Masterarbeit

Design and Implementation of a Secure NFC-based System for Mobile Electronic Coupons

Christian Lesjak

Institut für Technische Informatik
Technische Universität Graz

Begutachter:	Ass. Prof. Dipl.-Ing. Dr. techn. Christian Steger
Betreuer:	Ass. Prof. Dipl.-Ing. Dr. techn. Christian Steger
	Dipl.-Ing. Francesco Gallo (NXP Semiconductors Austria GmbH)

Graz, im August 2013

Kurzfassung

Near Field Communication (NFC) ist ein kontaktloser Kommunikationsstandard basierend auf RFID. Indem ein NFC-fähiges Gerät in die Nähe eines NFC Tags gehalten wird (< 10 cm), ermöglicht es benutzerfreundliche Interaktion. Ein NFC Tag ist ein kompaktes, relativ günstiges Gerät mit Speicher, welcher von einem Lesegerät im Reader/Writer Modus durch Erzeugung eines magnetischen Feldes mit Strom versorgt und gelesen wird. Ein verbreiteter Anwendungsfall ist das Smart Poster, wo ein Tag an eine Reklametafel angebracht wird. Bei Berührung wird eine URL aus dem Tag durch das NFC-fähige Smartphone ausgelesen und im Browser angezeigt. Dadurch können beispielsweise Eintrittskarten erworben werden, oder Gutscheine in Magazinen verteilt werden. Vor allem für Werbung und Loyalitätsprogramme eröffnen sich dadurch neue Anwendungsszenarien. Diese Diplomarbeit behandelt NFC Tag basierte Systeme, welche einerseits aus dem NFC fähigen Smartphone des Endanwenders, und andererseits aus NFC Tags und einem Server des Systembetreibers bestehen.

NFC wird allgemein als sicher wahrgenommen, u.a. wegen der kurzen Kommunikationsdistanz und dem Secure Element (SE). Nichtsdestotrotz sind Endanwender anfällig für Angriffe auf ihr Smartphone über die NFC Schnittstelle, welche zu Fehlfunktionen oder im schlimmsten Fall zu finanziellen Schaden durch Betrug (Phishing) führen können. Auch Systembetreiber müssen sich gegen Angriffe auf deren Systemverfügbarkeit, als auch gegen unautorisierten Zugriff auf ihre Infrastruktur absichern. Ansonsten könnte durch diese Schwachpunkte die NFC Technologie unnötig in Verruf geraten, oder unerwartete finanzielle Schäden auf Endanwender oder Systembetreiber zukommen.

Zunächst werden praktische und theoretische Bedrohungen gegen NFC Tag basierte Systeme untersucht. Darauf, und auch auf potentiellen zukünftigen Anwendungsfällen aufbauend, werden Sicherheitsziele identifiziert. Mehrere Systeme, fußend auf Zählern und kryptographischen Operationen in den NFC Tags, werden vorgeschlagen und evaluiert unter Berücksichtigung von Kosten, Sicherheit und Bedienbarkeit. Anschließend wird das Design und die Implementierung eines Systemprototypen beschrieben, der sichere, mobile, elektronische Gutscheine ermöglicht. Dafür werden u.a. implizite ECQV Zertifikate und digitale Signaturen eingesetzt. Der entwickelte Prototyp demonstriert, dass sowohl die Nutzerfreundlichkeit, als auch der Sicherheitsaspekt durch NFC erreicht werden können. Abschließend wird auf Implikationen für das Systemdesign und zukünftige Tag Produkte eingegangen.

Abstract

Near Field Communication (NFC) is a wireless communication standard that evolved from RFID. It advocates the intuitive user experience of a single touch by only requiring the user to bring an NFC enabled smart phone and an NFC tag closer than approx. 10 cm. An NFC tag is a small, rather cheap, unpowered device with memory, that is read by an NFC device in reader/writer mode by generating a magnetic field and powering a passive tag. A common use case is the smart poster, where a tag is attached to an object, e.g. a billboard. Upon touch, a URL stored inside the tag is read by the NFC-enabled phone, and the corresponding website is opened in the phone's browser. Example applications include smart posters to buy a sports ticket, or coupons distributed via a magazine. Especially for advertising and loyalty applications, tags promise a broad range of new business cases. This thesis investigates NFC tag based systems composed of NFC-enabled smart phones operated by users, and tags plus a backend information system operated by a system provider.

A common misconception about NFC assumes that the technology is intrinsically secure, inferred from its limited operating range, and the optional secure element (SE). Nevertheless, NFC tag based system are susceptible to a number of attacks, either targeted at the user of the system or at the system operator. End-users are prone to manipulation of their NFC-enabled phones or fraud, resulting in malfunction or financial loss. System providers require protection against attackers who disrupt service operation, or want to gain financial advantage or unauthorized access to provided services. All in all, system vulnerabilities related to the NFC technology may harm the reputation of NFC as well as dependent system providers, and could ultimately result in unexpected costs.

This thesis builds upon an investigation of theoretical and practical threats against NFC tag based systems. Based on these resulting findings, as well as the requirements of anticipated use cases, preferred security targets are identified. A number of system solutions, utilizing cryptographic methods and counters on NFC tags, are studied and evaluated in regard to cost, security and usability. Subsequently, the design and implementation of a mobile electronic coupon system prototype is described. The prototype utilizes ECQV implicit certificates and digital signatures, among others, to satisfy the specific requirements of coupon issuing, storage and redemption. The system implementation demonstrates that both, the usability aspect of NFC, and the necessary security mechanisms, are achievable in practice. Finally, implications on system design, and future tag product development, are given.

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

....

date (signature)

Acknowledgments

This master thesis was carried out during the first half of 2013 at NXP Semiconductors in Gratkorn, Austria. It was supervised by the Institute for Technical Informatics at Graz University of Technology.

At this point I want to thank Francesco Gallo from NXP. He initially invited me to conduct this thesis. I'm very thankful for the freedom he gave in respect of content and working style. Alike, I want to thank Hauke Meyn from NXP Hamburg. In numerous telephone conferences, he provided helpful input, and triggered new ideas. It was a pleasure to work with both of you in this company!

Also, I want to thank my supervisor Christian Steger from the institute, who guided me through the process of this thesis and supported me with input and hints.

Furthermore, I want to thank those persons at NXP, who in different ways assisted me with my work: Andreas Mühlberger, Oleksiy Kurbatov (JCOP support), Renke Bienert (discussions), Markus Gstöttner und Bernd Kaufmann.

Finally, I want to say thank you to my parents, who eventually made it possible for me to study at all.

Graz, August 2013 Christian Lesjak

Contents

List of Figures

List of Tables

1. Introduction

1.1. Motivation and problem statement

Near Field Communication (NFC) is a set of wireless communication standards that evolved from radio frequency identification (RFID). Its operating range is limited up to 10 centimeters, and communication is established by bringing two NFC Forum devices, or an NFC Forum device and and NFC Forum tag, in close proximity. The NFC Forum is an industry association that standardizes the technology and ensures compliance. It defines an NFC Forum device as a device capable of operating in reader/writer mode, in order to communicate with an NFC Forum tag. Such a tag is an unpowered NFC device that is not connected to a power supply in order to operate and contains a memory to be read via NFC. Also, contactless smart cards with a microprocessor and support for cryptographic operations are NFC tags, if they operate according to the tag operation specification. In order to access the data stored on the tag, which is usually encoded according to the NFC data exchange format (NDEF), the reading device (also called poller) generates an RF field, which powers the passive NFC tag (called listener or card). Other communication modes are the peer to peer mode, where two active devices communicate with each other, and the card emulation mode, where an active NFC device emulates a passive NFC tag.

The business relevance for the technology raises, as the number of end-user devices equipped with NFC technology is increasing. In 2012[1], an estimated 150 mil. NFC-enabled smart phones have been shipped. The total number is expected to more than double for 2013.

NFC technology advocates the intuitive user experience of touching. When a user brings its NFC-enabled smart phone in proximity of a tag, communication is established automatically. This principle is facilitated by the *smart poster* use case depicted in Figure 1.1, where an NFC tag is attached to an arbitrary object, e.g. a billboard, magazine or any other three dimensional object. The NFC tag has data in the form of an NDEF message stored inside, which carries e.g. a website URL. Upon touch, the address stored on the tag is read by the phone, which operates in the NFC Forum reader/writer mode. The received URL is then processed by the phone and an appropriate action is triggered, e.g. by accessing and displaying the corresponding web page in the phone's browser. Example applications include smart posters to buy sports tickets, or coupons distributed via a magazine. Especially for advertising and loyalty applications, NFC tags promise a broad range of new possibilities for businesses and retailers.

NFC tags are passive devices that are powered by the reader which generates the RF field. Usually, tags contain readable data, often encoded in the NFC Data Exchange Format (NDEF). This thesis investigates NFC tag based systems, where the smart phone operates in the reader/writer mode, communicating with a passive target, the NFC tag.

[1]NFC Times: "NFC smart phone chip shipments in 2012 surge past projections", `http://nfctimes.com/news/nfc-smartphone-chip-shipments-2012-surge-past-projections`, accessed on 2013-06-14

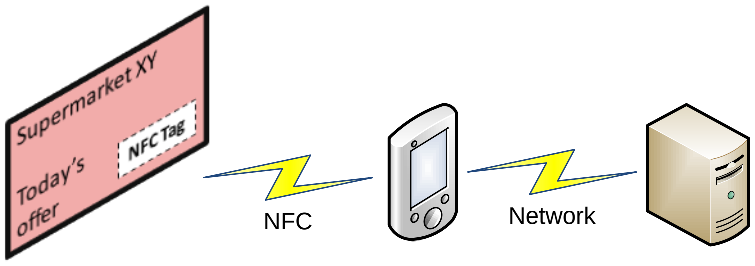


Figure 1.1.: The smart poster use case composed of the NFC tag attached to a billboard, the NFC enabled smart phone, and the backend system to which the content of the NFC tag relates.

Furthermore, a backend information system, accessible to the phone via Internet, provides a service to the phone. This service includes the delivery of web content (browsing), or other interactions with the backend system. The system provider operates the NFC tags and the backend system. Customers and end-users participate in the system using their personal, NFC-enabled smart phones.

Nevertheless, a common misconception about NFC assumes that the technology is intrinsically secure, mainly due to two reasons: Firstly, a Secure Element (SE) is often attached to an NFC interface in current smart phones. But in use cases that are based on reader/writer mode and NFC tags, such as the smart poster, no SE is utilized. Secondly, the short operating distance is expected to provide a certain level of security, implied by the proximity. Yet, NFC tag based system are susceptible to a number of threats, as the number of attacks is larger than just the attack of single NFC components. Attacks can be targeted at both, the users of the system, or the service provider itself. Users are prone to attacks that make their phones malfunction, or eventually result in financial loss due to threats like phishing. For service providers, system operation must be ensured, and attacks against the backend system prevented. Attackers might gain unauthorized access or financial advantage from cheating on the system.

Henceforth, this thesis focuses on the security aspect of the aforementioned NFC tag based systems. The central question is:

What *security targets* need to be defined, and in order to meet these targets, which *security mechanisms* and *NFC tag related features* have to be introduced?

A security target describes a generic assertion that a software component in the system can make regarding the communication. An example is the security target where the phone asserts that the content received from a tag is authentic and integer. Security targets are desired by system providers as well as end-users and they relate to the entire NFC tag based system, but not every specific business case requires all targets. The security mechanisms are composed of cryptographic schemes and protocols for data exchange in order to ensure one or more security targets. In order to carry out the mechanisms, necessary NFC tag features such as counters, need to be introduced.

Further, this thesis addresses the above question in the following context:

- Proposed or implemented systems shall comply to current and future (drafted) NFC Forum standards, in order to be compliant with the broadest possible range of NFC devices. Specifically, the smart phone shall operate in reader/writer mode, and only carry out commands to read and write NDEF messages from NFC tags.
- The prototype implementation shall consider the intuitive nature of NFC, and shall implicitly demonstrate the user experience of NFC. This aspect is important, as usability influences product adoption among end-users. The prototype shall demonstrate the feasibility of the security mechanisms in terms of usability.
- User privacy and control need to be accounted for, as smart phones are considered a highly personal device.
- The implemented prototype needs to promise a certain business benefit, in order to be attractive to prospective service providers.

1.2. Goal

The overall aim of this thesis is to design and implement a system and its corresponding components, which demonstrate a secure NFC tag based system for a loyalty application that enables electronic coupons. A generic library shall implement use-case agnostic functionality related to the security of the systems under investigation. The prototype shall then utilize this generic library. Furthermore, the system in its entirety shall demonstrate a specific business case, and account for both, security and usability aspects.

Therefore, a sub-goal of this thesis is to describe a security framework, which supports the design and implementation of NFC tag based systems that provide a certain security level, composed of individual security targets. To achieve this, an analysis of shortcomings in current systems and products needs to be analyzed. Furthermore, possible counter measures have to be developed, based on prior research and new technologies, and then have to be evaluated.

Finally, the thesis shall give a summary of tag features, in order to implement one or more of the proposed systems. Also, the implications discovered during the system design and the implementation of the prototype shall be examined.

To study the prospective systems and to design the prototype, this thesis focuses on the application layer and the corresponding cryptographic operations. It is neither the aim of this study to analyze the analog nor physical radio frequency characteristics of NFC. Also, the design of new cryptographic primitives shall not be carried out.

1.3. Outline

The thesis is structured in the following chapters:

Chapter 2 introduces the field of study and the state of the art. First, the technical background and respective technologies are given. Second, the related work is summarized in regard to security aspects of NFC technology, NFC based system solutions, and advertising and loyalty with NFC.

Chapter 3 builds a theoretical foundation for secure systems based on NFC tags. From the discovered flaws in the previous chapter, as well as from anticipated use cases, security targets and potential solutions are derived. Furthermore, evaluation procedures are introduced based on security targets and solutions. The described systems form a framework for a broad range of NFC tag based systems, and serve as a basis for the prototype.

Chapter 4 analyzes the requirements for a prototype implementation of a mobile coupon system. Subsequently, the system solutions from the previous chapter are evaluated and selected in respect to the specific prototype requirements. Finally, the software architecture and structure of the mobile coupon system is described.

Chapter 5 considers the implementation and realization of the demo system. An overview of the development environment and specific implementation issues, including their approached solutions, are given.

Chapter 6 evaluates and discusses the prototype implementation. Also, implications learned from the design and the implementation are given, which are further elaborated to future work.

Finally, Chapter 7 summarizes the results of this thesis and further work. In the appendix sample data structures, as generated by the NFC tags in the implemented prototype, are given.

2. State of the art

2.1. Near field communication

Near field communication (NFC) is a set of standards for short range wireless communication in the unlicensed ISM band, operating at 13.56 MHz with a limited range. Interoperability and compliance are ensured by the NFC Forum, an industry association founded in 2004 by NXP Semiconductors (and others) to promote the development and standardization of NFC technology. Use cases and applications of NFC advocate the paradigm of *touching*, where actions are triggered by bringing two NFC-enabled devices in close proximity (about 10 cm).

An NFC interface can operate in two modes, either as an active device (so called reader or poller), usually having a power supply, or as a passive device (so called card or listener), such as a contact-less smart card. The active device generates a magnetic field, which is used for data transfer and to supply power to the passive listener. In active communication mode, data is sent using amplitude shift keying (ASK). For a baud rate of 106 kBd, the Modified Miller coding scheme is used, for 212 and 424 kBd the Manchester coding scheme applies. In passive mode the data is always encoded using Manchester coding. NFC uses a message and reply concept, hence any device plays either the role of a poller, which sends a message, or card, which can only respond to messages. The reader must always be an active device. NFC communication does not have to be pair-wise; a poller can also talk to multiple listeners. All cards are enabled simultaneously, but the intended recipient card must be selected by the reader, and only the selected device is allowed to answer.

The NFC Forum defines three operating modes. In *reader/writer* mode, NFC communication takes place between an active and a passive device, where the reader powers the card by generating a radio frequency field. The *peer-to-peer* mode requires both devices to be active. In *card emulation* mode, an (normally active) device emulates a passive card to be read in reader/writer mode. The latter is mostly used in applications that require strong security, such as payment.

For a detailed description of the relevant NFC specifications on analog and digital layers, including international standards as well as vendor specific ones, refer to [25].

NFC Tags are passive devices, which either act as simple storage devices with internal memory, or also feature processing logic for data. Sophisticated contact-less smart cards, e.g. NXP SmartMX with JCOP, support also public key cryptography. NFC-enabled smart phones are able to communicate with these passive devices in the reader/writer mode, hence enabling a wide range of use cases.

The *smart poster*[1] constitutes a common use case based on the reader/writer mode. A readable NFC tag is attached to an object such as a billboard or a poster. The tag stores

[1]NFC Forum: Smart Posters. How to use NFC tags and readers to create interactive experiences that benefit both consumers and businesses, `http://www.nfc-forum.org/resources/white_papers/NFC_Smart_Posters_White_Paper.pdf`, accessed on 2013-07-10

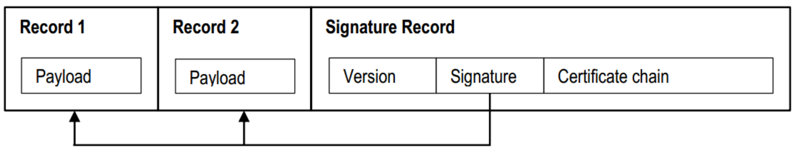


Figure 2.1.: A signed NDEF message composed of two arbitrary records and their payload, plus an additional signature record that protects the preceding records.

an NDEF message, which is read by an NFC device upon touch. The data contained is then processed by the reading device. An NFC enabled smart phone reading a URL from an NFC tag might open the corresponding web page in its browser.

NFC protocols. The initial absence of common standards, as well as the mission of interoperability, led to the foundation of the NFC Forum. It henceforth specified the protocols and defined a compliance program aiming at interoperability.

