Protecting NFC Data Exchange against Eavesdropping with Encryption Record Type Definition

Sufian Hameed
IT Security Labs, NUCES, Pakistan
Email: sufian.hameed@nu.edu.pk

Usman Murad Jamali
IT Security Labs, NUCES, Pakistan
Email: k133013@nu.edu.pk

Adnan Samad
IT Security Labs, NUCES, Pakistan
Email: k133014@nu.edu.pk

***Abstract*—Near Field Communication (NFC) is inherently vulnerable to eavesdropping and proximity hijacking attacks. NFC standards itself lack built-in security features against eavesdropping for all the modes of communication in NFC-ecosystem. This drives the application developers to implement customize security features on their own. These non-standard solutions in turn result in the system's security against vulnerabilities being subject to the developer's capability of designing a secure solution. Clearly, this model is a limiting factor in the widespread adoption and deployment of NFC applications.**

In this paper we propose a standard Encryption Record Type Definition (ERTD) to provide confidentiality to NFC Data Exchange format (NDEF). Subsequently, we develop a fully compliant prototype of our ERTD as a lightweight plug and play confidentiality middleware in the existing NFC communication architecture. Finally, we perform an in-depth performance evaluation, of different confidentiality related primitives that focuses on processing latency and data overheads.

***Index Terms*—Near Field Communication, NFC Security, Encryption Record, NFC Encryption, Tag Confidentiality**

I. Introduction

Near Field Communication (NFC) is a standards-based, wireless communication paradigm that allows NFC enabled devices to establish communication by simply coming into close proximity with each other. NFC devices can be used in arbitrary applications. Current and envisioned applications for NFC include contactless transactions [6], [11], retail [5], [23] and ticketing systems [26], data exchange, and simplified setups for more complex communications such as Bluetooth and Wi-Fi.

The proliferation of NFC technology has also increased the threat of abuse and security vulnerabilities. NFC alone does not ensure secure communications. Due to this short coming, attackers can easily exploit the NFC technology with eavesdropping or proximity hijacking attacks. With eavesdropping, an attacker can listen into NFC transactions and access victim's credentials such as his financial and personal information and use it for malicious intent [20]. For close proximity communication paradigm like NFC, distance is considered as a defense mechanism to limit the eavesdropping attacks. However, recent studies show that it is possible to tweak the devices using a dedicated amplifier and antenna to increase the active read range somewhere between 25 cm to 50 m [14]–[16], [19]. If an attacker is capable of eavesdropping the communication from 10 m, he could sit in his car outside any local corner store and record all the transactions conducted inside.

In this paper, we propose Encryption Record Type Definition (ERTD[1]) to provide confidentiality to NDEF message. We designed ERTD to provide data confidentiality in NFC applications. This will guarantee that no one can eavesdrop on the data held on the tag or exchanged during interactions between NFC devices. Further, we develop a fully compliant prototype of our ERTD as a lightweight confidentiality middleware in the existing NFC communication architecture. The confidentiality middleware implements standard confidentiality primitives needed by a wide array of applications. The confidentiality middleware/framework for NFC is lightweight and plug-n-play. It acts as an intermediate layer between the data read and execution of desired functionality to ensure protection against eavesdropping or proximity hijacking of the NDEF message.

We extensively evaluated the performance of the ERTD based confidentiality middleware. Based on different experiments, we concluded that all the standard symmetric and asymmetric ciphers defined within the ERTD have reasonable storage overhead. Further, even in the worst case scenario of resource intensive primitives, the overall processing delays remained below 60 ms, which is also perceived as instantaneous [18]. These overheads are also suitable in the time critical context of EMV smartcards, where the transactions must be done within a couple of hundred ms.

The rest of the paper is organized as follows. §II describes the state of the art. In §III we discuss the design of the ERTD. §IV discuss how the ERTD is realized and incorporated with the existing NFC reader and writer applications. In §V we demonstrate the performance and overheads of the proposed ERTD. Finally we conclude the paper in §VI.

II. Discussion on State of the Art

This section discusses existing work that highlights practical real life attacks and different security solutions in NFC based applications put forward by the research community.

[1]ERTD, Encryption RTD and encryption record will be used interchangeably.

