. The length of this attribute is $8\,\mathrm{Byte}$.
- *Encrypted Payload:* Payload of arbitrary length that is protected by AE using the previously established ephemeral forward-secure keys.

E. Cached Data Replacement

The 0-RTT capability of QSNFC entails that information such as public keys and source address tokens need to be cached at server and client side. Since many NFC-enabled IoT devices are resource constraint in terms of memory capacity, a mechanism for cached data replacement needs to be included in our proposed QSNFC protocol. Depending on the use-case scenario in which the protocol is used, different replacement strategies might be better suited than others. Therefore, we briefly discuss three cached data replacement methods that are suitable for IoT devices due to their minimal overhead in terms of complexity and resource requirements [34].

Least Frequently Used (LFU): In the LFU algorithm, an access counter for each cached dataset is kept that counts the number of usages of that respective cached dataset. After a fixed number of connection establishments, all counters are reset. If a dataset needs to be evicted from memory, the dataset with the smallest access count is selected for replacement.

+ Only counters are needed which is a minimal overhead.
– The required regular reset of counters might lead to the eviction of often used datasets.

Least Recently Used (LRU): The LRU algorithm is a special variante of the LFU algorithm. Instead of counters, timestamps are kept for each cached dataset. Whenever a dataset is accessed, the timestamp is updated. If a datasets needs to be evicted from memory, the dataset that was accessed the farthest back in history is selected for replacement.

+ Frequently accessed datasets not falsely evicted.
– Resource constraint devices such as smart cards do not provide the required timestamps.

First In First Out (FIFO): Cached datasets are kept in a queue. Each new dataset is added at the front of the queue. If a dataset needs to be evicted from memory, the last element in the queue is selected for replacement.

+ Smallest overhead of all three methods.
– Dataset that gets cached first gets evicted first although that element might be the most used one.

V. Evaluation

A. Security Analysis

As shown in the packet structures (see Fig. 5), each packet that is transmitted using our proposed QSNFC protocol contains a section dedicated to secured payload. To protect the confidentiality, integrity, and authenticity of this secured payload, AE with either initial keys or ephemeral forward-secure keys is used. Depending on which type of key is used, two levels of secrecy can be provided: (i) Initial data that is protected using initial keys is protected at a level similar to TLS session resumption with session tickets. (ii) If the forward-secure keys are used, even greater secrecy can be provided since these keys are ephemeral. However, depending on the application and use-case, probably only one message round trip is needed. In this case, the initial keys only will be used to protect the data, without ever using the forward-secure keys. Any information that is transmitted unsecured in QSNFC (e.g. server and client identifiers, or public keysis) is considered non-critical. That means, an adversary would gain no advantage by learning this information. To highlight the provided security, we analyse the countermeasures provided by QSNFC for each of the threats to NFC that were identified by Haselsteiner and Breitfuß [6].

Eavesdropping: Since confidential information is transmitted protected by AE at any step of QSNFC (during the handshake and SD messages), an eavesdropper would only be able to learn information that is considered public, such as server and client identifiers, or public keys.

Data Corruption, Data Modification, Data Insertion: An adversary would not be able to corrupt, modify, or insert data in the secured payload section of QSNFC without such failures being detected by the protocol since the secured payload is protected by AE. However, *denial-of-service (DoS)* attacks cannot be mitigated by QSNFC since an adversary can corrupt transferred information at any time, and thus, cause data to be invalid. Also, if an adversary is able to modify information such as server or client identifiers, successful DoS is possible.

Denial-of-Service (DoS) attacks: DoS attacks cannot be mitigated by QSNFC (and any other wireless or contactless communication protocol since data corruption can only be detected but not prevented.

Man-in-the-Middle (MITM) attacks: By relying on certificates for authentication, and on a DH based key agreement, MITM attacks mitigated by QSNFC.

Physical attacks: Cryptographic operations that are required for our proposed key agreement process can either be performed in software, or in a dedicated hardware secure element, such as SIM cards or security controllers. To provide a higher level of security, tamper-resistant security controllers need to be used, such that potential adversaries are not able to extract confidential information using physical attacks [35].