The NFC Analog Technical Specification [36] defines the RF interface for operation in NFC Forum reader/writer mode. The NFC Digital Protocol Technical Specification [34], and the NFC Activity Specification [33] describe frame formats and transmission handling. On top of these, the NFC Forum Tag Operation Specification depicts a set of operations and specific commands supported by NFC Forum Tags. An example of this specification is defined in [35] for so called Type 4 Tags. These specifications define for each tag type which specific commands need to be carried out by a reader in order to exchange NDEF messages with the tag.

The binary message format for application level information exchange between NFC devices is defined in the NFC Data Exchange Format (NDEF) specification [29] and called NDEF message. An NDEF message contains one or more NDEF records. Each record, as depicted in Table 2.1, carries application specific payload data described by type, length and an optional identifier. The payload type field indicates the data context to the processing application.

The record's type name field (TNF) specifies the structure of the payload type field. A TNF field value of 1 indicates an NFC Forum well-known type following the NFC record type definition [30]. The Smart Poster Record Type Definition [31] constitutes a widespread example how to encode URLs, SMSs, or phone numbers inside an NDEF message, using a TYPE field value of ”Sp”. Also, organizations can self allocate a name space in the TYPE field using the NFC Forum external type with TNF = 4. Furthermore, using TNF value 0x02, the type field contains a multipurpose internet mail extensions (MIME) media type. Further TNF fields are defined in [29].

The Signature Record Type Definition (RTD) [32] specifies how to protect one or more NDEF records against unauthorized modification by adding a signature record, which carries a digital signature and one or more corresponding certificates. Currently, version 2.0 is in candidate status [37], which fixes issues with version 1.0, and adds support for newer signature and certificate types, including ECQV implicit certificates. Figure 2.1 depicts an NDEF message composed of three records, where the last one carries a signature and a corresponding certificate chain to protect the first two.

`FIELD`	Description	Length
`MB`	The *message begin* flag indicates the start of an NDEF message, if set to 1.	1 bit
`ME`	The *message end* flag indicates the end of an NDEF message, if set to 1.	1 bit
`CF`	The *chunk flag* indicates a chunked payload that spans across multiple records.	1 bit
`SR`	The *short record* flag indicates the size of the `PAYLOAD_LENGTH` field: 1 byte if set to 1, otherwise 4 bytes.	1 bit
`IL`	The *ID length present* field indicates the presence of `ID_LENGTH` and `ID` fields.	1 bit
`TNF`	The *type name format* indicates the structure of the `TYPE` field.	3 bits
`TYPE_LENGTH`	Size of `TYPE` field in bytes.	1 byte
`PAYLOAD_LENGTH`	Size of `PAYLOAD` field in bytes.	1 or 4 bytes
`ID_LENGTH`	Optionally defines the length of the `ID` field in bytes.	0 or 1 bytes
`TYPE`	Contains the payload type definition, depending on the format given in the `TNF` field.	`TYPE_LENGTH` bytes
`ID`	Optionally defines an identifier for the payload.	`ID_LENGTH` bytes
`PAYLOAD`	Carries the actual binary payload data.	`PAYLOAD_LENGTH` bytes

Table 2.1.: The NDEF record layout, defined by the NFC Forum in [29].

Present NFC tag products. The NTAG21x[2] is the current generation of NFC Forum Tag Type 2 compatible ICs from NXP. The product enables the smart poster use case, and is targeted at applications such as advertising and loyalty applications, which specifically include vouchers and coupons. It features user memory sizes up to 888 bytes. This summary of present tag products uses this NXP product as a representative for the current generation of NFC tags.

A unique, seven byte UID identifies each tag, which is protected by a digital signature to prevent product counterfeiting. Using NXPs public key and a specific command, this signature can be verified (called originality check). As the signature must be calculated during production, the tag does not require cryptographic capabilities. But this also limits the security to the detection of mass cloned tags, where many copies using the same UID and signature are in circulation.

An internal NFC counter keeps track of the number of touches. Using this value, statistical analysis about tag usage can be recorded. In order to do this, the system operator can periodically visit the tag and read out the counter value. Alternatively, the "mirror" feature can be used to incorporate tag ID and counter value into the NDEF message served by the tag. By specifying the memory position inside the data format, a large number of tags can be easily serialized during production. Ultimately, if the NDEF message contains a URL, the following data construct can be served by the tag:

```
http://www.example.com/index.php?uid=AABBCCDD&counter=123456
```

A phone reading such a tag will get a distinguished URL upon each tag read. The tag ID and counter value will be transmitted to the web server upon each URL access. Both parameters are not protected by any cryptographic means. An attacker could access this URL, using arbitrarily modified parameters, to cheat on the backend system. Hence, this tag product cannot be used for applications that require security properties such as outlined in Chapter3, where the server needs a certain proof of tag touch. Furthermore, the dynamic nature of the counter does not allow the NDEF message to be signed upon production, as each value increment renders the signature invalid. Using a signed NDEF message based on the Signature RTD is henceforth not possible, if used with the counter.

For tag management, a password authentication mechanism is provided. A system operator using a specific tag management software can authenticate against the tag in order to set it read only, change its content, or access the counter (if not mirrored into the NDEF message). As this feature requires a secret that is not to be distributed to end users but must be kept confidential by the service provider, this functionality only supports the tag management, but does not give the user's smart phone means to securely interact with the tag.

[2]NXP NFC tag ICs: NTAG21x, `http://www.nxp.com/documents/leaflet/75017420.pdf`, accessed on 2013-06-24

2.2. Information security and cryptography

IT security[3]

> covers all the processes and mechanisms by which computer-based equipment, information and services are protected from unintended or unauthorized access, change or destruction.

Schneier [50] defines such an effective set of countermeasures to consist of three parts: protection, detection and reaction. For computer security, most systems employ prophylactic protection mechanisms such as cryptography or passwords. Some use detection mechanisms, and even less often deployed are reaction mechanisms, like locking a user account after a number of failed login attempts.

An attacker might have different motivations to attack a system. Finkenzeller gives a number of reasons related to radio frequency identification (RFID) systems [7], which have been adopted for NFC-tag based systems. The most common objectives are *fraud and phishing*, where either the system operator or other users of the system are being deceived. If the aim is to compromise the availability of the system, a denial of service (DoS) attack is carried out. Such an attack renders a system useless and might discourage further uses of a the system. The protection of a user's privacy could be another motivation to deceive an NFC-tag based system. Finally, *academia* continuously evaluates system and their security aspects and publishes them. Although these findings are often of theoretical nature, they might serve as a basis for attackers with other motivations, or could provide negative publicity for a product.

The digital, interconnected nature of the Internet increases the scope of attacks in several ways [51]:

- Class breaks: A line of products may have a common vulnerability. By breaking a single instance of a product, a whole class of products can be attacked with this exploit. An example is the software stack of networking routines in an operating system (OS).
- Automation: Attacks on information systems can be automated, which makes also attacks profitable where a single shot is of low value to an attacker.
- Action at a distance: If a system is connected to the Internet, an attacker must not be physically present to seize a system.
- Technique propagation: Only the first attacker must be skilled, every subsequent attempt by other defectors simply requires the software or exploit from the first attempt.
- Technique iteration and improvement: An attacker can learn from any other in the world, as exploits are often shared via the Internet.
- Defector aggregation: The Internet allows like-minded attackers to easily cooperate.

[3]Wikipedia: IT Security, `http://en.wikipedia.org/wiki/It_security`, accessed on 2013-06-14

To quantify security cost is simple, but the benefits of security are hardly visible. NFC-tag based systems are designed by NXPs customers. NXP offers the NFC-related IC components and provides its customers with support for design and implementation. But a bad system design by the customer might still hurt NXPs trustworthiness and reputation if the system solution exhibits security flaws leading to incidents or fraud.[4]

This thesis deals with security from a technical viewpoint, where the four core security principles [40] are:

- *Confidentiality* protects data from being read by unauthorized parties.
- *Integrity* allows to detect intentional or accidental modifications of data.
- *Authenticity* identifies the origin of data.
- *Non-repudiation* prevents the deniability of statements made by a party.

Furthermore, additional security properties might be desirable. *Availability* ensures the reliability of a system in order to provide its services. The protection of identity is described as *anonymity* or *privacy*. *Auditing* provides information about security related events by logging incidents. Finally, *physical security* provides defense against physical attacks to the components of the system infrastructure.

Public key cryptography with ECC. In a symmetric cryptography system, the same secret key is used for encryption and decryption of messages. Albeit these systems offer confidentiality, they cannot provide non-repudiation as the secret key is shared among sender and receiver. Furthermore, the distribution of secret keys requires a secure channel and the number of keys increases rapidly as each pair of users needs a separate secret key. [40].

Asymmetric cryptography operates with two distinct keys: a public key, which is not required to be kept secret, is used for encryption, and a secret private key is used for decryption. This is achieved by one-way functions, where it is easy to compute $f(x) = y$, but computationally infeasible to invert the function. The popular Rivest, Shamir and Adleman (RSA) public-key scheme is based on the difficulty of factoring of large integers.

Elliptic curve cryptography (ECC) is a public-key scheme that is based on the generalized discrete logarithm problem. An elliptic curve is a set of points that are solutions of the equation $y^2 = x^3 + ax + b$. Given a base point G (the domain), a private key d, and a public key $Q = dG$, it is computationally intractable to compute $d = log_G Q$.

Digital signatures and PKIs. A *hash function* $y = h(x)$ computes a fixed length digest of arbitrary length data using an irreversible procedure. The digest is always the same given the same input. Three properties need to be satisfied for such a function to be secure: a hash function must be preimage resistant, so that given $h(x)$, it is computationally infeasible to recover x. Furthermore, second preimage resistance prevents that given some data and its hash $y_1 = h(x_1)$, no x_2 with $h(x_1) = h(x_2)$ can be found. This means that two different inputs shall not produce the same output or digest. Finally, collision resistance

[4]NXP internal presentation by Philippe Teuwan.

denotes the computational infeasibility to find two inputs x_1 and x_2, so that $h(x_1)$ equals $h(x_2)$. Examples for currently used hash function include SHA-256.

A digital signature provides authenticity, integrity and non-repudiation for a message and requires public-key cryptography. A scheme to compute and verify digital signatures based on ECC, is the elliptic curve digital signature algorithm (ECDSA), specified in [4]. To sign a message, both the hash of the message and the private key of the sender, are used to calculate the signature value. The receiver uses the sender's public key, together with the received message and the signature value, to verify whether the message is authentic and integer.

A main application of digital signatures are digital certificates (or public key certificates), which bind identity information to a public key using the signature. A certificate is issued and signed by a certificate authority (CA). In practice, a root level CA issues certificates to sub-ordinate CAs (called intermediate CA), which themselves might issue certificates to CAs on another sub-level. A public key infrastructure (PKI) defines this hierarchy of trust, but also all means to generate, distribute, and store these certificates.

A common certificate scheme used on the Internet is X.509 [54]. The most important fields of such a certificate are:

- A *serial number*, which is unique for each certificate issued by the same CA. This number is used for revocation, in case a certificate needs to be declared invalid after e.g. the compromise of a secret, private key to unauthorized outsiders.
- The *issuer*, who generated and signed the certificate for the subject.
- The *validity period* defines the time frame within the certificate is valid.
- The *subject* identifies the entity to which the certificate was issued to.
- The *subject's public key* that corresponds to the secret, private key of the subject.
- The *certificate algorithm* specifies what signature algorithm was used to calculate signature value.
- The *signature value* protects all the other certificate fields above.

ECQV implicit certificate scheme. ECQV implicit certificates [23] are a novel approach to general purpose digital certificates, aimed at environments with constrained resources such as limited bandwidth or storage. A traditional certificate binds identity information to a public key, using a digital signature of an issuing authority. An implicit certificate replaces both, public key and signature, by a *public key reconstruction value*. The term implicit denotes the characteristic that the certificate alone cannot be validated. From the public key reconstruction value, the public key is reconstructed, without indicating the validity of the certificate. Not until upon first use of the key, such as verifying a signature over some data, the certificate correctness is proven (if the signature validation succeeds).

In order to achieve certificate sizes of about 60 bytes, compared to 300+ Bytes with traditional (X.509) certificates, two things are required:

1. a compact certificate encoding scheme, such as the Minimal ASN.1 Fixed-Length Encoding given in [23]; and

2. a method to generate the public key reconstruction value P.

The certificate encoding defines which fields are required, how to interpret these fields, and how to encode the fields as a byte stream for digital transmission and storage.

The ECQV scheme defines the setup, the key generation, and the public key extraction. During *setup*, all involved parties need to establish common parameters. The ECC domain parameters consist of q, a, b, and the base point G. Furthermore, a common hash function h, and a public/private key pair (Q_{CA}, d_{CA}) for the certificate authority (CA) need to be determined and distributed among all participants.

The *certificate generation* for a subject S includes the computation of the public key reconstruction value (P_S) to a corresponding certificate Cert_S, and the computation of the associated private key d_S. Therefore, the following steps are carried out:

1. The subject randomly generates an ECC key pair ($R_S = k_S \times G$), includes identity information, and sends $\{\text{ID}_S, R_S\}$ to the CA.

2. The CA generates the certificate:
   a) An ECC key pair $P_S = R_S + k \times G$ is randomly selected, where P_S denotes the public key reconstruction value.
   b) $\text{Cert}_S = \{P_S, ID_S, *\}$ encodes the certificate in the chosen format. $*$ denotes other fields, such as the validity period.
   c) The hash $e = h(\text{Cert}_S)$ of the certificate is computed.
   d) The integer $r = e \times k + d_{CA}$ is computed.

3. The subject receives $\{r, Cert_S\}$ from the CA and finishes the process:
   a) The hash $e = h(Cert_S)$ of the certificate is computed.
   b) The subject's private key is computed by $d_S = e \times k_S + r$.
   c) Using the public key reconstruction value P_S extracted from $Cert_S$, the subject public key is reconstructed: $Q_S = e \times P_S + Q_{CA}$, giving the private public key pair of the subject with $Q_S = d_S \times G$.

The *public key reconstruction* from a $Cert_S$, which is also given in the last step of the certificate generation above, starts with extracting the public key reconstruction value P_S from the certificate, and then calculating the public key using the hash of the certificate and the public key of the CA: $Q_S = h(Cert_S) \times P_U + Q_{CA}$.

Symmetric cryptography and message authentication codes (MAC). A message authentication code (MAC) is also known as a keyed hash function. For a given message of arbitrary length, the MAC represents a fixed-length, signature-like value that provides authentication and integrity for the message. Due to the fact that with symmetric cryptography all participating parties share the same secret key, MACs do not provide non-repudiation.

A specific MAC function, based on a hash function, is the keyed-hash MAC defined in RFC 2104[5]. To compute the HMAC, given message m, secret key k and hash function h, perform

[5]RFC 2103, `http://tools.ietf.org/html/rfc2104`, accessed on 2013-06-20

$$\mathrm{HMAC}(m,k) = h(k \oplus opad || h((k \oplus ipad)||m)).$$

where *opad* denotes a $0x36$ repeated n times, *ipad* denotes $0x5C$ repeated n times, and where n is the block length in bytes.

Cryptographic protocols. A cryptographic protocol defines the rules for exchanging messages between entities, and applying cryptographic operations, in order to perform a security related function, such as authentication. Kerberos [28] is a well known example for a network authentication protocol.

Two basic attacks on protocols are the replay and the reflection attack. In a replay attack, an adversary replays or delays information captured during a protocol. With a reflection attack, an adversary sends back captured message to the original sender.

For authentication, the principle of *challenge-response* is used to authenticate one party against another (or mutually). The idea is that the verifying party presents the authenticating party a questions (challenge), such as nonce. The other party answers this question (response), e.g. by encrypting the nonce with a private key. The authentication succeeds if the answer provided is correct, e.g. if the corresponding public key deciphers the nonce correctly.

A *nonce* is a value that is used no more than once for the same purpose. For challenge-response protocols, it provides freshness to prevent replay attacks. The numbers used must be selected from a sample space large enough in respect to the birthday paradox [24]. If a 64-bit nonce is used, a collision (generating the same number again) is expected after 2^{32} trials, which is sufficient to generate one number per second for over 130 years. Besides nonces, also counters or timestamp can provide freshness to a protocol.

The following three sections survey related work from three perspectives. At first, threats and attacks on NFC related applications are discussed. Secondly, various NFC based systems and ways to enhance security are being reviewed. Finally, loyalty systems and mobile electronic coupon systems are presented.

2.3. Related work on NFC security

A common misconception about NFC is that it is intrinsically secure, because communication takes place over a rather short distance. Furthermore, terms like *Secure NFC*, which actually mean *Secure smart card with NFC* [12], support this misbelief among end-users as well as implementors of NFC based system solutions. Furthermore, in such short range communicating systems as NFC, close proximity of devices is assumed while communicating, which is not necessarily the case, as relay attacks show.

A number of threats on systems that involve NFC have been identified in related research. Table 2.2 gives an overview about which threats relate to which components of the system. A clear distinction of attacks in regard to the affected component(s) is hardly possible. For example, an attacker might change the content of an NFC tag. After a reading device retrieves the manipulated content and processes it, it causes a software routine to malfunction. As a result, malicious code gets executed on the phone, ultimately installing a harmful application. This application might then result in unexpected cost for

threats	discussed in
NFC tag related	
clone	[7], [21]
destroy/deactivate	[52], [7]
move (detach or peel off)	[52], [7]
replace	[52]
manipulate tag ID or content	[52], [7]
invasive attacks: e.g. extract data (cryptographic keys)	[22]
NFC air interface between tag and reader	
eavesdrop	[12], [19], [7], [21]
long distance read	[7]
data corruption	[12]
block	[12], [7]
insert data	[12]
man in the middle	[12]
relay	[21], [58], [8], [10], [11]
NFC enabled smart phone related	
unintentionally modified software (malware on phone of user)	[52], [21]
intentionally modified software (on phone of attacker)	[52]
NFC stack software bugs, e.g. in parsing of NDEF message	[41], [25]
NFC data format design weaknesses, e.g. Signature RTD	[42], [45]
loss or theft of device	[52]
Attacks on the entire system, based on a combination of threats	
proof-of-concept NFC worm	[26]
denial of service	[26], [21]
remote exploitation	[25]
cheating on location-based-services	[55], [27]
phishing, e.g. by URI spoofing	[26]

Table 2.2.: Threats on NFC based systems.

the phone user, as it sends SMS to a costly premium rate number. This theoretical and rather complex example illustrates that an attacker needs to undertake a number of steps, effecting different system components:

1. Discover a flaw in the NFC stack that processes incoming data from NFC tags.
2. Find a way to exploit this flaw with manipulated tag content.
3. Distribute the malicious content, e.g. by replacing public tags.
4. Develop an application that is installed on the phone and contacts the premium rate number. Also set up the premium rate number to collect the charges.