NFC based applications are potentially vulnerable to various attacks without proper protection and normal user will not be able to distinguish malicious/forged tags from genuine ones [21]. In [8], Haselsteiner et al. highlights different threats, for instance eavesdropping, data corruption, man-in-the-middle attack, data insertion and data modification, which can compromise NFC interactions.

Collin Mulliner [22] was among the first few security researchers to analyze a number of previously unknown vulnerabilities. These vulnerabilities can easily be exploited for spoofing of the tag content. Shortly after Mulliner published the tag spoofing attack, the NFC-Forum released a specification of Signature Record Type Definition (SRTD) [3]. The specification adds digital signatures to provide authenticity and integrity to the NFC Data Exchange Format (NDEF) messages [2].

Roland et al. [24] analyzed the potential of SRTD and emphasize that digital signatures must be used in a proper way and that the signature is only useful for some elements of an NDEF record. In [25], Roland et al. further investigates the security of SRTD and introduced "*Record Composition Attack*". In [13], the authors proposed extensions in SRTD specifications to incorporate a larger and more flexible combination of standard signature/certification primitives. This also includes HMAC based authentication code, which was previously ignored altogether.

Being a wireless data transmission standard, NFC is inherently prone to eavesdropping. Standards body ECMA International has published NFC-SEC standards (ECMA-385 [9] and ECMA-386 [10]) to ensure the security of NFC communication. NFC-SEC provides the communication between NFC devices with confidentiality and integrity using Elliptic Curve Diffie-Hellman (ECDH) protocol for key agreement and the AES algorithm for data encryption and integrity. NFC-SEC is a link layer standard that enable two NFC devices to establish a secure communication channel in peer-to-peer mode of communication. This means that NFC-SEC cannot be deployed against the other two modes of communication in NFC i.e. reader-writer mode and card emulation mode and they still remain vulnerable to typical security threats. NFC-SEC also have a very limited set of cryptographic primitives with no flexibility for different application requirements, and the actual implementation and evaluation of the confirmed primitives is outside the scope this standard.

SRTD only deals with the integrity of NDEF message and the NFC forum do not provide any standard record type specifications for data confidentiality in all the three different modes of NFC communication. This drives the application developers to come up with confidentiality related solutions that are customized to their requirements [7], [17] only. Lack of specification sometime also results in poor design of critical applications. NFC based contactless payment card is one such example where no encryption or authentication is applied to transmitted data. In [20] Renaud Lifchitz, a researcher from British Telecom (BT), developed a proof of concept Android and Desktop application to launch a proximity hijacking attack against NFC-based credit cards.

III. Encryption Record Type Definition (ERTD)

This paper proposes ERTD in an attempt to standardize the confidentiality related operations during NFC interactions. The ERTD aims to provide confidentiality to NDEF messages and enable users to encrypt/decrypt sensitive data within an NDEF message. The record type lists encryption algorithms that can be used. In its current form the ERTD supports AES, RC4, Salsa20, elliptic curve (EC) and RSA as symmetric and asymmetric encryption algorithms.

The ERTD is like any other NDEF record and easily blends within standard NDEF messages[2]. We have design the structure of ERTD similar to signature record type [3] in anticipation that both SRTD and ERTD can be combined to provide integrity along with confidentiality. Fig. 1 outlines the complete layout of a typical ERTD, which consists of three parts in the payload of the record: version information, key ID and an encryption type. These parts are discussed in the following subsections.

A. Version field

The *Version* field specifies the version of the ERTD and occupies 1 byte of space. Since this is the first proposal of ERTD, all the acceptable ERTD compliant implementations must have this field set to a value of 0x01.

B. Key ID

The *Key ID* field is used to uniquely identify and fetch keys (both symmetric and asymmetric) from the key store while performing encryption or decryption on the NDEF record. *Key ID* is 8 bytes (64-bits) in length, which is reasonable enough for unique identification of keys. An example Key ID value in hex-decimal form could be something like 0x0FEA954E103B.

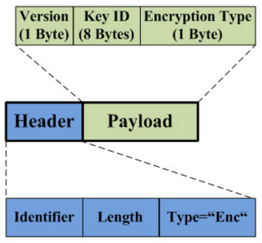


Fig. 1. Structure of a Encryption Record Type Definition (ERTD).