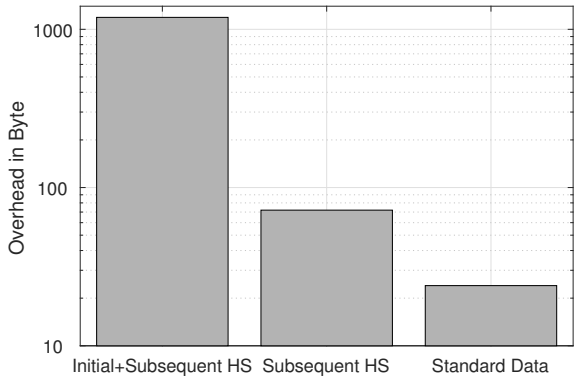


Fig. 7: Comparing three scenarios: initial and subsequent handshake (HS), subsequent HS, and standard data (SD) messages.

B. Protocol Overhead

To analyze the protocol overhead entailed by additional security mechanisms, we analyze three different communication scenarios: (i) Server and client have never completed a handshake, so an initial handshake followed by a subsequent handshake is required. (ii) Server and client only need to perform the subsequent handshake. (iii) Standard data transfer using SD messages that can be exchanged after a completed handshake. An overview of the resulting overhead for all three scenarios is shown in Fig. 7. The resulting overheads that can be seen there are $1182\,\mathrm{Byte}$ for initial+subsequent handshake, $72\,\mathrm{Byte}$ for subsequent handshake only, and $24\,\mathrm{Byte}$ for an exchange of two SD messages. The largest part of the overhead required by the initial handshake results from the included certificate chain. For this evaluation, we used a self-signed certificate that was generated with the following two commands: `openssl ecparam -name secp521r1 -genkey -out key.pem` and `openssl req -new -x509 -key key.pem -out cert.pem -days 365`. The generated certificate already has a size of roughly $1000\,\mathrm{Byte}$. Using a chain of certificates would result in an even larger overhead. By not requiring initial handshakes for recurring connections, roughly 90 percent of overhead can be avoided, compared to traditional protocols such as TLS that involve key agreements. Thus, it can easily be seen why introducing a secured protocol for NFC that is capable of fulfilling the 0-RTT requirement is crucial.

C. Necessary Trade-off

Since most IoT devices are resource constraint, the adoption of a secured communication protocol also entails drawbacks for these devices in terms of complexity, energy consumption, cost, and memory requirements. However, since we see a secured communication channel as a given requirement, we only compare our presented approach to other protocols that involve key agreement, specifically TLS. While QSNFC significantly reduces the protocol overhead for recurring connections, additional, non-volatile memory is required to store cached information. While the additional memory required by QSNFC might be unproblematic for most IoT devices, very constraint devices might require additional memory to be added. This additional requirement will likely cause a slight increase in complexity, energy consumption, and subsequently device costs compared to using TLS. This means, a trade-off between communication efficiency and device complexity, energy consumption and costs needs to be considered.

VI. Example Use-Cases

Depending on the use-case scenario, the roles of server and client might be assigned differently, since a client must be able to validate the certificate chain provided by the server (see Section IV-B). For validation, the client either needs to be in possession of a higher-ranking certificate or have an active Internet connection. Therefore, we list three use-cases that we see as the most common scenarios for our QSNFC protocol and briefly discuss the role assignment.

Card and Reader: The reader in this scenario acts as active NFC device and provides the required energy to power the smartcard through its NFC field. Therefore, in this scenario the reader should be assigned the client role and initiate the connection establishment. Also, a reader will have the capability to validate the server's (smartcard) certificate chain over the Internet in most cases.

Smartphone and IoT Device: In this scenario, the smartphone should initiate the QSNFC handshake and thus, act as a client. Since all modern smartphones are equipped with ample storage and Internet connections, the required validation of the server's certificate chain is also feasible in such a setting.

Machine-to-Machine (M2M): In M2M communication settings such as Robot-to-Machine, the assignment of client and server role cannot be determined in general. The roles should be assigned accordingly, such that the validation of the server's certificate chain is feasible for the client.