Henceforth, first a look at specific, singular threats targeted at particular system components is given.

Attackers and their motivation. Before discussing threats, the potential parties and their motivations needs to be studied. Finkenzeller [7] identifies a number of reasons why RFID systems might be attacked, which include:

- Spying, where an attacker gets unauthorized access to information.
- Deception, where an attacker feeds incorrect information into the RFID system in order to deceive the system provider.
- Denial of service, where the availability of a system is being compromised.
- Protection of privacy, where the attacker considers the RFID system a threat to his privacy and wants to protect himself by attacking the system.

Finkenzeller numbers three involved parties: first, the system operator and secondly, the user of the system. In a closed system, these two parties are even considered to be the same, e.g. inside a factory. Additionally, a third party, the attacker (which can also be a competitor) of the system is given by the authors.

NFC tag related threats. Regarding the NFC tag, a number of physical attacks are possible, where an attacker mechanically alters a tag or its environment. In the simplest case, this includes the *destruction* or *deactivation* of the device. This especially applies to the smart poster use case, where tags are intentionally mounted at publicly accessible places. These threats harm the availability of the system. Furthermore, a tag might be physically *moved* to another location, by peeling it off its carrier object. In the case where tags mounted at bus stops are used to order tickets, an attacker might swap multiple tags. Henceforth, users order wrong tickets, as the context of the tags has been changed (their location). If tags are not set to read only, an attacker is able to *manipulate the content* of the tag. To circumvent this, tags are often protected against unauthorzied writing of content. But anyone, not just businesses or system operators, can buy NFC tags for a few Euros or less. This allows an attacker to acquire a number of empty tags. By shielding the genuine NFC tags using materials that block RF signals, and putting a new, manipulated tags thereon, users interact with non-genuine tags. Henceforth, setting tags to readonly does not prevent the *replacing* of NFC tags. The *cloning* of NFC tags is associated with copying the content or ID to another, previously empty tag. That way, a tag must not be physically moved, but just their content is copied to other tags.

Schoo and Paolucci identify security and privacy requirements of NFC based applications in [52]. They investigate the PERCI platform, where a tag is read and then used to get access to a service provided by a web server. Albeit this is a specific platform, the authors see this as general representative for systems using touchless technology to provision services to users.

By touching the tag attached to a poster, a user might be able to buy advertised tickets with his phone. Fraud is e.g. possible if an attacker *replaces the physical tags* attached to a poster. The manipulated tags point the user to a website where the money is not spent on the actual tickets, but collected by the attacker.

For RFID and NFC specific threats, the authors examine the subsystem tag and phone. The threat of *eavesdropping* can be mitigated by using tags with low reading distance and specific protocols. Another issue is *blocking or disturbing the communication*, for which the authors suggest physical protection such as metal frames on the back side of the smart poster. As for the tag itself, they suggest to only use read-only tags to protect them against changing the content or tag identity. Furthermore, physical protection means must hinder peeling off the tag from the poster.

Contactless smart cards that adhere to an NFC Forum tag operation specification can be used as NFC tags. Henceforth, the security of such cards needs to be considered as well. In [22], Markantonakis et al. give a summary about the security of smart cards. Smart cards can be divided into three types, in ascending order or their cost: memory cards only contain a preloaded value, such as an identifier that links to data in a backend system. More sophisticated memory cards with logic also include access control, e.g. to change their memory content. Microprocessor cards typically contain an operating system, support specific commands, and have a non-volatile memory, which can be modified by the card as required. If an attacker can physically isolate such a smart card, various attacks are possible. With *invasive attacks*, a smart card is being physically accessed and inspected, e.g. in order to reverse-engineer algorithms, or to read contents of otherwise unaccessable memory (e.g. secret or private keys used for cryptographic operations). Side channel analysis observes characteristics of a smart card while it processes data. Such side channels include timing information or power consumption. Fault analysis attempts to inject a fault, hence changing the functioning of the smart card.

NFC air interface threats. In "Security in Near Field Communication" [12], Haselsteiner and Breitfuß outline threats and possible solutions regarding NFC based applications. The paper focuses on the wireless communication link between two NFC interfaces. The authors identify eavesdropping, data corruption, data modification, data insertion and man-in-the-middle attacks as possible threats.

With *eavesdropping*, an attacker receives the transmitted signals and reconstructs the data, either by experimenting or literature research. NFC devices usually communicate in close proximity, with a maximum distance of about 10 cm. Due to a number of factors, including RF field characteristics and antenna properties, the distance varies greatly. As a rough estimation, the authors suggest that data transmitted in active mode can be captured up to 10 m, whereas with passive devices, this distance is reduced to 1 m. In a master thesis [19] by Kortvedt it has been demonstrated that passive communication can be eavesdropped up to 30 cm, using improvised antenna. As the only solution, Haselsteiner and Breitfuß propose the establishment of a secure channel.

With *data corruption*, an attacker may mount a denial of service attack by disturbing the communication; such that the receiving device can no longer interpret the data correctly. NFC devices should be able to detect this attack.

With *data modification*, an attacker wants the receiving device to obtain modified, but valid data. This attack is highly dependent on the strength of the amplitude modulation, but for certain codings, such as Manchester, it is possible. The best solution against this attack is to establish a secure channel.

With *data insertion*, an attacker inserts a message into the communication sequence. This is possible if the responding device requires a long time to answer. Again, a secure channel is suggested as a countermeasure.

With a man-in-the-middle attack, an adversary intercepts the communication link between two devices A and B and relays their communication via his own interface. The communicating devices still believe they directly talk to each other. Due to the implicit characteristics of RF communication, Haselsteiner and Breitfuß claim that "it is practically infeasible to mount a man-in-the-middle attack in real world".

The aforementioned secure channel can e.g. be established using Diffie-Hellmann based RSA or elliptic curve cryptography (ECC). Authentication is not necessary, as man-in-the-middle attacks are not a threat. The authors conclude that this secure channel is the only solution to protect against the outlined threats.

Relay attacks. As NFC technology operates on short range, the implied proximity property is often used for authentication in secure systems (e.g. payment, access control, ticketing), where some kind of security token is used to authenticate against a terminal. An attacker using a proxy token at the terminal and a proxy reader at a remote token, can trick the terminal into believing it communicates with the actually remotely located token. Successful relay attacks have been demonstrated in various papers [58, 8, 10]. To implement such an attack, consumer hardware like smart phones already suffice. As analyzed by [11], application layer security cannot protect against these attacks, as the actual challenge in the protocol is relayed to a legitimate (albeit remote) prover.

As countermeasures, Hancke et al. suggests three theoretical, but mostly impractical options, which were summarized from previous research:

- Tougher *timing constraints* reduce the attack feasibility. But the limits implied by current standards allow too much flexibility and hence allow to relay communication.
- *Distance bounding* uses cryptographic challenge-response round trip times to determine an upper bound on the physical distance between two entities. In practice, these approaches raise further security questions and practical issues.
- *Additional verification procedures* use human validation or two-factor authentication. Still these systems imply practical issues, like taking much more time, or being relayed as well.

In the paper ”Applying Relay Attacks to Google Wallet” [43], Roland et al. evaluate the feasibility of a software based relay attack in the existing mobile contactless payment system "Google Wallet". A relay attack can be simply seen as a range extension of contactless communication. Through nowadays widespread availability of NFC-enabled devices, practical demonstrations of such attacks become more frequent. The authors demonstrate a relay attack where a remote reader ("mole") is not a physical device, but a software running on a users smart phone. The user is unaware of this, and the software installs itself by using exploits to bypass operating system security restrictions. This malicious software directly communicates with the Secure Element of the user's phone and forwards the SE responses to a remote proxy (card emulator) at an POS terminal, where the attacker claims to pay. Even the process of entering a PIN has been successfully bypassed by the researchers. Using a PIN verification procedure, the access control for the secure element is delegated to a potentially insecure component (the application processor of the smart phone), where the application checks the users PIN and subsequently allows access to the SE. As the authors suggest, the PIN verification should take place on-card (within the SE).

Threats on NFC enabled smart phone. A main concern regarding smart phones is modified software. In case of an NFC tag based system, a system operator might distribute a specific application in order to allow user to consume the NFC based services. The application is responsible for communicating with the NFC tag, and subsequently with the backend system. But today's smart phone platforms can be compared to personal computers in terms of flexibility for software development and distribution. As an example, on the Android platform users can install any applications they want, distributed either via an official channel from the platform vendor (Google Play Store), or via any other channel as binaries. Furthermore, through a process called "rooting", user can gain system level access to any software or data stored on its phone. This contributes two threats:

1. *Intentionally modified software* A skilled attacker reverse engineers the official application. Through that, he eventually discovers either secret cryptographic material that is part of the application, or can find out how the application communicates with tag or server. The attacker can then develop a modified variant of the application, that does not behave as the system operator intended, in order to cheat on the system and gain access to services that should e.g. only be provided as a result of the touch of a specific NFC tag operated by the system operator.

2. *Unintentionally modified software* By bypassing smart phone security, an attacker might distribute a modified application to other, unaware users of the system. Then again, this application does disturb the intended operation of the system, and might either harm the users, or also the system operator.

Also [52] consider the manipulation of the client application a potential weak point, and hence suggest the certification of the installed software.

In "Effects of Android's SMS-URI parsing bug on NFC applications" [41], Roland describes an error in the parsing of NFC data exchange format (NDEF) records on Android 2.3.4, where a Smart Poster Record containing an SMS message and the recipient's phone number where misinterpreted by the built-in NFC handling operations. The issue however does not directly impose a security threat, but demonstrates how rather obvious implementation errors propagate into official versions of an operating system.

Roland et al. address in "Security Vulnerabilities of the NDEF Signature Record Type" [42] a number of weaknesses of a candidate version of the Signature Record Type Definition 1.0. Albeit currently version 2 is already in candidate status as of this writing [37], the discovered issues still are worth considering. To establish trust, it is not sufficient to verify the signature on some NDEF Record, which only establishes trust that the signer was in possession of any private key, and the content is unmodified since the time of signing. Only through certificates, which create a hierarchy of trust, and an ultimate root authority that is trusted, signed NDEF records provide a meaning of trust, and ultimately, security.

Furthermore, the standard allows individual records to be signed by different parties. As a consequence, an attacker might replace some records, and create an NDEF message that contains a trusted textual description of a URL, but add a malicious and unsigned

URL. Depending on how the reading device handles such contents, it might classify the tag trusted and access the attacker's URL. Hence the author's propose that "each and every record of a common context, like a smart poster, should be signed by the same party".

With the record composition attack, Roland et al. describe another attack that allows denial of service and fraud. The main idea is to use the fact that the Signature RTD (Version 1) explicitly excludes certain parts of the header of an NDEF record from being signed. Hence the authors propose that the message receiver should only trust the relationship of records if "they share a common signature record". A detailed description of the record composition attack is given by Saeed and Walter in [45].

Finally, the possibilities to include a URI pointing to a certificate (instead of an actual certificate), offers further options for attackers. As the URI is not part of the signed content, it can be arbitrarily modified. The certificate data that is being received can then be used to harm the system e.g. creating a buffer overflow. Additionally, the access of the URI might cause a privacy problem, as the user's identification data such as IP addresses can be collected upon access.

2.3.1. Discovering threats, and attacking entire systems by combining threats

This section discusses how threats related to NFC technology can be discovered, and how they are combined to facilitate attacks based on multiple threats.

Vulnerability analysis and attacks on NFC-enabled mobile phones. Mulliner describes in [26] a set of tools for security testing of NFC-enabled mobile phones and related software components; and introduces multiple attacks against NFC-enabled mobile phones based on these results. Also services deployed in practice are investigated.

For the analysis, a Nokia 6131 was taken into account. On the one hand, the reading, parsing and displaying of NDEF data was investigated, and on the other hand how components that are not part of the NFC system (e.g. web browser) can be controlled through the NFC interface. As no source code of the NFC related software implementation was available, Mulliner used fuzz testing methods[6]. Several bugs were discovered, which caused the phone to crash if certain fields had specific values (e.g. maximum length).

The first attack discovered was *URI spoofing* with smart posters. By inserting characters like space, tab or newline, the phone's GUI displayed the content only partially to the user, which could be exploited by an attacker to hide the actual malicious URL from the user and display only harmless parts for acknowledgement. Through that, the user could be directed to a web site looking like the one of his bank, but actually records the credentials of the user. Similar attacks can be carried out by spoofing the phone number for SMS messages, e.g. in cases where an SMS is used to order a ticket or get tourist information.

Secondly, a *proof-of-concept NFC worm* is presented. On an infected phone, an application intercepts all NDEF messages containing a URI. The worm spreads itself by writing a malicious URI, including the original one as a parameter, to the tag. Once an uninfected phone visits the URL, the worm gets installed. Furthermore, a cookie in the phone's

[6]Fuzz testing is an automated testing method that systematically provides random input to a device or software in order to discover implementation issues that yield to memory leaks or crashes.

browser is stored, so if an infected phone visits the URL again, it gets actually redirected to the original URL to keep the tag working and hide the worm from being detected.

Thirdly, simple *denial of service attacks* are demonstrated, which yield to destroying the trust relationship between customer and service provider. By sticking a tag with a malformed NDEF message (discovered by fuzzing) over the actual tag; the phone is caused to crash each time the user tries to access the service.

Furthermore, Mulliner examined a service deployed in real world: with certain Selecta vending machines, a customer can pay for snacks using his mobile phone. Each machine has an NFC tag, which contains an SMS message with the machine ID. After the customer has sent the SMS, the machine displays it is ready to dispense and allows the user to select an item (this happens via some backend system). The user chooses, and he gets charged for the snack. An attacker could place tags with machine ID X at machine Y and Z. Then he waits at machine X, until a customer at machine Y or Z sends the SMS and effectively pays for the good.

Exploring the NFC attack surface. Miller analyses in [25] the NFC attack surface by fuzzing the NFC protocol stack of three NFC handsets running Android and MeeGo. His findings are concerned with both, the low level protocol stack, as well as higher level attacks that utilize built-in applications of the phone.

On the lower level, the code responsible for handling NFC transmissions ranges from the kernel drivers, various NFC data services, and finally the application handling a certain content. Through fuzzing, crashes on Android 2.3.3 were generated. These resulted from various memory corruption issues. On the MeeGo 1.2 platform, no crashes could be produced by the author, which he claims may result from a flawed method or a rather robust implementation.

For higher level applications, the author suggests that "obviously, the browser represents an extremely large attack surface, and in Android version Ice Cream Sandwich, that attack surface is now available through NFC".

Miller concludes with a number of possible attacks, which he calls "*remote exploitation*". First, the Android NFC stack can be attacked to get control of the NFC service implementation. But through that, the attacker cannot gain internet access due to the OS permissions. Secondly, using manipulated tag contents, an attack could compromise the Android browser using a browser exploit to which a malicious URL points. That way, an attacker gets his code to be run in the browser context. Third, on the Nokia N9, an attacker can gain full access to the phone via Bluetooth, as per default the phone does not ask whether he wants to pair after touching a Bluetooth pairing tag. Finally, on the Nokia N9 an attacker might use the content sharing feature as an attack vector. After receiving a content, the device will try to parse and display that content. By exploiting bugs in PNG and DOC libraries, an attacker can execute code.

Practical attacks on NFC enabled cell phones. Verdult and Kooman demonstrate in [56] how the combination of NFC, Bluetooth, operating system, and software features in a Nokia 6212 can be exploited to install any kind of software on the users phone, requiring only the touch of a malicious NFC tag. The authors infer that this would allow the implementation of a worm that spreads itself by touching NFC phones.

NFC devices: security and privacy. Madlmayr et al. explore the security and privacy of NFC devices in [21]. First, the authors identify NFC uses cases and involved components. The use cases considered include the use of the unique ID; both the internal and external use of the secure element; data exchange via NFC; and the mobile phone reading an external tag. The host controller, together with the secure element and the NFC interface, are part of a GSM/UMTS handset. The identified threats, which are relevant to this thesis, are DoS attacks on the handset; the relaying of data transmitted via RF; and phishing attacks. The use of signed tags is recommended as a countermeasure for the latter.

2.4. Related work on NFC and internet based systems

This section describes related NFC tag based systems proposed in literature.

Trusted location based services

Teufl et al. describe a system for trusted location based services (LBS) based on NFC. They argue that current LBS mainly focus on improving the service quality for individual users by utilizing location information acquired and provided by a smart phone. But these approaches are limited to use cases where the user incentive to cheat the system is low (e.g. improved quality of provided service). Applications that require legal proof that a user was at an assured location at a certain time cannot be implemented with current methods. Smart phones are considered an insecure environment to generate location information (denoted Location-Time-Ticket), for two reasons: First, the location information obtained by a smart phone can be easily faked or manipulated (e.g. by malware). Secondly, the location information collected from e.g. GPS signals, cell-towers, or Wi-Fi access points can be modified by external parties. Thus, an approach incorporating a trusted third party (TTP) and electronic signatures is proposed.