C. Encryption Type

Confidentiality of sensitive data in NDEF message is necessary to provide trust in the NFC ecosystem, where users connect over-the-air to unknown readers, tags and peers. The major challenge for NDEF message confidentiality is to make it as fast, easy and cheap as possible and still offer

[2]NDEF is a lightweight, binary message encapsulation format to exchange information between two NFC devices or an NFC device and a tag. In short, NDEF outlines the format and the rules for exchanging data structures through NFC. For details see [2].

TABLE I
PROPOSED VALUES OF ENCRYPTION TYPE FIELD

Hex	Encryption Type
0x00	No encryption present
0x01	AES_128
0x02	AES_256
0x03	RC4_128
0x04	RC4_256
0x05	Salsa20_128
0x06	Salsa20_256
0x07	EC_192
0x08	EC_256
0x09	RSA_1024
0x0a	RSA_2048
0x0b-0xff	for future use

sufficient security. NFC tags are constrained in memory and still are quite expensive. The choice of algorithms in ERTD has a major impact on tag memory usage, cost and device performance.

There are certain fields in the SRTD [3] that contains the actual signature along with the type of signature algorithm being utilized. Unlike SRTD, the ERTD do not contain any confidential data. It only maintains the reference of algorithms that are used to encrypt the payload of the preceding records. The *Encryption Type* in the ERTD is a 1-byte field used to identify the type of encryption algorithm being utilized (see Fig. 1). The listed encryption algorithm in the current ERTD proposal includes AES_128, AES_256, RC4_128, RC4_256, Salsa20_128, Salsa20_256, EC_192, EC_256, RSA_1024 and RSA_2048 with Hex identification of 0x01 to 0x0a respectively (also see Table. I). Values 0x0b-0xff is reserved for future use. AES_128 means that AES is used as an encryption algorithm with 128 bit key. Similarly, EC_192 means that elliptic curve based public-key cryptography is used to encrypt the record with 192 bit curve. *Encryption Type* can be further extended to support more flexible and lightweight range of encryption algorithms. At the moment we have restricted the proof of concept implementation to the above mentioned symmetric and asymmetric cryptographic primitives.

From the confidentiality point of view the *Payload* field in any NDEF record is the central element to be protected by the ERTD. This field contains the actual sensitive data which needs to be protected. We have left *ID* and *Type* fields in plain text for performance optimization, since they only provide category information and the attacker cannot infer meaningful information by eavesdropping on encrypted payload.

IV. ERTD MIDDLEWARE DESIGN AND PROTOTYPE IMPLEMENTATION

In this paper we designed and implemented a lightweight confidentiality middleware based on the proposed ERTD. We extended the idea of NFC security middleware first presented in [12] to protect the users from accessing the malicious or spoofed URIs in NFC tags.

The core objective of the confidentiality middleware, within the scope of this paper, is to ensure secure and confidential transmission of data in NFC communication. The design goal of this solution is that it should be lightweight in terms of encrypted data size, processing time, and support multiple symmetric and asymmetric cryptographic algorithms. Fig. 2 shows the basic communication architecture of NFC with the addition of confidentiality middleware.

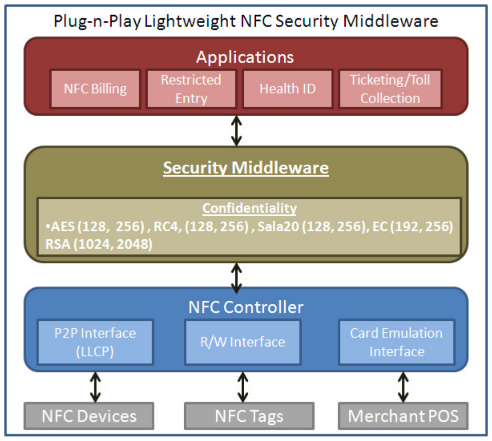


Fig. 2. NFC Application Architecture with Lightweight Confidentiality Middleware

The implementation of the proposed ERTD based confidentiality middleware is divided into two modules i.e. the writing module and the reading module. The writing module ensures that the NDEF message is written in the encrypted form in the NFC tag. Whereas the reading module ensures that the NDEF message is read and successfully decrypted before presenting it to the application. For better understanding we discuss the architecture of both modules separately.