VII. Conclusion and Future Work

To foster the use of NFC-technology in IoT devices and use-cases, a standardized and secured, yet efficient protocol is required. Currently, either application specific security solutions, or protocols that entail too much overhead such as TLS are used to secure NFC-based data transfers. The protocol presented in this publication, QSNFC, is designed with both standardized security mechanisms and efficiency in mind. The protocol fulfils the 0-RTT requirement to increase the performance of recurring connections between devices. Data confidentiality, integrity, and authenticity for transferred data is provided by relying on AE, while the imposed protocol overhead is kept at a minimum. As a trade-off compared to traditional protocols that involve key agreement such as TLS, our proposed algorithm requires more local memory to store cached information. As future work, we plan to investigate protocol improvements that further reduce the protocols overhead. Thus, making secured NFC data transfer even more efficient and suitable for IoT devices.

ACKNOWLEDGMENT

We thank our shepherd, Shashi Ramamurthy, and all the anonymous reviewers for their constructive comments. This work has been performed within the IoSense (http://iosense.eu) project. This project has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 692480. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and Germany, Netherlands, Spain, Austria, Belgium, Slovakia. IoSense is funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program "ICT of the Future" between May 2016 and April 2019. More information: https://iktderzukunft.at/en/.

REFERENCES

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[2] M. U. Farooq, M. Waseem, A. Khairi, and S. Mazhar, "A Critical Analysis on the Security Concerns of Internet of Things (IoT)," *International Journal of Computer Applications*, vol. 111, no. 7, 2015.

[3] A.-R. Sadeghi, C. Wachsmann, and M. Waidner, "Security and Privacy Challenges in Industrial Internet of Things," in *Design Automation Conference (DAC), 52nd ACM/EDAC/IEEE*. IEEE, 2015, pp. 1–6.

[4] L. M. R. Tarouco, L. M. Bertholdo, L. Z. Granville, L. M. R. Arbiza, F. Carbone, M. Marotta, and J. J. C. de Santanna, "Internet of Things in Healthcare: Interoperatibility and Security Issues," in *Communications (ICC), IEEE International Conference on*. IEEE, 2012, pp. 6121–6125.

[5] L. Finžgar and M. Trebar, "Use of NFC and QR code Identification in an Electronic Ticket System for Public Transport," in *Software, Telecommunications and Computer Networks (SoftCOM), 19th International Conference on*. IEEE, 2011, pp. 1–6.

[6] E. Haselsteiner and K. Breitfuß, "Security in Near Field Communication (NFC)," in *Workshop on RFID security*, 2006, pp. 12–14.

[7] M. Emms and A. van Moorsel, "Practical Attack on Contactless Payment Cards," in *HCI2011 Workshop-Heath, Wealth and Identity Theft*, 2011.

[8] P. Fraga-Lamas and T. M. Fernández-Caramés, "Reverse Engineering the Communications Protocol of an RFID Public Transportation Card," in *RFID (RFID), 2017 IEEE International Conference on*. IEEE, 2017, pp. 9–11.

[9] S. Plósz, A. Farshad, M. Tauber, C. Lesjak, T. Ruprechter, and N. Pereira, "Security Vulnerabilities and Risks in Industrial Usage of Wireless Communication," in *Emerging Technology and Factory Automation (ETFA), 2014 IEEE*. IEEE, 2014, pp. 1–8.

[10] C. H. Chen, I. C. Lin, and C. C. Yang, "NFC Attacks Analysis and Survey," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2014 Eighth International Conference on*. IEEE, 2014, pp. 458–462.

[11] N. A. Chattha, "NFC - Vulnerabilities and Defense," in *Information Assurance and Cyber Security (CIACS), 2014 Conference on*. IEEE, 2014, pp. 35–38.

[12] W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[13] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," Internet Requests for Comments, RFC 5246, August 2008.

[14] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security Version 1.2," Internet Requests for Comments, RFC 6347, January 2012, accessed on 2017-11-20. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6347.txt

[15] M. Bellare and C. Namprempre, "Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2000, pp. 531–545.

[16] D. McGrew and D. Bailey, "AES-CCM Cipher Suites for Transport Layer Security (TLS)," Internet Requests for Comments, RFC 6655, July 2012.