Using a trusted third party that generates the LTT, and an electronic signature based on the Austrian Mobile Phone Signature, the system generates a trusted LTT (T-LTT), which fulfills a number of security properties:

1. T-LTTs can neither be forged by an attacker nor by a user of the system.
2. T-LTTs are unambiguously linked to the identity of the user.
3. T-LTTs incorporate a timestamp to bind location information to a certain point in time.
4. T-LTTs cannot be repudiated once successfully created.

Two variants are presented. In the first, another smart phone user acts as a TTP when generating the T-LTT. In the second variant, the TTP is composed of an cryptographic NFC tag and a web server. The key concepts in this paper are:

1. User credentials do not need be stored on the smart phone. The Austrian Mobile Phone Signature provides a signing service via a web service on demand.
2. A trusted third party generates and verifies the location information.

3. Electronic signatures by both, the user and the TTP, protect the information contained in the T-LTT from manipulation or unauthorized generation.
4. NFC is used to prove proximity of the user to a certain location (an NFC Forum Reader or NFC Forum Tag).

The system variant involving the TTP is depicted in Figure 2.2. The system is composed of the same components as are the subject of this study: an NFC tag (”Crypto Tag”), an NFC enabled smart phone (”Prover”), and a backend system (”TTP”). The process is initiated by touching the tag, from which the phone reads the UID. Using this UID, a nonce is requested from the TTP. The nonce is sent to the tag, which signs it using its private key. The resulting signature is presented to the TTP to prove the proximity of the Prover to the tag. The server issues location-time information, which is then signed by the Prover, and finally again by the server. Thus, a mutually signed T-LTT is generated, involving the TTP and the crypto tag.

The main limitations of the proposed system are relay attacks, which circumvent the necessary proximity of the user, and general issues with smart phone security (malware).

Enabling new mobile applications with location proofs

In [49] Saroiu and Wolman suggest that certain yet-to-emerge location based services might require a user to prove that he visited a certain location at a certain time in the past. Such systems include services that require evidence of repeated store visits, or services that need to limit digital content delivery to certain physical locations due to copyright laws.

To prevent users from cheating, a piece of data that certifies a geographical location is introduced. This proof, which is encoded in XML, consists of five fields: the issuer, the recipient, a timestamp, a location as longitude and latitude coordinates, and a digital signature. The issuing of location proofs is done by Wi-Fi access points (APs), which periodically broadcast beacons including a sequence number. A client in need of a location proof signs the sequence number to request a proof from an AP. The receiving AP checks for the sequence number signed by the client to be current, and subsequently issues a location proof.

The design fulfills four security properties. As location proofs are signed by the infrastructure, they are protected against modification (integrity) and forging. Furthermore, the identity of the requester is included to avoid transferring location proofs to other identities. Finally, a user’s privacy is preserved as a location proof is issued only on request. The infrastructure cannot monitor devices in range by the presented means.

The authors are aware that the identity scheme used matters, in order to discourage users from sharing their credentials or devices with others to impersonate other identities and hence cheat on the system. Finally, relay attacks (see 2.3) cannot be mitigated with the proposed system.

Off-line NFC Tag authentication

The NFC Forum Signature RTD provides authenticity and integrity for the content of an NFC Forum Tag by using digital signatures. Still, it does not protect from duplicating or cloning the contents of a tag. Even if a tag’s unique identifier is incorporated into the signed content, an attacker can still emulate the ID using a programmable product.

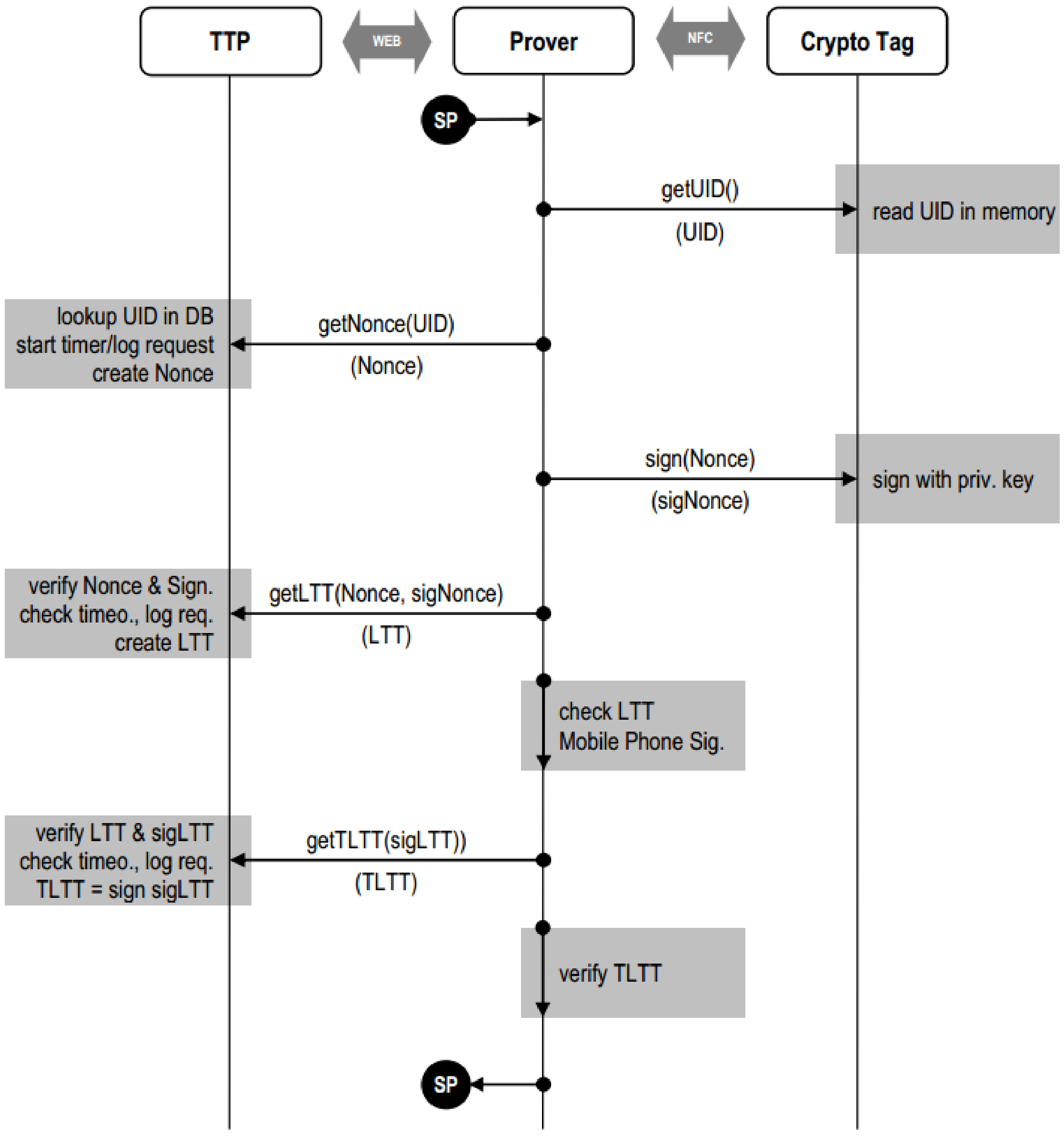


Figure 2.2.: The protocol proposed for trusted location based systems in [55].

The authors in [46] differ between two authentication scenarios: Off-line and on-line. With off-line authentication, there is no shared secret between reader and tag. Hence, a public key scheme is required. With on-line authentication, the tag shares a secret with a server. For authentication, the reader accesses the secret from the server to verify the tag's authenticity. An attacker cannot access the secret stored in the reader, as the authors assume the reader to be a closed system. Thus, in both scenarios cloned tags can be detected as the challenge-response protocol (see below) cannot be carried out successfully by just emulating the tag ID.

The proposed off-line authentication features three main components:

1. A challenge-response protocol, where the tag digitally signs a nonce received from the reader to authenticate itself. This idea is also known as active authentication (AA).
2. A PKI, that provides a trusted root anchor and verification scheme to avoid online connection.
3. A tag authentication record, which is proposed as a possible future NFC Forum well known type. The presence of this record on a tag indicates its capability of doing off-line authentication.

For the system to work, the authors conclude three essential requirements: Both, reader and tag are capable of performing public key cryptography operations. Also, a secret key can be securely stored inside a tag. And finally, the reader knows the public key of the tag to verify the signature.

From identification to authentication – a review of RFID product authentication techniques

In [20], Lehtonen et al. review and analyze existing approaches to authenticate RFID tags attached to consumer products. They state that a difference between label and product authentication should be made. The attacks on RFID based systems (essentially the counterfeiting of goods) are classified into four categories: First, the omission of security features, where faked goods simply do not even include attempts to fake authentication. Secondly, the use of misleading security features. The third category describes the removal and re-applicance of security features to counterfeit goods. The most advanced form of attacks includes the cloning and imitation of security features. The authors consider cloning and forgery resistance the most important security properties of RFID authentication products.

The analyzed approaches are split into four categories:

1. Serial numbering: Give items an identity and keep a catalog of valid, issued product IDs in some backend system. IDs must be randomly chosen from a large name space. Cloning cannot be prevented, but detection is possible.
2. Track and trace: This approach expands serial numbering by storing a trace record of a product during its move through the supply chain. Plausibility checks allow detecting to reason whether a product might be counterfeit or not.

3. Secure authentication: With this approach, cryptography is used to keep critical information confidential or to avoid cloning. Various systems are discussed, mainly using digital signatures, PRNGs, hashes, symmetric cryptography, or challenge response protocols. Some systems also mutually authenticate, or preserve the privacy of the consumer (using zero-knowledge proofs).

4. Product specific features: Here, a digital signature is written onto the tag. The signature authenticates the ID as well as a product specific feature such as physical or chemical properties (e.g. precise weight). As part of the authentication, the physical property is measured and must match the signed quantity.

All approaches provide a different trade-off between security and complexity, as well as cost and effort. Lehtonen et al. state three criteria how to evaluate RFID product authentication systems: Cloning resistance; the ability to detect cloned tags; and the resistance against the removal of the tag from the product. Also, they found out that current systems require an online backend system, and off-line authentication currently remains practically unresolved.

Switching the role of NFC tag and reader for the implementation of smart posters

Volland et al. propose a system [57] where the users are given identification badges augmented with NFC tags, and smart phones are attached behind the smart posters (instead of tags), to essentially switch the role of reader and tag. The approach yields at equipping visitors with cheaper tags instead of phones to tackle the problem of users who lack an NFC enabled handset.

All phones mounted behind the posters are connected via a wireless router with a backend system. Users are hence enabled to show appreciation for a certain smart poster by touching the location where the phone is mounted with their badge. An audio feedback indicates a successful vote, and the vote is transferred to the server. Via this backend system, the total count of votes for each smart poster are displayed at a central location. In set-ups where the number of smart posters clearly outnumbers the amount of users, this system idea can reduce cost while still equipping each user with the necessary NFC equipment (rather cheap tags).

Although the approach provides limited interaction possibilities for users, testers of the system perceived it as an interactive element that provided extra value to an actual event where the setup was assessed. Finally, usability is a critical criterion, in order for users to accept the system. An unambiguous, immediate feedback about a successful touch must be given, otherwise interest in the system is lost.

An NFC Ticketing System with a new approach of an Inverse Reader Mode

Saminger et al. describe in [47] an inverse reader mode, where the phone operates in reader/writer mode, and the ticketing system (specifically the turnstile) in card emulation mode. In a normal approach, the user would either use a contactless card or a smart phone with secure element (SE) in card emulation mode, and hold it against a reader at the turnstile.

The authors consider the reader/writer mode much more suitable for widespread use due to three reasons:

1. The owner of a secure element controls access for third party developers to the SE in a phone. In practice the SE is owned by the phone manufacturer, the network operator, or a trusted services manager. A third party must negotiate with the owner to get access. This cumbersome and costly process requires many applications, which would be suitable for the SE, to be implemented alternatively.
2. Reading and writing NDEF message in reader/writer mode is the most basic NFC operation that is supported by most smart phones. The NFC Forum recently standardized how application protocol data units (APDUs) can be exchanged by encapsulating them in NDEF message for personal health device communication.
3. Communication in reader/writer mode only requires two layers of communication, namely ISO/IEC 7816-4 and an ISO data exchange protocol such as ISO/IEC 14443-4. In comparison, peer-to-peer [38] mode requires three levels, on top either NPP or SNEP; then LLCP; and finally NFC-DEP.

The prototype implemented by the authors communicates with ISO/IEC 7816-4 APDUs. To build a command-response system on top, files for the parameters to be exchanged between phone and the NFC reader in card emulation mode were defined on the emulated card. To send a command from the phone, a dedicated file is selected and written, and to get a command's response, another dedicated file is selected and written. The authors state that the presented approach can be used for many applications besides ticketing, whenever peer-to-peer mode and/or a secure element is being used.

Secure smart poster

Gallo and Lesjak [9] describe a method for a secure smart poster system, composed of an NFC tag, a smart phone and a backend system. The patent application describes various methods to achieve enhanced security in the common smart poster use case. The communication protocol between the three components is depicted in Figure 2.3: Upon a NDEF read command, the tag generates and sends a specific NDEF message to the reading smart phone. This message contains a URL, which is accessed by the smart phone. The URL includes a number of parameters, which are encoded into the URL, e.g. by using percent-encoding. The parameters are used to transport information from the tag into the backend system upon access of the URL, such as the tag UID, and others.

In order to provide authenticity and integrity of the URL (and its parameters), a digital signature is included in the NDEF message served by the tag, which can be verified by the smart phone using a public key infrastructure.

A counter inside the tag is incremented upon each read access. This counter value is also encoded into the URL. On the one hand, this information can be used by the server to reason how many times a tag was read. Furthermore, the URL generated upon each access is different from all previous ones. Using this property, the server can allow only one request per URL read from the tag, and hence prevents sharing of a single URL read from the tag, with other users. Finally, the counter causes a new digital signature being computed on each access, hence making it difficult for an attacker to copy the tag without extracting also the protected, private key.

Although the digital signature allows the tag to reason about the authenticity and integrity of the URL, it is not transmitted to the server. Thus, the parameters of the URL

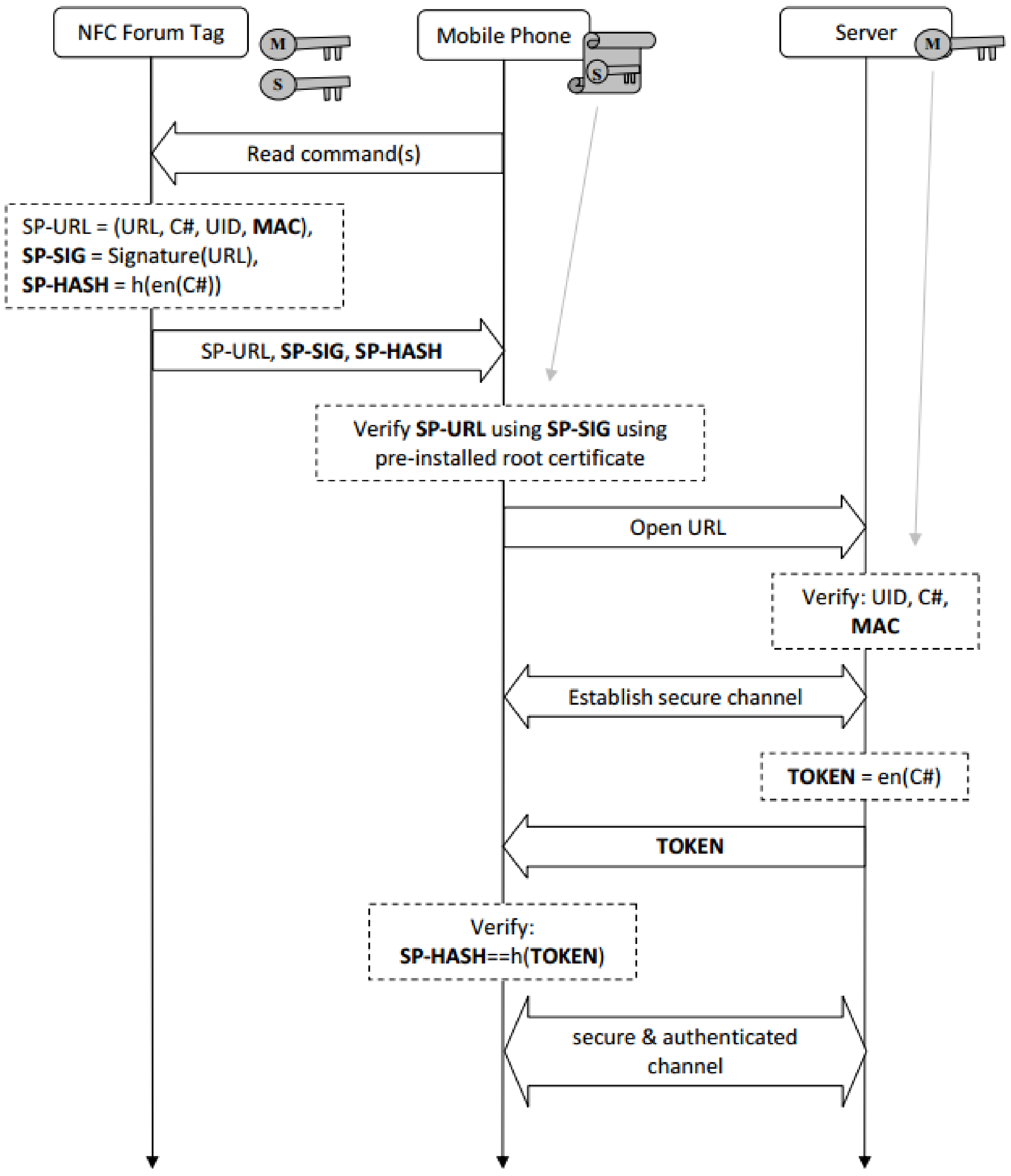

NFC Forum Tag
M
S
Mobile Phone
S
Server
M
Read command(s)
SP-URL = (URL, C#, UID, **MAC**),
SP-SIG = Signature(URL),
SP-HASH = h(en(C#))
SP-URL, **SP-SIG**, **SP-HASH**
Verify **SP-URL** using **SP-SIG** using pre-installed root certificate
Open URL
Verify: UID, C#, **MAC**
Establish secure channel
TOKEN = en(C#)
TOKEN
Verify: **SP-HASH**==h(**TOKEN**)
secure & authenticated channel

Figure 2.3.: The protocol of the secure smart poster patent application [9].

are not protected against manipulation before the address is being accessed. Therefore, a message authentication code (MAC) is also encoded into the URL parameters, protecting the counter value and the tag UID. A shared secret key between tag and the backend system (identified by the UID) is used by the tag to calculate the MAC, and by the server to verify the MAC.