A. NFC Writer

The *NFC writer* is responsible for encrypting NDEF record and constructing encryption record (ERTD). The encryption record includes the key ID and information about the chosen algorithms and formats. Once the encryption record is constructed, it is appended with the encrypted NDEF record, and the message is ready to be written to a tag. Different components of the writer module are discussed below (see fig. 3).

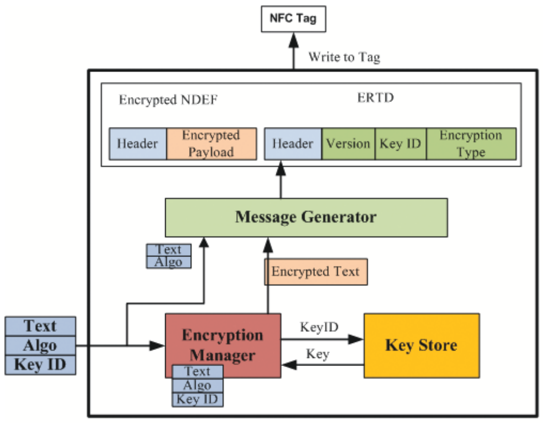


Fig. 3. Component Diagram of NFC Writer Module

1) Encryption Manager: The writing process starts with the user input. The NFC tag writer module accepts the user's input in the form of *type* of the NDEF record, *text/payload* for the NDEF record, desired *algorithm* to encrypt the payload and *identification for the key*. For symmetric encryption the key ID represent the pre-shared secret key and for asymmetric encryption the key ID represents the public key of the recipient. The writer module passes the user input (text, algorithm, key ID) to the *Encryption Manager*. Here, the *Encryption Manager* will fetch the encryption key from the *Key Store* using the key ID. The encryption key and desired algorithms are used to encrypt the input text. This encrypted text is then passed on to *Message Generator* for further processing. The user or application is free to select different families and variants of encryption algorithms listed in Table I.

2) Message Generator: Once the encryption process is finished, *type* of the NDEF record, encryption *algorithm*, *key ID* from original user input and encrypted payload/text from *Encryption Manager* are passed on to *Message Generator*. The *Message Generator* use this information to generate two records i.e. encrypted NDEF record using type and encrypted payload, and ERTD using key ID and algorithm information. The ERTD is appended with the encrypted NDEF record and written into the tag. In case the NDEF message consists of multiple records, a separate encrypted NDEF record will be generated for each of them.

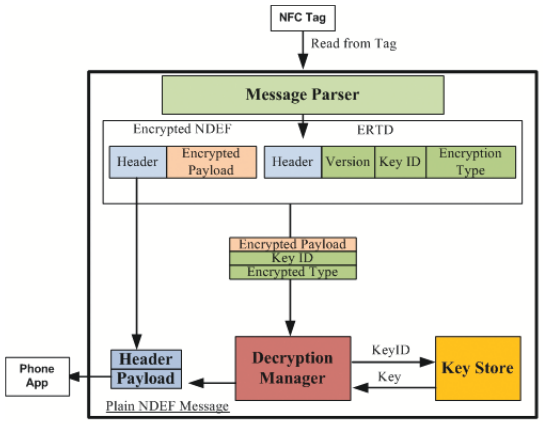


Fig. 4. Component Diagram of NFC Reader Module

B. NFC Reader

The *NFC reader* module allows the encrypted NDEF message to be read and decrypted from any NDEF message. The major components of the reader module are discussed below (see Fig. 4).

1) Message Parser: The *Message Parser* component is responsible for parsing different fields within a NDEF message. Primarily, it demarcates the NDEF and encryption record so that individual fields may be parsed. For example, the parser may parse the key ID and encryption type from the encryption record, and the encrypted payload from the NDEF record. The message parser, parses all the elements from the records and passes the relevant fields to other modules for processing.

2) Decryption Manager: The *Message Parser* passes the encrypted payload, key ID and encryption payload on to the *Decryption Manager*. Here, the *Decryption Manager* fetches the decryption key from the *Key Store* using the key ID. The decryption key and encryption type information is used to decrypt the payload.