[17] B. Dowling, M. Fischlin, F. Günther, and D. Stebila, "A Cryptographic Analysis of the TLS 1.3 Handshake Protocol Candidates," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 1197–1210.

[18] H. Krawczyk and H. Wee, "The OPTLS Protocol and TLS 1.3," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 2016, pp. 81–96.

[19] M. Fischlin and F. Günther, "Multi-Stage Key Exchange and the Case of Google's QUIC Protocol," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 1193–1204.

[20] G. Carlucci, L. De Cicco, and S. Mascolo, "HTTP over UDP: an Experimental Investigation of QUIC," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. ACM, 2015, pp. 609–614.

[21] R. Lychev, S. Jero, A. Boldyreva, and C. Nita-Rotaru, "How Secure and Quick is QUIC? Provable Security and Performance Analyses," in *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE, 2015, pp. 214–231.

[22] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar *et al.*, "The QUIC Transport Protocol: Design and Internet-Scale Deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 2017, pp. 183–196.

[23] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[24] K. Toyoda and I. Sasase, "Secret Sharing Based Unidirectional Key Distribution with Dummy Tags in Gen2v2 RFID-enabled Supply Chains," in *RFID (RFID), 2015 IEEE International Conference on*. IEEE, 2015, pp. 63–69.

[25] Q. Li, Z. Sun, J. Huang, S. Liu, J. Wang, N. Yan, L. Wang, and H. Min, "Secure UHF-RFID Tag for Vehicular Traffic Management System," in *RFID (RFID), 2017 IEEE International Conference on*. IEEE, 2017, pp. 26–29.

[26] K. S. Kadambi, J. Li, and A. H. Karp, "Near-Field Communication-Based Secure Mobile Payment Service," in *Proceedings of the 11th international Conference on Electronic Commerce*. ACM, 2009, pp. 142–151.

[27] M. Pasquet, J. Reynaud, and C. Rosenberger, "Secure Payment with NFC Mobile Phone in the SmartTouch Project," in *Collaborative Technologies and Systems, 2008. CTS 2008. International Symposium on*. IEEE, 2008, pp. 121–126.

[28] U. B. Ceipidor, C. M. Medaglia, A. Marino, S. Sposato, and A. Moroni, "KerNeeS: A protocol for mutual authentication between NFC phones and POS terminals for secure payment transactions," in *Information Security and Cryptology (ISCISC), 2012 9th International ISC Conference on*. IEEE, 2012, pp. 115–120.

[29] D. Sethia, D. Gupta, T. Mittal, U. Arora, and H. Saran, "NFC Based Secure Mobile Healthcare System," in *Communication Systems and Networks (COMSNETS), 6th International Conference on*. IEEE, 2014, pp. 1–6.

[30] A. J. Jara, M. A. Zamora, and A. F. Skarmeta, "Secure use of NFC in medical environments," in *RFID Systems and Technologies (RFID SysTech), 2009 5th European Workshop on*. VDE, 2009, pp. 1–8.

[31] C. Busold, A. Taha, C. Wachsmann, A. Dmitrienko, H. Seudié, M. Sobhani, and A.-R. Sadeghi, "Smart Keys for Cyber-Cars: Secure Smartphone-based NFC-enabled Car Immobilizer," in *Proceedings of the third ACM conference on Data and application security and privacy*. ACM, 2013, pp. 233–242.

[32] P. Urien and S. Piramuthu, "LLCPS and SISO: A TLS-Based Framework with RFID for NFC P2P Retail Transaction Processing," in *RFID (RFID), 2013 IEEE International Conference on*. IEEE, 2013, pp. 152–159.

[33] M. Roland, J. Langer, and J. Scharinger, "Security Vulnerabilities of the NDEF Signature Record Type," in *Near field communication (NFC), 2011 3rd International Workshop on*. IEEE, 2011, pp. 65–70.

[34] S. Podlipnig and L. Böszörmenyi, "A Survey of Web Cache Replacement Strategies," *ACM Computing Surveys (CSUR)*, vol. 35, no. 4, pp. 374–398, 2003.

[35] T. Korak, T. Plos, and M. Hutter, "Attacking an AES-Enabled NFC Tag: Implications from Design to a Real-World Scenario," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 2012, pp. 17–32.