Further methods to allow the phone to verify the server's authenticity, and other uses of the counter value are proposed in patent application.

2.5. Related work on advertising, loyalty and mobile, electronic coupons

This section introduces the field of advertising and loyalty, and describes proposed systems in that specific application field.

Advertising and loyalty programs

Advertising and loyalty programs are two means of marketing to communicate to existing or potential customers and influence their purchase behavior. In [13], a *loyalty program* is defined as

> any institutionalized incentive system that attempts to enhance consumers' consumption behavior over time beyond the direct effects of changes to the price or the core offering

Also [53] defines *loyalty programs* as

> structured marketing efforts which reward, and therefore encourage, loyal behavior.

Henderson et al. give a definition of *consumer loyalty* in [13] as

> repeat seller specific consumption that may result from desires to self-enhance (status-based), learned memory advantages (habit-based), or forward-looking desires to maintain a strong, committed social relationship based on trust and reciprocity with an organization (relational-based).

Using NFC in advertising is described in [14], where Holleis et al. attempt to find better connections between the user and the provided advertisements. They propose to utilize user profiles, generated by an application on the smart phones of the users, and the use of interactive displays. Such displays can either be static, like a smart poster, or dynamic with a touch screen to get user input as well. The main advantage of using the phone as carrier of advertising elements is that ads can be accessed on the smart phone even when the user is no longer at the display.

Holleis et al. conclude, that dynamic displays are suitable in supervised areas, whereas printed posters can be placed freely around in e.g a city, as they are far cheaper to produce and put up.

Wiechert et al. explore in [59] NFC based applications suitable for retail stores, and their effects on the customer shopping process. For businesses, the NFC related use cases can be split up in two categories: those that support shoppers while they are on the shop floor (download product information, collect coupons), and those that assist the checkout process (payment, collection or redemption of rewards from a loyalty program, coupon redemption).

A customer shopping process constitutes a number of phases. After a store is entered, a consumer either searches the store for a specific product or strolls through the store. After locating a product, he checks its features and decides whether to buy it. The customer might also consult a shop assistant for further information or to find a desired product. These steps might repeat, until the client finally proceeds to the checkout area. After possibly waiting in line, the products are placed on the counter. The cashier scans all product barcodes, and optionally takes a customer's coupons or loyalty card for identification. Finally, the customer pays using cash or electronic methods, and may get rewarded by the loyalty program.

To support the in-store experience, the authors suggest various NFC enabled use cases. Smart posters allow the access or download of information, such as rebate coupons. Tags embedded in product packaging or attached to the shelf trigger access to product specific information, instead of consulting a shop assistant. The checkout process may be supported by NFC based payment, loyalty applications, and electronic coupons. A customer's NFC device could combine all these plastic cards or papers into a single device, without the need to get these items out of the wallet.

Through interviews, the authors found out that supermarkets would profit most from a speedup of the checkout process. For department stores that focus on a specific product category, such as footwear stores, customers spend more time in the store and buy fewer products at once. The shopping experience is more considered a pleasure, and hence optimization of the checkout process is less of an issue than increasing the customer experience on the shop floor. All in all, the authors suggest that NFC technology will not fundamentally change the entire shopping process, but essentially support it.

Coupons

Coupons are a specific mean of advertising. Juniper research expects ten billion mobile coupons to be redeemed in 2013, 50 % more than in 2012[7]. According to Juniper, mobile coupons are redeemed at a rate of about 10 %, whereas classic print media coupons are below 1 %.

Chiou-Wei and Inman investigate in [3] the drivers and applicability of electronic coupons by a two year observation of Taiwanese shoppers. The authors confirm that shoppers actually like electronic coupons, especially well educated and employed ones. A number of managerial and marketing related implications are given.

Dickinger and Kleijnen investigate the intentions of users to redeem mobile coupons by surveying 370 mobile phone users in Austria [5]. The authors consider the mobile phone a highly personal device, and hence believe it to be an effective means to reach out to potential customers. The mobile device is used to retrieve and store mobile coupons, and the focus is set on coupons distributed via SMS. Furthermore, coupons may be differentiated by the way of their distribution: *push or pull* – either pushed by the company to the

[7]Juniper, `http://www.juniperresearch.com/viewpressrelease.php?pr=361`, accessed on 2013-04-22

customers to stimulate impulse purchases, or pulled by the customer as a consequence of interest in a product or offer.

A customer's attitude towards mobile coupons depends on two factors: the economic benefits, and the effort for redeeming the coupon. Hence, as companies cannot increase the value of coupons indefinitely, usability and handling on redemption are critical on implementing such a system. Furthermore, a user's privacy must be addressed. This means diminishing the fear of spamming, and reducing the perceived intrusiveness. To provide customers with a *sense of control*, an opt-in mechanism is suggested by the authors. Especially value seekers, due to their positive disposition towards coupons, are likely to be interested in mobile coupons.

Electronic coupon systems

In the literature, a number of schemes for electronic coupons have been proposed. This section summarizes related works for mobile electronic coupons systems.

Hsueh and Chen propose a mobile coupon scheme that utilizes peer-generated targeting using electronic word-of-mouth (eWOM) communication [17], in order to increase both, popularity and redemption rates of m-coupons. The simplest form of eWOM constitutes the forwarding of mobile coupons by users. Marketers address customers using individually (first-degree targeting), segmented (second-degree), or self-selected (third-degree) targeting. The proposed *peer-generated targeting* offers a new forth approach. While coupons transmitted from a merchant to a customer might be considered even spam, peer-forwarded coupons may be considered a recommendation instead.

The authors differ between electronic (e-) and mobile coupons (m-coupons). E-coupons are issued via Internet to customers, and must be printed out before visiting the store. M-coupons present the next evolutionary step, where the coupons are stored on the mobile device. Issuing takes place through four channels: Publishing by a mobile operator (e.g. SMS); downloading of coupons from a website; coupons transmitted via location based services (LBS); and finally coupons generated using RFID (e.g. smart poster).

The scheme proposed by Hsueh and Chen satisfies two important security properties. Double-redemption is prevented, and the unforgeability of coupons is achieved by detecting forged coupons on redemption.

In *An NFC-based solution for discount and loyalty mobile coupons* [48] the authors describe a platform for both, the management of coupons as well as advertising and market research. The demonstrated mobile coupon system offers a number of advantages to businesses: cost reduction, paper elimination, reaching of more customers, elimination of forgeries, real time tracking, market analysis and trend studies. Eventually, the systems helps businesses to increase customer loyalty. The focus of the system proposed and implemented is not primarily on security, but on the complete ecosystem. Coupons can be collected not just via NFC, but also via Bluetooth, peer-to-peer or from a web server. Acquiring the coupon from an NFC tag happens in either of two ways: (1) the complete coupon is contained on the tag, or (2) using the information provided by the tag the coupon is requested from the WingBonus web server.

Secure virtual coupons (mCoupons) using NFC technology

A system of secure virtual coupons, based on NFC technology, is presented in [1]. A further elaboration of this concept is given by the same authors in [6]. As the two papers are closely related, they're discussed in common.

Coupons are a well-established way of marketing. Although a coupon often represents a rather small value, a high number of uncontrolled copies can put a business at risk. Especially electronic coupons can be copied easily due to their digital nature. Hence, a secure system of protected electronic coupons is presented, preventing the following attacks:

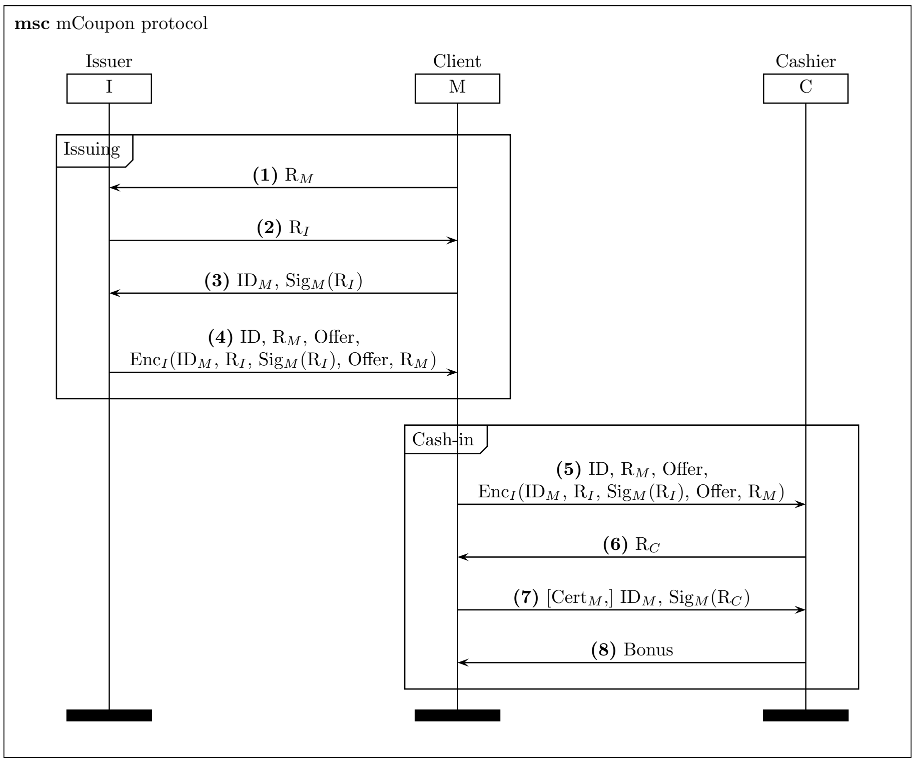
1. The *unauthorized generation* of coupons by an attacker.
2. The *unauthorized manipulation* of coupons by an attacker where a modified coupon can successfully be cashed in.
3. The *unauthorized copying* of coupons by the attacker where a legitimate copy can be produced and cashed in.
4. The *multiple cash-in* of coupons, where an attacker can use a single coupon multiple times.

System. The proposed system is composed of three parties involved. A *client* represents the user who collects and redeems coupons using his NFC-enabled smart phone. The *issuer* generates and hands out the coupons, and the *cashier* takes coupons for cash in. The latter is also responsible for checking the validity of the coupon. For a client to pick up a coupon, he touches a passive NFC-tag. Then a cryptographic protocol, based on challenge-response, is run, resulting in the final generation of a coupon. This issuing process only involves the client's smart phone and the tag, hence does not required online access. To cash-in the coupon, the client touches a NFC-enabled cash register. Again, running a cryptographic protocol, the coupon is redeemed. This process requires the cash register to be online to ensure certain security properties.

In general, symmetric key cryptography is used to ensure the integrity of a coupon, whereas asymmetric key cryptography and a public key infrastructure is used for client authentication. As the prevention of attacks 3 and 4 are only optional and depending on the system requirements, a simple coupon protocol (preventing attacks 1 and 2), and an advanced protocol are proposed (preventing all four attacks). As the simple protocol is just a subset of the advanced version, only the latter version is presented here.

Requirements. The client needs to have his smart phone registered with the service. By registering, he gets a private key and a corresponding public key certificate. The certificate is also stored by the cashier terminal for lookup on cash-in. Furthermore, symmetric key pairs need to be set up for each issuer tag with the cashier terminal.

Protocol. The protocol is depicted in Figure 2.4 and consists of the issuing and the cash-in use case. In steps (1) and (2), challenges R_M and R_I are exchanged. In step (3), the smart phone provides its identity ID_M and signs the received challenge using its private key to get $\mathrm{Sig}_M(R_I)$. The issuing tag cannot yet verify this signature (this is postponed until



msc mCoupon protocol
Issuer
I
Client
M
Cashier
C
Issuing
(1) R_M
(2) R_I
(3) ID_M, $Sig_M(R_I)$
(4) ID, R_M, Offer, $Enc_I(ID_M, R_I, Sig_M(R_I), Offer, R_M)$
Cash-in
(5) ID, R_M, Offer, $Enc_I(ID_M, R_I, Sig_M(R_I), Offer, R_M)$
(6) R_C
(7) $[Cert_M,]$ ID_M, $Sig_M(R_C)$
(8) Bonus

Figure 2.4.: The mCoupon protocol from [1] and [6], describing the issuing and the cash-in of secure electronic coupons.

cash-in), but it incorporates the proof into an encrypted part of the response in step (4), denoted $\text{Enc}_I(...)$. This encryption is done using a symmetric key shared between issuing tag and cashier terminal. Furthermore, information about the offer (Offer) is included. The complete data set received in this step denotes a coupon.

For cash-in, the smart phone sends the coupon received in step (4) to the cash-in terminal upon touch. The terminal challenges the client by returning a random nonce R_C. The client responds in step (7) with a signature of the nonce and identity information. The corresponding certificate might also be replied, or can be looked up by the terminal in its database using only the identity information ID_M. This check verifies that the same client, who initially got the coupon issued, is now redeeming the coupon. To ensure protection against double spending, the cashier terminal looks up in its database whether this client already cashed-in this offer. If the verification finishes successfully, step (8) informs the client about the bonus.

Lightweight protocol for secure e-coupons

The related work by Blundo et al. suggests a lightweight protocol for generation and distribution of secure e-coupons [2]. The authors consider coupons as a means of alternative payment form for (part of) a product's value. But the digital nature makes electronic coupons prone to fraudulent attempts of attacking the system.

The authors identify three different scenarios, involving the following entities: the manufacturers who want to advertise their produced goods, the retailers that sell promoted goods and accept coupons, the customers that collect and redeem the coupons, and finally the advertisers that distribute the coupons, e.g. through an online advertising campaign. In the simplest scenario, the retailer is both, issuer and redeemer of coupons, which simplifies the infrastructure required for validating coupons. Introducing an advertiser who is authorized to distribute the coupons, issuing and redemption are executed with different entities. In the so called extended model, manufacturers agreed with an advertising company to issued coupons, which are then cashed-in at different retailers selling the goods. In this case, a third use case besides issuing and redeeming coupons, the clearing, must be supported, where a retailer claims the granted discount back from the producer.

Furthermore, coupons are differentiated in static and in dynamic ones. A static one holds a fixed discount, valid within a fixed period. With a dynamic coupon, its value decreases with time, and thus pushes customers to redeem the coupon as soon as possible.

The requirements for their system are efficiency in terms of computational cost; ease of use for both, customers and system operators; and interoperability with existing online systems. Furthermore, a customers anonymity shall be preserved. From the security perspective, coupons shall be protected against unauthorized issuance or manipulation. Also, double spending is an undesirable property. From a cryptographic perspective, a message authentication code (MAC) is used to issue and protect coupons. The proposed protocol is depicted in detail in [2].

A prototype implementation by the authors uses a browser plugin to store the coupons on the user's computer. Issuing and redemption takes place on respective web sites.

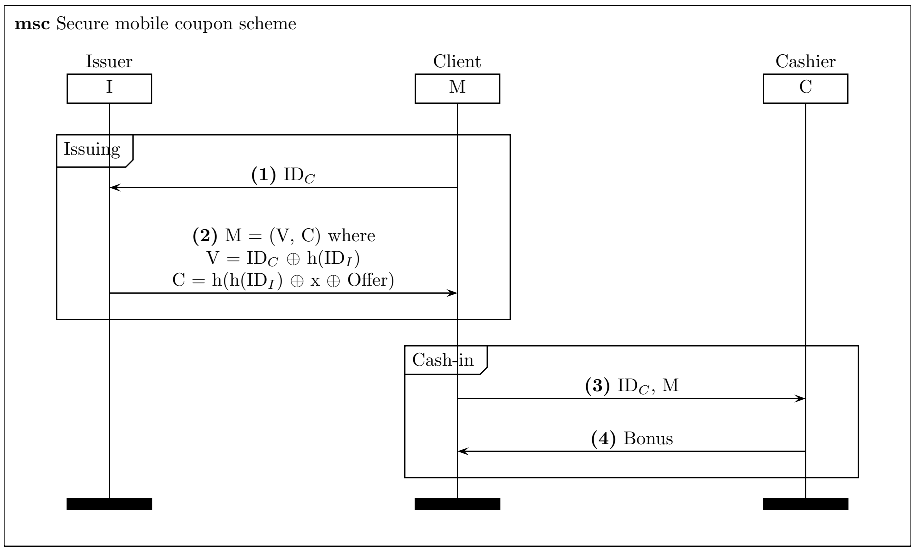


Figure 2.5.: The simple and secure mobile coupon scheme from [16].

A Simple and secure mobile coupons scheme

In [16], a modified version of previously discussed mCoupons scheme is presented, where only half as many messages need to be exchanged in each phase, issuing and redemption.

The protocol is depicted in Figure 2.5. On issuing, in step (1) the client presents the issuer tag its identity information. In step (2), the issuer incorporates this information together with the offer information and a secret key into the mobile coupon using a hash function. On redemption, the mobile coupon is sent to the cashier tag for verification step (3):

1. By xoring ID_C and V, $h(\mathrm{ID}_I)$ is calculated and used to look up the corresponding Offer and secret key x in its database.
2. ID_C and Offer is used to check whether this coupons has already redeemed by the current client.
3. The Offer itself is checked for validity (e.g. time period).
4. Finally, C' is computed analogous to C in step (2) and compared to C.
5. If all steps succeed, the redemption is recorded to the database and cleared (step 4).