The reader module use the type field from the *Message Parser* and decrypted payload from *Decryption Manager* to construct a plain NDEF message. The plain NDEF message is then forwarded to the NFC Application to perform the desired functionality.

C. Prototype Implementation

We have used Android platform (Eclipse IDE with Android SDK) for the development of the NFC confidentiality middleware. ERTD supports multiple cryptographic algorithms, therefore we have used Spongy Castle API [4] (Android version of open source Bouncy Castle API [1]), since it has support for almost all of the algorithms needed.

For the transparent operation of the middleware application, the following permissions and features is added to its manifest file.

```
<uses-permission android:name="android.
 permission.NFC"/>
```

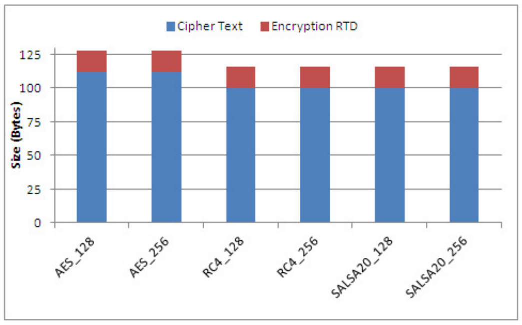
This will take permission from the user to read and write using NFC. This permission is essential as the middleware is intended to be the first point of access to the content fetched from the tags. By defining this permission in the manifest file the middleware can intercept all the NFC communication streams. The middleware application is programmed to start as soon as the Smartphone boots successfully and load all the necessary components in the starting. The service of the lightweight confidentiality middleware is invoked whenever there is any interaction between the phone and the NFC tags.

V. Evaluations

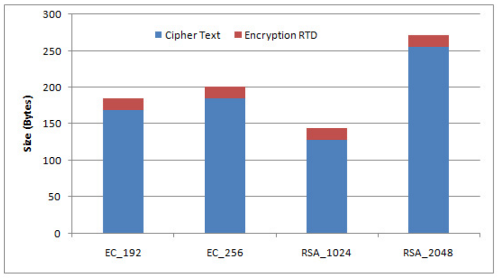
This section evaluates the efficacy of using encryption (confidentiality) in NFC communication. Our evaluations mainly focus on cipher text size overheads and added delays (processing time) incurred with the addition of the ERTD based confidentiality middleware. Any evaluation that involves interaction between a NFC tag and a Smartphone requires a lot of patience. The NFC tag is manually read and the evaluation reading is then noted by inserting measurement stamps at different function points. For all the experiments we used fix 100 bytes payload of the text RTD and any given measurement is an average of 10 readings. We used HTC One X: Android (4.1.1) Jelly Bean with 1 GB RAM and a 1.5 GHz Quad Core CPU.

Based on the performance metrics we have divided the evaluations in two parts. Part I covers the message size overhead using different cryptographic algorithms. Part II covers the processing time experienced in the generation of encrypted records at writer end and decryption time to get the plain text at the reader end.

A. Message Size Overhead



(a) Symmetric Cipher



(b) Asymmetric Cipher

Fig. 5. Size Overhead Comparison

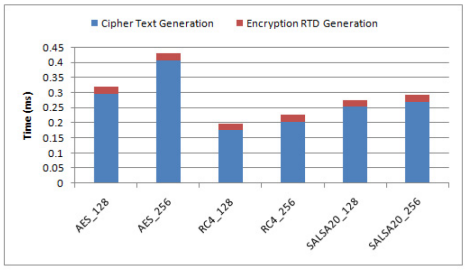
Based on the prototype implementation, we evaluated and compared the cipher text and ERTD size overhead of 10 cryptographic algorithms listed in Table. I. To evaluate overhead comparison we created one text record of 100 bytes and applied encryption on it using different cipher algorithm and appended the encryption record next to the text record using the prototype application.

Fig. 5 shows the size overhead comparison for overall encryption records using different algorithms. We have grouped the graph into two different families i.e. symmetric ciphers and asymmetric ciphers. The encryption record with 128 bit AES has a combined size of 128 bytes where 112 bytes belong to cipher text and 16 bytes of fix ERTD (see fig. 5(a)). This means that the overall overhead using AES with 128 bit key is of 28 bytes. This 28 bytes overhead is divided into 12 bytes cipher text overhead and 16 bytes of fix ERTD overhead. AES works on a block size of 128 bits or 16 bytes. For a text record of 100 bytes we had to append a padding of 12 bytes to make the input size of AES function multiple of 16 bytes. In worst case scenario, AES would have a cipher text overhead of 15 bytes. Stream ciphers (RC4, Salsa20) operate at bit level and have no cipher text overhead. They only have the constant 16 bytes overhead of ERTD (see fig. 5(a)).

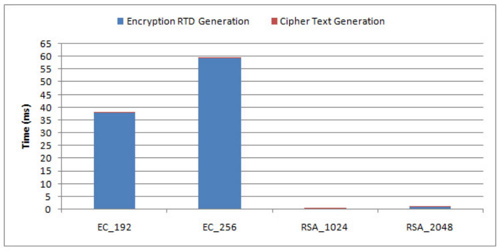
Fig. 5(b) shows the size overhead comparison of asymmetric ciphers. The cipher text overhead for EC_192 and EC_256 is approx. 69 bytes and 85 bytes respectively. Whereas, for RSA-1024 and RSA-2048 the cipher text overhead is about 28 bytes and 156 bytes respectively. The RSA encryption function $f(m)(e,n) := m^e mod n$ maps the inputs from Zn to outputs of Zn, where Zn ={0,...,n-1}. Therefore, the output is always an integer in Zn and if you use RSA with a 1024 key, the cipher text will be multiple of 128 bytes and for 2048 bit key the cipher text will be multiple of 256 bytes.

B. Processing Time

Processing time is another very important performance metric for evaluating the confidentiality middleware. We want to measure the extra processing delay incurred by the encrypted message and this processing time is measured separately for writing and reading of tag content. Again, we have grouped the graphs in two different families i.e. symmetric and asymmetric ciphers for better presentation.



(a) Symmetric Cipher

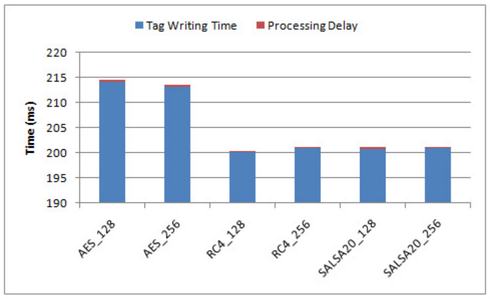


(b) Asymmetric Cipher

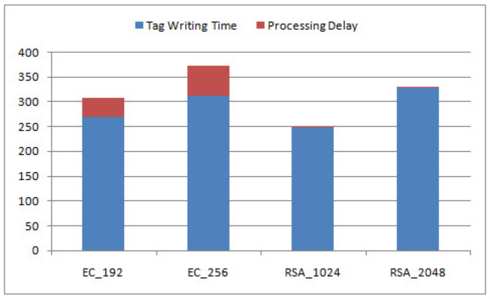
Fig. 6. Processing Delay for Encryption and ERTD Generation

1) Processing Delay at Writer End: Processing time for encrypting an NDEF message is the total time for applying encryption as well as constructing and appending the ERTD to the message before it is ready to be written to the tag. Fig. 6 shows the processing time for generating a secure NDEF message using different algorithms. For AES_256 the maximum processing time is approx. 0.428 ms. This includes cipher text generation as well as creation of ERTD. Fig. 6(a) shows that the majority of processing time is consumed in cipher text generation whereas the generation of encryption record takes approx. 0.022 ms. With RSA_2048 the maximum processing delay is approx. 1.23 ms (see fig. 6(b)). EC based algorithms are most expensive with maximum processing time of approx. 59.80 ms for EC_256 (see fig. 6(b)).