Although stated by the author, the proposed scheme does not completely hinder multiple cash-ins or unauthorized copying. An attacker can easily get multiple coupons issued by writing a malicious application that always uses a different ID_C in step (1) (and consequently on cash-in in step 3). Furthermore, copying the coupon to a peer only requires to also hand over the identity information as well to allow cash in. Besides that, the protocol might satisfy all security requirements of secure electronic coupons based on NFC-technology.

3. Study of NFC tag based systems and evaluation of their security

This chapter builds upon the related work on NFC security discussed in Chapter 2, and derives some desirable properties that shall be achieved by a secure NFC tag based system. These properties are denoted security targets. For different use case scenarios, different such targets apply, as not each business case requires the same level of security. Hence, this chapter describes both, the targets and the proposed systems, in a use case agnostic way to form a generic framework for secure NFC tag based systems.

3.1. System description

A common, concise naming and system description is given before analyzing and evaluating the system. In an NFC tag based system, such as depicted in Figure 3.1, an NFC enabled smart phone (phone) reads an NDEF message from an NFC Forum Tag (tag) via the NFC interface. Optionally, the phone also provides some input to the tag by writing an NDEF message, and then reading a second NDEF message from the tag. Subsequently, some software on the phone processes this NDEF message, and generates a request, which is then sent to a backend information system (server) at a specific endpoint (URI). This server handles this request, and provides some kind of service to the phone, such as serving a web page, triggering an action in the backend, or granting access to a web service. This happens by one or more request/response pairs exchanged between phone and server.

Both, the processing inside tag and server, are considered trusted, which means they are programmed and operated by the system operator. Tags are assumed to be produced sufficiently tamper-resistant. The phone is in possession of a user of the system. The software responsible for receiving the NDEF message, processing it, and sending the request to the server, is either provided by the system operator, or included in the phone's operating system (such as a browser).

An attacker against this system is anyone whose behavior aims at circumventing the intended purpose of the system, motivated by a number of reasons like profit, challenge or protest. This party has many possibilities to exploit the system, e.g. by manipulating the software on the phone, manipulating the tag, or by creating unauthorized requests to the server.

Following the related work on NFC security discussed in Section 2.3, a consolidated overview on threats is given in Figure 3.2. In order to define meaningful security targets, they shall not only relate to current weaknesses of NFC tag based systems, but also potential future use cases.

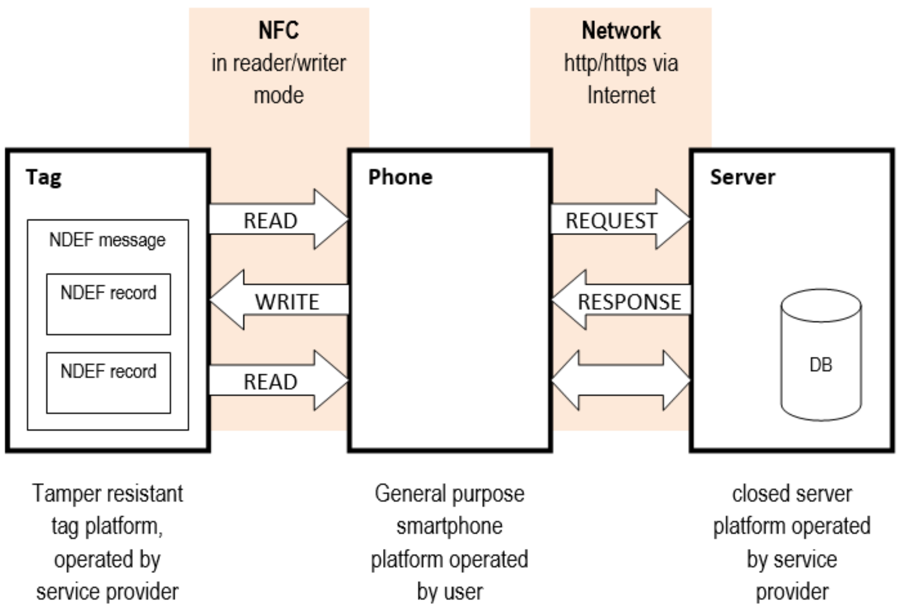

NFC
in reader/writer mode
Network
http/https via Internet
Tag
NDEF message
NDEF record
NDEF record
READ
WRITE
READ
Phone
REQUEST
RESPONSE
Server
DB
Tamper resistant tag platform, operated by service provider
General purpose smartphone platform operated by user
closed server platform operated by service provider

Figure 3.1.: Simplified view of an NFC tag based systems, and its components.

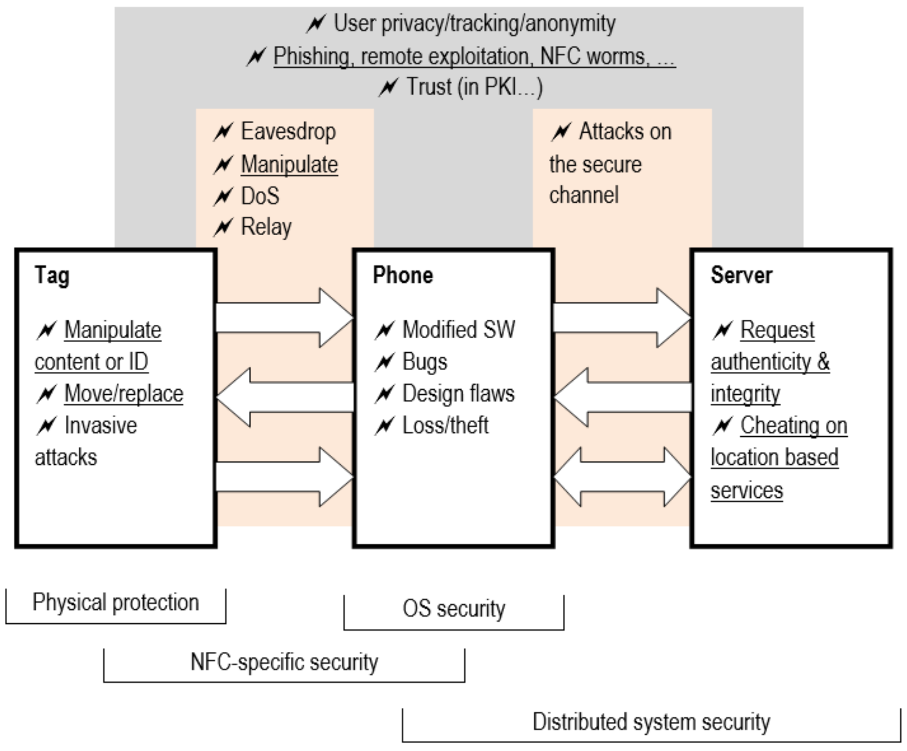

User privacy/tracking/anonymity
Phishing, remote exploitation, NFC worms, ...
Trust (in PKI...)
Eavesdrop
Manipulate
DoS
Relay
Attacks on the secure channel
Tag
Manipulate content or ID
Move/replace
Invasive attacks
Phone
Modified SW
Bugs
Design flaws
Loss/theft
Server
Request authenticity & integrity
Cheating on location based services
Physical protection
OS security
NFC-specific security
Distributed system security

Figure 3.2.: Summary of identified threats to NFC tag based systems. Underlined threats are counter-measured by methods and systems presented in this document.

Requirements of potential future applications. Research on NFC and its security also anticipates novel use cases, that may be enabled by providing a certain level of security in the system. The inherent main idea is that the server gets a cryptographically protected assurance, or proof, as part of the incoming request. This proof is the result of a tag touch by the user's phone.

In [55], Teufl et al. claim that location based services are currently meant to improve service quality for individual users. By using contextual information such as time and location, acquired by the sensors of the smart phone and sent to the server, customized responses can be provided. As this information provides advantages for users (improved service quality), no incentive to cheat on the location information is basically given. The authors anticipate a system that provides the user a proof about his location at a specific time. Through the use of cryptography and a specific protocol, a trusted ticket that contains identity, time and location information, is generated, which might also satisfy legal requirements. Such a proof might be used to document car accidents, or for guards to prove that they actually visited all check points on their nightly route.

Saroiu and Wolman propose in [49] a system to generate location proofs using Wi-Fi access points. Albeit the system is not based on NFC, a number of applications anticipated by the authors also apply to the systems considered in this thesis. Among the proposed potential applications there are store discounts for loyal customers, where to retaining customers certain rewards are offered. To prevent misuse, customers need to acquire trusted evidences of their store visits. Another prospective application is location restricted content delivery. Digital content distribution is limited to customers within a specific location. In practice, this might be applicable for internet TV content that can only be consumed in specific countries. But this might also apply to other digital content, that should only be accessible while visiting a specific location. The authors even propose to use their location proofs for police investigations, where users can also use the location information generated by the system as an alibi.

In the next section, both the weaknesses and the anticipated use cases, will be used to derive desired security targets.

3.2. Security targets

The security analysis and the potential novel use cases reveal a number of threats. For this study, the attacks are divided into two classes: Firstly, attacks targeted at users of the system, that are those operating their NFC enabled phone in order to consume a service. Secondly, the system operator itself, which provides the service via its backend system. Hence, security targets for *users*, as well as the *server* need to be identified.

The phone requires means to verify the tag, and the server requires ways to assess incoming requests that result from tag touches. In Table 3.1, a number of specific properties are given, for both phone and server:

Both main questions raise different sub-questions. Depending on the actual use case, all of them need to be answered and satisfied by a prospective system. To structure the discussion and possible solutions, we define each one by one. But first, a common understanding of the involved parties, and data slices is necessary.

The request can be compared to a ticket in the network authentication protocol Kerberos

What can the phone infer about the tag, from which it received an NDEF message?	
M1	Is the NDEF message received from the tag authentic and integer?
M2	Is the the tag, that serves the NDEF message, authentic and integer?
M3	Is the location of the tag authentic and integer, or has it been moved?

What can the server infer about an incoming request?	
R1	Is the request, and all the parameters it is composed of, integer and authentic?
R2	Is the request unique, and can an order be derived, or has it been sent multiple times (replay attack), e.g. accessing the same URI twice?
R3	Has the request been sent in a timely manner after touching the tag? E.g. an attacker might cache a request for a certain time in order to attack the system.
R4	Has the phone authenticated itself to the server, and possibly also against the tag? If not, the request to the server can be considered anonymous.

Table 3.1.: The identified security targets for phone and server.

[28]. From certain data elements provided by the tag, and protected with cryptographic means, the phone creates a request, which is then received and verified by the server in order to provide the service.

An ID uniquely identifies a tag in the context of the system. This identifier is initialized once in the lifetime of a tag, and hence static upon each read of the tag. ID_T denotes the ID of tag T. All tag IDs are stored in the server's database, including additional information such as corresponding certificates or shared, secret keys. This ID also allows to derive the context of the request, such as the location of where the user touched the tag, or which action to trigger following the request.

A counter inside the tag is incremented upon each read. The particular counter-value is then incorporated into the NDEF message served by the tag. This value is denoted C_T. It provides two important properties: Requests received by the server can be distinguished from all previous requests (uniqueness), and are ordered due to the nature of an increasing counter (order).

Using a digital signature, some or all parts of the the NDEF message are protected by the signature-value. This signature can be based on any asymmetric-key cryptography scheme, such as ECDSA. A signature-value is denoted by $\mathrm{Sig}_X(Y)$, where the data Y is signed by the private key known to X only. Furthermore, a digital certificate, denoted Cert_X, allows to verify the signature-value using the public key contained in the certificate and corresponding to the private key of party X.

A message authentication code (MAC) protects the data in the same way as a digital signature. A secret, shared key is used to both, calculate and verify, the MAC-value. Such a value is denoted $\mathrm{MAC}_X(Y)$, if generated by X in order to protect data Y. For the systems analysis, it is assumed that private and secret keys have been securely initialized on tags. In case of symmetric keys, the server stores in its database the corresponding secret keys for each tag. In case of asymmetric keys, a certificate has been issued for each tag corresponding to its private key, and both phone and server have a trusted root certificate (or a chain of certificates leading to a root).

Nonces provide freshness in terms of random and unique numbers. A nonce generated

by party X is denoted N_X. Information about the current date and time, provided by X, is denoted Time_X. Analogous, information about some location information, provided by X, is denoted Location_X. Finally, URI points to the network endpoint to which the request is being sent. It might encode any number of parameters in a form such as
`http://www.example.com/action.php?id=1234&counter=7890`.
Alternatively, one or more of the above data fields are encoded in other means inside the NDEF message.

When a tag is read, it dynamically constructs the NDEF message, and incorporates a subset of the above features (counter value, ...).

The NFC communication between tag and phone considers the exchange of NDEF messages. To read or write an NDEF message from or to a tag, a number of steps according to the NFC Forum Type Tag Operation is necessary. In this conceptual view, an abstraction of reading (R) and writing (W) an NDEF message from a tag is introduced. This allows to purely focus on the application level security, and not on specific commands of any Type Tag Operation.

In order for the phone to authenticate, it requires some kind of credentials which are acquired from the service provider. For the following discussion, it is assumed that the phone has a private, secret key to which the server knows the corresponding public key in order to verify the phone's authenticity using digital signatures. How the user and his phone gets registered with the service provider is out of scope.

The next section gives an overview of different system variants and a discussion of how to satisfy the aforementioned questions.

Out of scope fields. In order to focus on what security mechanisms (especially on the tag) can improve the security of the whole system, the scope explicitly excludes the following related, but too comprehensive fields.

Smart phone security, and especially operating system security, depends on many factors. The number of different platforms currently include Android, iOS, Windows Phone and Blackberry. Each of them includes different policies and mechanisms to hinder attackers from installing malicious software, different application distribution platforms, and different means to update the operating system once known security issues are fixed. Also, often smart phone user "root" their device in order to intentionally execute applications with elevated rights. This enables attackers to run their own malicious apps, but also allows malware to enter the phone in even more ways. As a countermeasure, research on trusted execution environments (TEE) aims at detecting the execution of malicious code inside a smart phone. Bottom line, for now no smart phone platform can be considered absolutely secure. As this thesis focuses on NFC and tags, the smart phone security research area is out of consideration. All presented means are expected to be carried out in a sufficiently secure manner by the phone.

A *secure element* is defined by GlobalPlatform[1] as "a tamper-resistant platform [...] capable of securely hosting applications and their confidential and cryptographic data (e.g. keys) in accordance with the rules and security requirements set forth by a set of well-identified trusted authorities". Despite the security benefits provided, the tight control of the owner requires a third party to negotiate access to this SE. This often is a long

[1]GlobalPlatform, `http://www.globalplatform.org/mediaguideSE.asp`, accessed on 2013-06-20

and expensive process. Furthermore, many different owners control the secure elements of different devices. Hence, for many applications, this is not a suitable approach. Also Saminger et al. in [47] propose a solution that does not require a secure element, based on this argumentation. Thus, presented system solutions do not utilize the SE on purpose.

Relay attacks, as previously outlined, cannot be prevented on the application layer. Thus this thesis neglects the threat that results from such attacks. The same applies to physical attacks, where the NFC tag is mechanically being affected (destroyed, ...).

Furthermore, for any kind of secret or private keys stored inside the tag, it is assumed that this data is protected against extraction or *tampering*. It is not the focus of this study to investigate how to prevent unauthorized access to this material, as current smart card solutions already provide such means.

Finally, the *confidentiality* of the NFC based communication channel between tag and phone is disregarded. Although NFC communication can be eavesdropped, an attacker must at least be within the range of meters. This would allow him to touch the tag himself in order to read its content anyway, so he cannot gain advantage from listening to this channel and protection means would not increase security.

The following sections cover meaningful combinations of the presented features. The different systems can be classified based on which parties contribute to the final NDEF message generated by the tag: only the tag itself, tag and phone, or tag and server. In order to provide such input to the tag, a sequence of reading, writing and reading the tag might be necessary by the phone. To name the different systems, the following scheme is used: R denotes the reading, W the writing of a NDEF message by the phone. S denotes a request being sent to the server, and its associated reply. Thus, a system named RWRS is based on the following protocol steps: read an NDEF message, write an NDEF message, read an NDEF message, and finally communicate with the server.

3.3. Systems that require only a single read of the NDEF message

In the following system proposals, only a single read operation (R) from the tag is executed in order to retrieve the NDEF message, followed by a request sent to the server (S). No input to the tag is provided by means of write operations. The systems are henceforth called RSx.

3.3.1. Basic approach (RS1)

Current state of the art tags provide an ID and a memory to store an NDEF message. In such a basic system as depicted in Figure 3.3, the NDEF message essentially provides some context information to the server in terms of a tag ID (ID_T). The NDEF message received from the tag is static, hence not changing between successive reads. The phone can neither verify any authenticity or integrity regarding the tag. Furthermore, the resulting request does not provide the server any means to assert it, as an attacker can arbitrarily modify any aspects of the request. E.g., using a modified software for the phone, requests to the server can be sent without actually physically being in proximity of a tag (besides the first

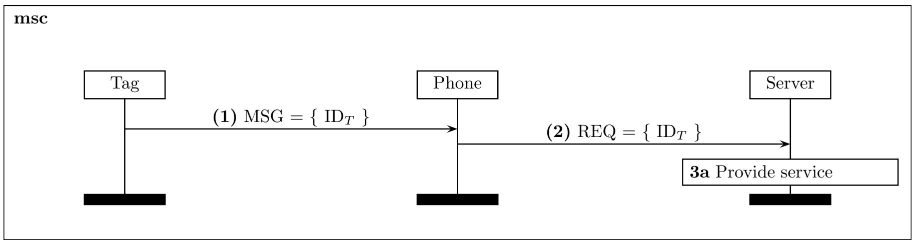


Figure 3.3.: System RS1.