Fig. 7 shows the total time to generate and write an encrypted NDEF message on a tag. The time taken to write data on a tag is independent of our middleware prototype

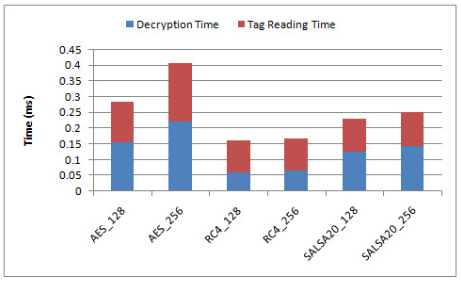


(a) Symmetric Cipher

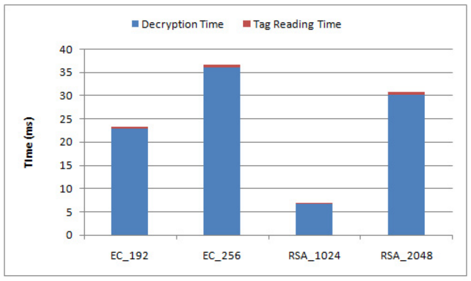


(b) Asymmetric Cipher

Fig. 7. Total Delay for Generating and Writing Encrypted Record



(a) Symmetric Cipher



(b) Asymmetric Cipher

Fig. 8. Delays for Decryption and Reading Encrypted Record

implementation. This time is totally hardware dependent and also depends on the size of the data in NDEF message that is going to be written. Since RSA_2048 has the biggest encrypted NDEF message, it takes the longest time i.e. approx. 328.36 ms to write the tag on the tag. On the other hand EC_256 takes approx. 311.92 ms. However, with highest processing cost, the EC based algorithms are on the whole most time intensive algorithm at the writer end (see fig. 7(b)). From the results, it is quite clear that the processing delays are negligible compared to actual time spent on writing the data to the tag. Here, writing delay is the real limiting factor, which inherently depends on the size of the message and due to this asymmetric ciphers suffer the most.

2) Processing Delay at Reader End: The processing time at the reader end starts with the parsing of the encrypted NDEF message from the tag, until the decryption operation is finished and the user can see the notification. Just like encryption process, we have performed all the experiments to evaluate the processing time for the decryption of tag content. Fig. 8 shows the processing time comparison for reading and decrypting the tag content using different algorithms. For AES based algorithms, the maximum processing time incurred in decrypting the tag content is approx. 0.184 ms for the AES_256 encryption type (see fig. 8(a)). If we add the time to read the tag content, the maximum reading time reaches 0.404 ms. If we consider asymmetric ciphers, the EC algorithms are most time intensive due to their computational complexity. For EC_256 the decryption and reading delay are 35 ms and 0.58 ms on average respectively (see fig. 8(b)). For asymmetric ciphers the decryption process consumes the maximum time at the reader end. Nevertheless, on average the total tag read and decryption time is less than 50 ms.

VI. Conclusion and Future Work

In this paper we proposed a new standard Encryption Record Type Definition (ERTD) to facilitate confidentiality related security services in NDEF messages. We also implemented an Android based lightweight middleware prototype of the proposed ERTD with standard symmetric/asymmetric primitives to provide efficient and flexible means of confidentiality for NDEF messages. In the end we performed a comprehensive performance study of different algorithms that focused on data overheads and processing delays.

Based on our evaluations, we observed that symmetric ciphers have the lowest storage requirement and processing latency. Further, asymmetric ciphers have relatively higher requirement for storage. They can be effectively used for 1 KB tags, but for very low end tag of couple of hundred bytes, asymmetric cipher won't be a perfect fit. The middleware is also lightweight when it comes to processing latency. Even with EC_256 the overall processing delays for encryption and decryption remained below 60 ms. For symmetric ciphers the delays remained under couple of milliseconds.

References

[1] Bouncy castle crypto api. http://bouncycastle.org/.
[2] Nfc data exchange format (ndef), nfc forum technical specification, rev. 1.0, jul. 2006. http://www.nfc-forum.org/specs/spec_list/.
[3] Signature record type definition, nfc forum technical specification, rev. 1.0, nov. 2010. http://www.nfc-forum.org/specs/spec_list/.
[4] Spongy castle crypto api. http://rtyley.github.io/spongycastle/.
[5] G. Broll, H. Palleis, H. Richter, and A. Wiethoff. Exploring multimodal feedback for an nfc-based mobile shopping assistant. In *5th International Workshop on NFC*, Feb 2013.