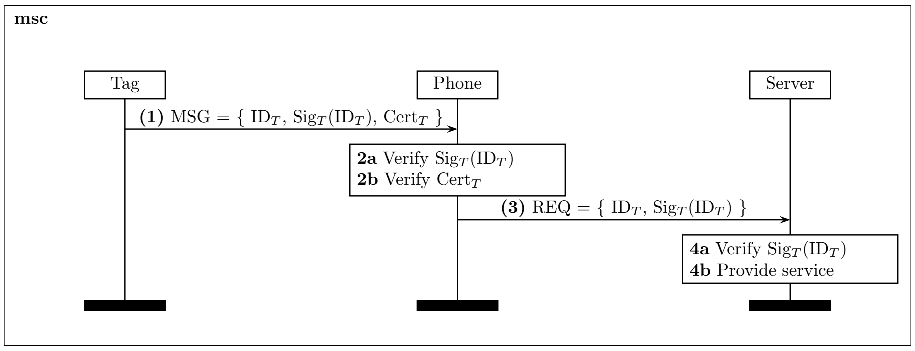


Figure 3.4.: System RS2.

time). This attack is called a replay attack. Thus, none of the security targets in Table 3.1 is satisfied.

3.3.2. Digitally signing the tag content (RS2)

Digitally signing the content of the tag allows the smart phone to verify the received NDEF message. The signing process is done by the tag author (i.e. system operator or provider) upon initialization and hence does not require the tag to be capable of cryptographic operations. A system based on this approach is shown in Figure 3.4.

Upon receiving the data and the signature ($\mathrm{Sig}_T(\mathrm{ID}_T)$), the phone first verifies the signature protecting the content (which is essentially the ID). In the next step, the digital certificate Cert_T provided by the tag must be verified (step 2b). Therefore, some kind of public key infrastructure is necessary in order to also verify the certificate, which requires a trusted root anchor.

By that, the phone can trust the authenticity and integrity of the NDEF message received (M1, see Table 3.1), but no further conclusions can be drawn. The server checks the signature of the incoming request REQ as well, in order to check that the request was unmodified (R1). As the complete message received from the tag is static, an incoming request to the server is neither distinguishable from others, nor can any information about the touch time (timeliness) be inferred. Due to the static nature of the NDEF message, replay attacks are possible using modified phone software.

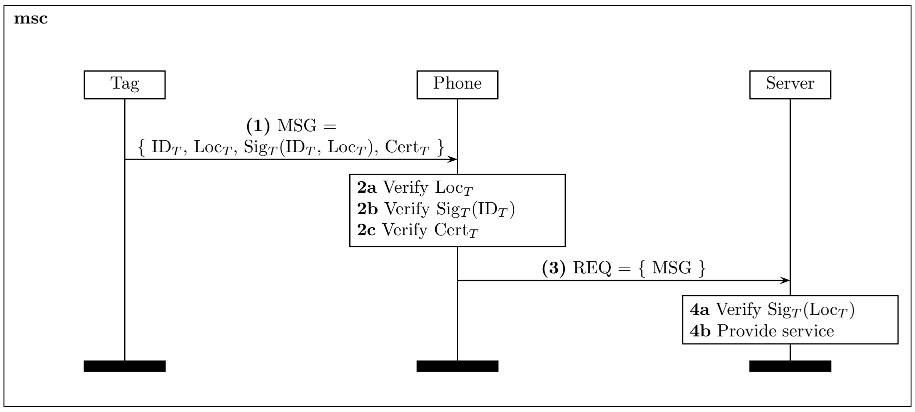


Figure 3.5.: System RS3.

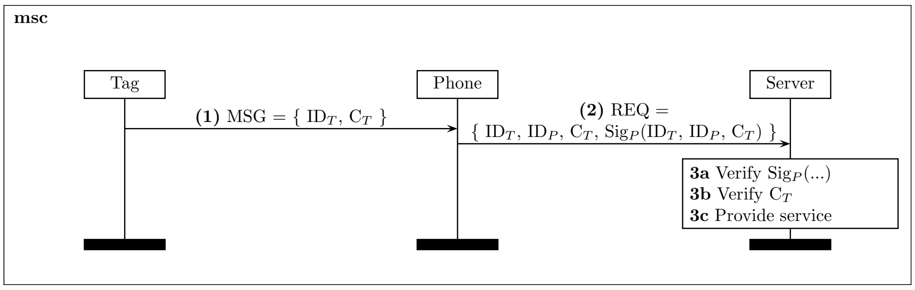


Figure 3.6.: System RS4.

3.3.3. Incorporating location information (RS3)

The system in Figure 3.5 allows the phone to verify the tags location. Location information can be represented by GPS coordinates. Incorporating this inside the signed portion of the NDEF message, additional mechanisms for the phone to assert the tag are possible: using location information acquired by phone sensors, the phone compares whether the signed location information matches its own, to determine whether the content's location is authentic (M3 - tag has not been moved). The server's options to assert the incoming request REQ are limited to what is possible with system RS2 (security target R1).

3.3.4. Providing dynamic input with a counter (RS4)

In the system shown in Figure 3.6, the tag features a counter, which is incremented upon each read of the tag. This counter value is incorporated into the NDEF message transmitted to the phone. Considering a simple tag with no signing capabilities, it is not possible to sign the counter value. Hence, the phone cannot verify by any means either tag nor NDEF message authenticity. The server receives the counter value as part of the request. Although no cryptographic protection of the value is given by the tag, the server can apply statistical measurements whether this counter value is meaningful given the context of previous requests initiated by the same tag (R2). Furthermore, as the phone provides authentication against the server (R4) by signing the counter value, an attacker might be

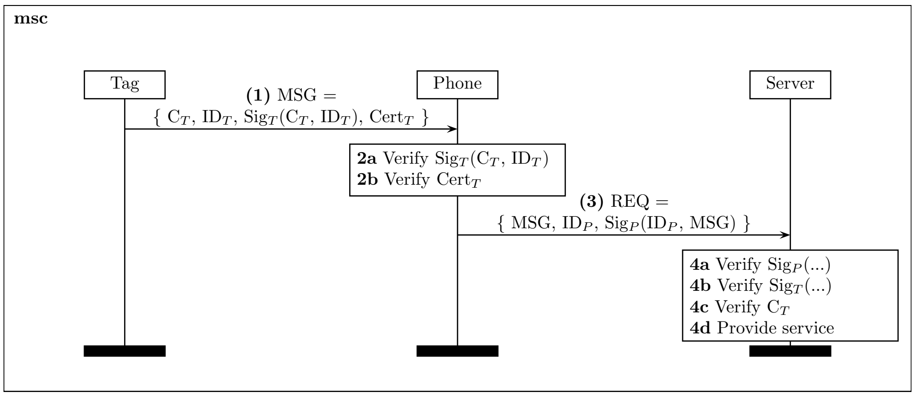


Figure 3.7.: System RS5.

discouraged to guess valid counter values. The server might detect the guessing, and block the associated user account. Thus, these two security targets are satisfied partially.

3.3.5. Protecting the counter value with a digital signature (RS5)

In the system illustrated in Figure 3.7, a tag that is capable of calculating a digital signature using a pre-initialized, secretly stored, private key. This allows to sign the counter value. Due to the every-increasing nature of this value, the phone can rely that not only the NDEF message is authentic and integer (M1), but also the tag itself (M2). A non-genuine tag would not be capable of calculating the same signature again for a different counter value.

The server receiving the signed counter value can rely that this value must have been the result of a tag touch, because it could not have been signed otherwise (R1). Furthermore, by tracking the values, multiple requests with the same number can be reliably rejected (R2). Also request timeliness can be ensured to a limited extent (R3): If an attacker caches a specific counter value and its signature, this request must be sent by the attacker before a higher counter value is received at the server. Otherwise the request with the highest counter value automatically invalidates all subsequently incoming, lower values. Finally, as the phone signs the message, it authenticates itself to the server (R4).

3.3.6. Protecting the counter with a MAC (RS6)

In the system depicted in Figure 3.8, the digital signature from the previous system is replaced by a MAC. As a MAC is based on symmetric cryptography, the tag and server must share the same secret key, which is required to create and verify the MAC value. Hence, the key cannot be shared with the smart phone, as an attacker might extract it from there. No reasoning about the NDEF message or tag is possible for the phone. But for the server-side, the same security applies as with using a digital signature (R1, R2, and to some extent R3 and R4).

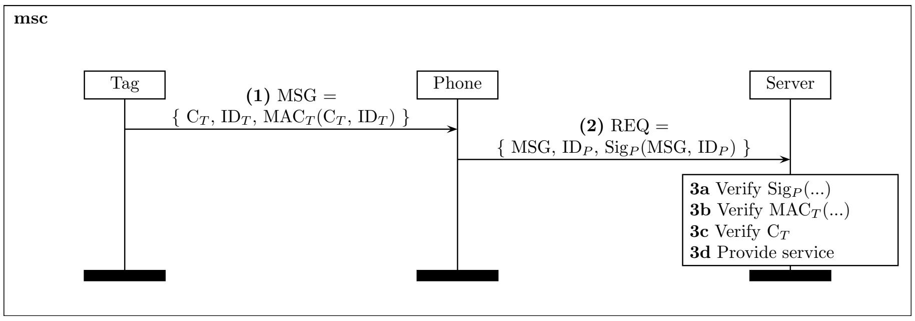


Figure 3.8.: System RS6.

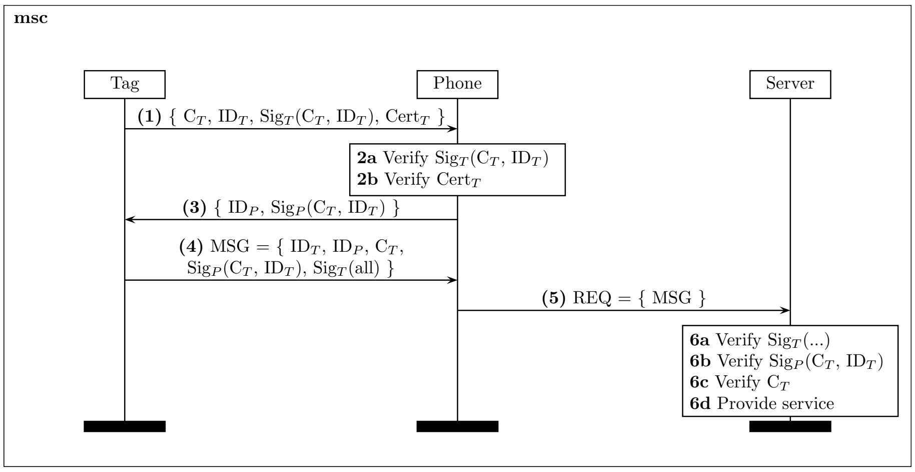

msc
Tag
Phone
Server
(1) { C_T, ID_T, $Sig_T(C_T, ID_T)$, $Cert_T$ }
2a Verify $Sig_T(C_T, ID_T)$
2b Verify $Cert_T$
(3) { ID_P, $Sig_P(C_T, ID_T)$ }
(4) MSG = { ID_T, ID_P, C_T, $Sig_P(C_T, ID_T)$, $Sig_T(all)$ }
(5) REQ = { MSG }
6a Verify $Sig_T(...)$
6b Verify $Sig_P(C_T, ID_T)$
6c Verify C_T
6d Provide service

Figure 3.9.: System RWRS1.

3.4. Systems where the phone provides protocol input

The following systems generate the NDEF message by exchanging information in three steps: a read operation (R), a write operation (W), and a final read operation (R) that finalizes the NFC communication by transferring the actual NDEF message to the phone, from which the request is then being formed and sent to the server (S). This more complex process also increases the time required for the NFC communication to complete, hence a longer touch of the tag is required. The systems are denoted RWRSx.

3.4.1. Authenticating the phone against the tag (RWRS1)

In the system depicted in Figure 3.9, the phone authenticates to the tag. The verification of the authentication provided to the tag is delegated to the server, once it receives the request containing the authentication information. The tag itself is not capable of verifying the authentication provided by the phone.

After completing the first NDEF read procedure and verifying the authenticity of NDEF message and tag (M1 and M2), the phone authenticates to the tag. Therefore, in step (3) it replies the signed counter value and tag ID, plus information about its identity. From that, the tag constructs the second, and final NDEF message, which the phone

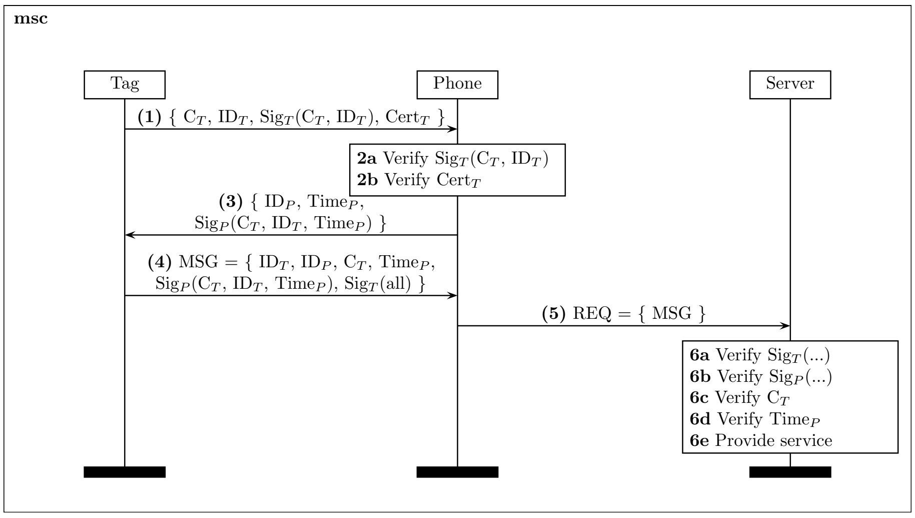

msc
Tag
Phone
Server
(1) { C_T, ID_T, $Sig_T(C_T, ID_T)$, $Cert_T$ }
2a Verify $Sig_T(C_T, ID_T)$
2b Verify $Cert_T$
(3) { ID_P, $Time_P$, $Sig_P(C_T, ID_T, Time_P)$ }
(4) MSG = { ID_T, ID_P, C_T, $Time_P$, $Sig_P(C_T, ID_T, Time_P)$, Sig_T(all) }
(5) REQ = { MSG }
6a Verify Sig_T(...)
6b Verify Sig_P(...)
6c Verify C_T
6d Verify $Time_P$
6e Provide service

Figure 3.10.: System RWRS2.

uses to send the request to the server. The ever increasing counter provides freshness to the protocol to counteract replay attacks. The signature by the tag protects the request parameters (R1), and the included counter satisfies the request uniqueness and order (R2). Request timeliness can be ensured to a certain extent, as each incoming request must have an increased counter value compared to previous requests (R3). This makes attacks by postponing requests harder for an attacker, but still not impossible.

3.4.2. Providing phone-authenticated time to the tag (RWRS2)

This system depicted in Figure 3.10 differs from the previous one in a single aspect: In step (3), the phone also provides the current time to be incorporated into the mutually signed, final NDEF message. This makes attacks regarding security property R3 (timeliness) even harder for an attacker, as he also needs to know the time when he wants to send the postponed request.

3.5. Systems where the server provides a nonce as protocol input (RSWRS1)

The system seen in Figure 3.11 is similar to the one presented by Teufl et al. in [55]. It requires the phone, after retrieving the first NDEF message from the tag, to request a nonce N_S from the server immediately (steps 3 and 4). Then the phone, and finally the tag sign this nonce using their respective private keys. That way, in step 6 the final NDEF message is being generated. Using the corresponding public keys, which the server stores in its database, it verifies the mutually signed nonce. By limiting the time frame between steps 3 and 7, the server also ensures that the request happens in a timely manner in respect to the actual tag touch (R3). However, this system requires the longest touch time of the tag, as it needs to retrieve the nonce from the server before doing the write, and the final NDEF message read operation.

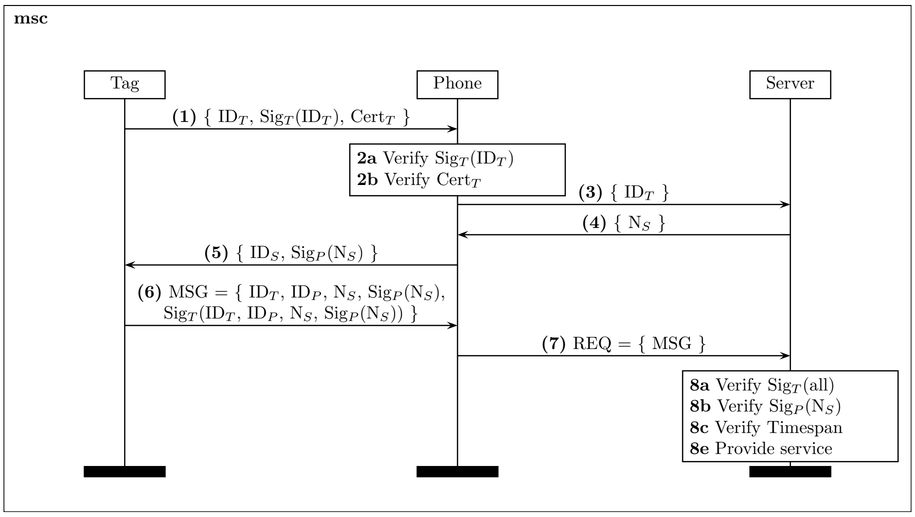


Figure 3.11.: System RSWRS1.