[6] U.B. Ceipidor, C.M. Medaglia, A. Opromolla, V. Volpi, A. Moroni, and S. Sposato. A survey about user experience improvement in mobile proximity payment. In *4th International Workshop on NFC*, March 2012.

[7] Sebastian Dunnebeil, Felix Kobler, Philip Koene, Jan Marco Leimeister, and Helmut Krcmar. Encrypted nfc emergency tags based on the german telematics infrastructure. In *Proceedings of the 2011 Third International Workshop on Near Field Communication*, NFC '11, pages 50–55, Washington, DC, USA, 2011. IEEE Computer Society.

[8] Haselsteiner E and K. Breitfu. Security in near field communication (nfc). In *Proceedings of the International workshop on RFIDSecurity*, RFIDSec '06, 2006.

[9] Standard ECMA-385. Nfc-sec: Nfcip-1 security services and protocol also approved as iso/iec 13157-1.

[10] Standard ECMA-386. Nfc-sec-01: Nfc-sec cryptography standard using ecdh and aes also approved as iso/iec 13157-2.

[11] Omkar Ghag and Saket Hegde. Article: A comprehensive study of google wallet as an nfc application. *International Journal of Computer Applications*, 58(16):37–42, November 2012. Published by Foundation of Computer Science, New York, USA.

[12] S. Hameed, B. Hameed, S.A. Hussain, and W. Khalid. Lightweight security middleware to detect malicious content in nfc tags or smart posters. In *13th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 900–905, Sept 2014.

[13] S. Hameed, U.M. Jamali, and A. Samad. Integrity protection of ndef message with flexible and enhanced nfc signature records. In *14th IEEE Trustcom/BigDataSE/ISPA*, volume 1, pages 368–375, Aug 2015.

[14] Gerhard P Hancke. A practical relay attack on iso 14443 proximity cards. *Technical report, University of Cambridge Computer Laboratory*, 59:382–385, 2005.

[15] Gerhard P Hancke. Practical attacks on proximity identification systems. In *IEEE Symposium on Security and Privacy*, pages 6–pp. IEEE, 2006.

[16] Gerhard P Hancke et al. Practical eavesdropping and skimming attacks on high-frequency rfid tokens. *Journal of Computer Security*, 19(2):259–288, 2011.

[17] E. Husni, K. Kuspriyanto, N. Basjaruddin, T. Purboyo, S. Purwantoro, and H. Ubaya. Efficient tag-to-tag near field communication (nfc) protocol for secure mobile payment. In *Proceedings of the Third International workshop on Near Field Communication*, pages 97–101, Nov 2011.

[18] Markus Kilas. Digital signatures on nfc tags. *Unpublished Master of Science). KTH Royal Institute of Technology, Sweden*, 2009.

[19] Ilan Kirschenbaum and Avishai Wool. How to build a low-cost, extended-range rfid skimmer. In *Usenix Security*, 2006.

[20] Renaud Lifchitz. Hacking the nfc credit cards for fun and debit. In the Hackito Ergo Summit, 2012.

[21] G. Madlmayr, J. Langer, C. Kantner, and J. Scharinger. Nfc devices: Security and privacy. In *Third International Conference on Availability, Reliability and Security, 2008. ARES 08*, pages 642–647, March 2008.

[22] Collin Mulliner. Vulnerability analysis and attacks on nfc-enabled mobile phones. In *ARES*, pages 695–700, 2009.

[23] Denise Paradowski and Antonio Kruger. Modularization of mobile shopping assistance systems. In *Near Field Communication (NFC), 2013 5th International Workshop on*, pages 1–6, Feb 2013.

[24] Michael Roland and Josef Langer. Digital signature records for the nfc data exchange format. In *Second International Workshop on NFC*, 2010.

[25] Michael Roland, Josef Langer, and Josef Scharinger. Security vulnerabilities of the ndef signature record type. In *3rd International Workshop on Near field communication (NFC) 2011*, pages 65–70. IEEE, 2011.

[26] R. Widmann, S. Grunberger, B. Stadlmann, and J. Langer. System integration of nfc ticketing into an existing public transport infrastructure. In *4th International Workshop on NFC*, March 2012.