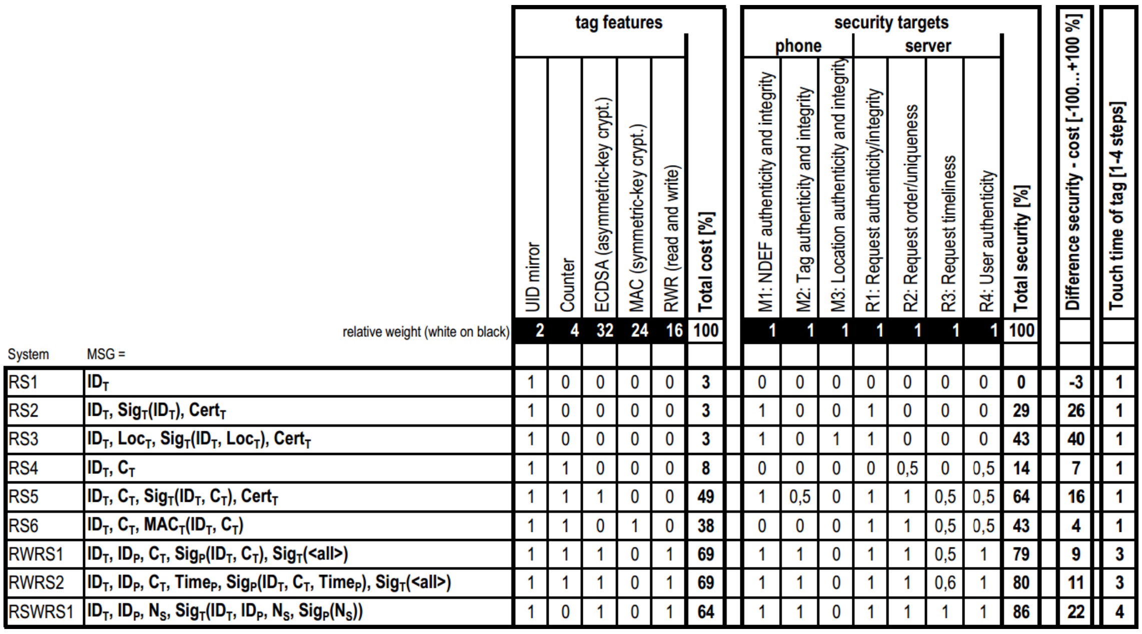
3.6. Evaluation of the systems

An overview of the presented systems in given in Figure 3.12. On the left, for each system the essential content of the final NDEF message is given. On the top, the necessary features of each tag are depicted. A UID mirror is required to incorporate the tag ID into the NDEF message. The counter mirror internally increments and incorporates the updated value on each read of the tag (for systems where the tag is read twice, this increment happens only before reading the first NDEF message in the first step). ECDSA and MAC features compute the corresponding cryptographic values. The feature RWR requires the tag to support the read, write and read sequence, where the phone provides input to the tag.

Both, the tag features as well as the security properties, are weighted. E.g. integrating ECDSA into a tag is assumed to be eight times as expensive as building in a counter. Also the desired security targets can be weighted. In the given figure, all features are weighted equally in order to give an use-case independent evaluation. Depending on a specific use case, some properties might be more important, and others might not even be required.

A general recommendation for a system cannot be given. Depending on the specific requirements, the relative weights in terms of cost and security need to be determined, and then compared. The column "Difference security - cost" indicates a possible tradeoff between cost and security, but also the complexity of the tag touch needs to be considered, as it increases touch time, and the time required to communicate with the server.

In the next chapter, the requirements for a specific system will be analyzed, and in regard to that, an appropriate solution will be selected.



		tag features						security targets									
								phone			server						
		UID mirror	Counter	ECDSA (asymmetric-key crypt.)	MAC (symmetric-key crypt.)	RWR (read and write)	**Total cost [%]**	M1: NDEF authenticity and integrity	M2: Tag authenticity and integrity	M3: Location authenticity and integrity	R1: Request authenticity/integrity	R2: Request order/uniqueness	R3: Request timeliness	R4: User authenticity	**Total security [%]**	**Difference security - cost [-100...+100 %]**	**Touch time of tag [1-4 steps]**
	relative weight (white on black)	2	4	32	24	16	100	1	1	1	1	1	1	1	100		
System	MSG =																
RS1	**ID_T**	1	0	0	0	0	**3**	0	0	0	0	0	0	0	**0**	**-3**	**1**
RS2	**ID_T, $Sig_T(ID_T)$, $Cert_T$**	1	0	0	0	0	**3**	1	0	0	1	0	0	0	**29**	**26**	**1**
RS3	**ID_T, Loc_T, $Sig_T(ID_T, Loc_T)$, $Cert_T$**	1	0	0	0	0	**3**	1	0	1	1	0	0	0	**43**	**40**	**1**
RS4	**ID_T, C_T**	1	1	0	0	0	**8**	0	0	0	0	0,5	0	0,5	**14**	**7**	**1**
RS5	**ID_T, C_T, $Sig_T(ID_T, C_T)$, $Cert_T$**	1	1	1	0	0	**49**	1	0,5	0	1	1	0,5	0,5	**64**	**16**	**1**
RS6	**ID_T, C_T, $MAC_T(ID_T, C_T)$**	1	1	0	1	0	**38**	0	0	0	1	1	0,5	0,5	**43**	**4**	**1**
RWRS1	**ID_T, ID_P, C_T, $Sig_P(ID_T, C_T)$, $Sig_T(<all>)$**	1	1	1	0	1	**69**	1	1	0	1	1	0,5	1	**79**	**9**	**3**
RWRS2	**ID_T, ID_P, C_T, $Time_P$, $Sig_P(ID_T, C_T, Time_P)$, $Sig_T(<all>)$**	1	1	1	0	1	**69**	1	1	0	1	1	0,6	1	**80**	**11**	**3**
RSWRS1	**ID_T, ID_P, N_S, $Sig_T(ID_T, ID_P, N_S, Sig_P(N_S))$**	1	0	1	0	1	**64**	1	1	0	1	1	1	1	**86**	**22**	**4**

Figure 3.12.: Evaluation of possible systems regarding the required tag features and cost, the complexity of the protocol, and the provided security targets.

4. Analysis and design of the prototype system

This chapter describes *what* the prospective system does. Thus, the requirements for a coupon demo system are elaborated. Using the theoretical foundation from Chapter 3, the proposed system solutions are evaluated and a suitable one selected. Finally, the object oriented design of the prototype, and its interfaces and data structures, are given.

4.1. Requirements analysis for a mobile, electronic coupon demo system

Mobile, electronic coupons are a specific form of loyalty marketing. This section analyses the requirements for the implementation of a system prototype.

4.1.1. Scope

The aim is to demonstrate a secure, NFC tag based system that enables mobile and electronic coupons. Henceforth, the system involves two parties, the end user who collects and redeems coupons, and the system operator. As the prospective system is a prototype, the system operator party unifies various independent organizations in practice, such as the advertiser, the retailer, and the technical solution provider.

Research goal: security. One goal of this system is to find ways how to design and implement secure, tag based system. Different options shall be considered and evaluated, and the most suitable option shall be implemented.

Development goal: demo. The purpose of the implemented system is to serve as a demo for customers of NXP to demonstrate both, the improved security and the general utilization of NFC tags for loyalty applications. Hence, the demo system must be easily deploy- and demonstrable, ie. require as few initialization or set-up steps, as well as as few as possible hardware equipment (servers, etc.).

Functional goal: loyalty. The function of the implemented system shall be related to loyalty, in this particular case to mobile electronic coupons. Classic paper coupons are a wide-spread means of marketing, and this system shall demonstrate the usefulness of electronic coupons stored on the users smart phone.

4.1.2. Use cases

The application supports the three basic use cases, depicted in Figure 4.1:

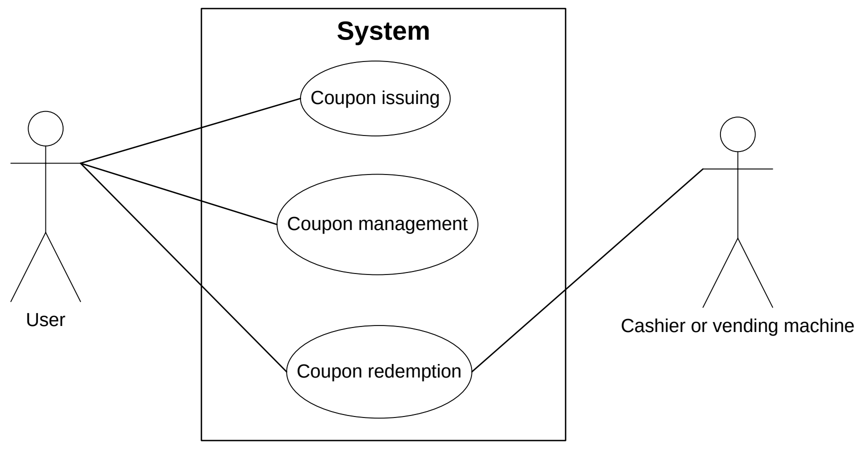


Figure 4.1.: Use case overview.

1. *Coupon issuing* is the process of collecting coupons by touching a smart poster with an NFC enabled smart phone. The process involves the user's phone, and the NFC tag attached to the poster advertisement that offers the coupon. After a coupon is received from a poster, it is stored in the memory of the phone. This approach of issuing coupons is also known as *pull coupons*, as the user actively decides to gather a coupon [5].

2. *Coupon management* allows the user to view and delete his coupons, which are stored in his smart phone. For each coupon, details about its value and collection time need to be shown. Also, previously redeemed coupons can be reviewed.

3. *Coupon redemption* allows the user to cash in a coupon that was previously collected and stored in the phone. The process is triggered by the touch of a redemption tag, located at the point of sales terminal. Depending on the use case or scenario, this redemption might take place before or after serving/delivering the counter value of the coupon.
   - Store (variant 1): The user gathers the desired goods inside the store and then proceeds to checkout, where the cashier calculates the total amount. At the POS terminal, the user touches the redemption tag using his smart phone to redeem one or more suitable coupons. The cashed-in coupons are deducted from the total amount, and the user only needs to pay the remaining (if any) amount.
   - Self-service restaurant (variant 2): The user touches the redemption tag. The waiter gets informed about the cashed-in coupon, and serves the corresponding goods (e.g. a coffee, burger).
   - Vending machine (Variant 3): The user touches the redemption tag. The vending machine gets informed (via a backend system), and dispenses the corresponding goods.

The system is composed of the following components. The user owns his personal, private smart *phone* featuring an NFC interface, and with a coupon management software installed. The system operator issues coupons using NFC tags attached to public ads. Such a tag that issues coupons is denoted the *issuing tag.* At each specific cash-in location, a *redemption tag* is attached. A backend information system, called *server* connects the users phone with the terminal where the coupon is being redeemed.

4.1.3. Coupons and security

A mobile, electronic coupon is represented by a piece of data, and protected by cryptographic means. The coupon is in possession of the owner of the phone on which this data is stored. According to [1], a coupon system shall enforce the following security properties regarding the coupon:

1. A coupon shall only be generated upon physically touching an issuing tag. Neither unauthorized party shall be able to generate or issue coupons. Nor shall anyone who is not in proximity of the issuing tag be able to collect coupons.
2. A validly generated coupon must not be modifiable, such as changing its face value, or any other data fields encoded in the data representing the coupon.
3. The multiple cash-in of a single coupon shall be detected by the system, and redemption must then be declined.

Regarding the overall system, further security mechanisms shall be provided:

1. The authenticity of the issuing tags content, as well as of the tag itself, shall be ensured by means of digital signatures, a suitable PKI infrastructure and a compact certificate scheme, in order to protect the user from malicious issuing tags.
2. User privacy shall be preserved, hence each user shall be anonymous in the context of the system, and not be required to register with the system.
3. On redemption, the coupon must be transferred from the user's phone to the backend information system via an internet connection. The phone shall only upon physically touching a redemption tag be able to access the server, in order to minimize potential attack vectors on the backend system of the service operator.

4.1.4. Further requirements

The coupon issuing use case shall be carried out offline. The phone receives the coupon from the issuing tag without having to access (e.g. via internet) any other component of the system (such as downloading a coupon from a webpage). This allows coupons to be also issued at remote locations, where no network connection might be available.

The cash-in of a coupon is triggered by the touch of the redemption tag. This tag shall not require any connection to the POS terminal or vending machine directly. After tag touch, the phone redeems the coupon by transmitting it via its network interface to the server of the system operator. This backend system then informs the terminal or vending machine about the successful cash-in. That way, the system demonstrates how to upgrade existing terminals without connecting an NFC interface, but by only adding a passive and rather cheap NFC tag.

Current smart phones are based on different OS platforms. As the access to the NFC interface varies in terms of protocol layers, all use cases shall only require the reading, and potentially writing, of NDEF message from/to the tag. No low level NFC commands, or specific operations must be carried out in order to collect or redeem a coupon. Specifically, NFC communication with the tag shall be carried out in reader/writer mode. Cryptographic protection mechanisms may utilize the NFC Forum Signature RTD, if suitable.

As the prototype shall evaluate the usefulness of future NFC tag products, the functionality of the issuing and redemption tags shall be implemented using NXP JCOP contactless smart cards. This platform allows to implement the necessary features, such as cryptographic operations, in software.

The NFC paradigm of intuitive touching shall be enforced. This implies that upon touch of a redemption tag, a suitable coupon may be automatically selected, depending on the context of the redemption tag (e.g. which specific store). For issuing, no user interaction besides the tag touch shall be necessary to acquire a coupon.

As the phone is a highly personal device, and in possession of the user, he shall have full control over all collected coupons. In order to redeem a specific coupon, the user shall be able to choose one before touching the redemption tag.

The purpose of the prototype system implementation is to provide (1) a demo system to be shown to customers, and (2) source code components for the generation and validation of coupons. Henceforth, the system shall be in a pre-initialized state, with pre-configured tags. Furthermore, reusable software components, such as the ECQV related logic, shall be encapsulated in a separate library.

The backend information system is considered trusted, as it is operated by the service provider. The NFC tags are considered to be tamper resistant against various kinds of attacks which disclose secret information (e.g. cryptographic material) or allow to change their internal state. It is out of scope to develop counter measures on hardware level.

4.1.5. System evaluation and selection

In this section, the identified requirements will be matched against potential systems discussed in Chapter 3.

Considering an electronic coupon, one can think of it as a the previously described request to a server. The NDEF message received from the tag denotes the coupon, encoding several specific, additional parameters such as the offer. On redemption, this coupon is sent as a request to the server, in order to get it verified and cashed-in. Henceforth, the following security properties (from Table 3.1) apply: M1 and M2 provide tag and coupon authenticity and integrity to the phone when it receives the coupon. As the coupon is then stored on the phone until redemption, property R3 (timeliness) is explicitly not required. Yet, in order to protect the coupon, R1 provides authenticity and integrity of the coupon to the server. Furthermore, the uniqueness property (R2) is necessary in order to make each coupon distinguishable from all others, and to prevent double spending. As users are anonymous, no registration and hence user authentication is necessary (R4).

An evaluation of the coupon system is depicted in Figure 4.2, where the specifically required security targets have been weighted with 1, in order to get an overall score for the total security in respect to the specific requirements of coupons. Systems RS1 to RS4 require a counter as the most complex tag feature. As they do not provide cryptographic

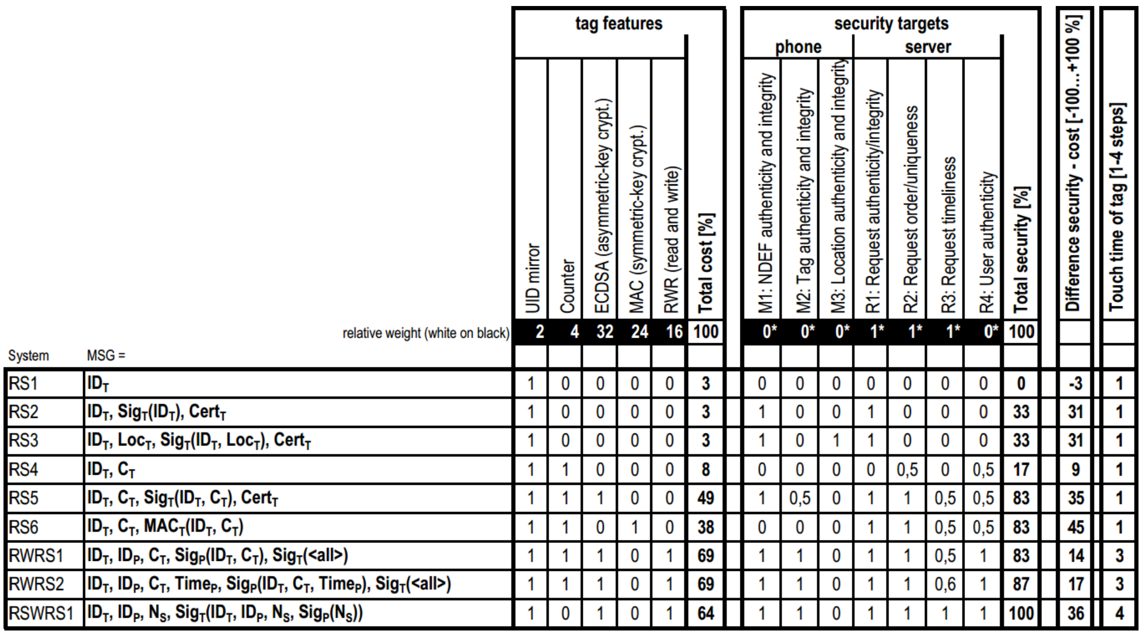
		tag features						security targets									
								phone			server						
System	MSG =	UID mirror	Counter	ECDSA (asymmetric-key crypt.)	MAC (symmetric-key crypt.)	RWR (read and write)	**Total cost [%]**	M1: NDEF authenticity and integrity	M2: Tag authenticity and integrity	M3: Location authenticity and integrity	R1: Request authenticity/integrity	R2: Request order/uniqueness	R3: Request timeliness	R4: User authenticity	**Total security [%]**	**Difference security - cost [-100...+100 %]**	**Touch time of tag [1-4 steps]**
	relative weight (white on black)	**2**	**4**	**32**	**24**	**16**	**100**	**1***	**1***	**0***	**1***	**1***	**0***	**0***	**100**		
RS1	**ID_T**	1	0	0	0	0	**3**	0	0	0	0	0	0	0	**0**	**-3**	**1**
RS2	**ID_T, $Sig_T(ID_T)$, $Cert_T$**	1	0	0	0	0	**3**	1	0	0	1	0	0	0	**50**	**47**	**1**
RS3	**ID_T, Loc_T, $Sig_T(ID_T, Loc_T)$, $Cert_T$**	1	0	0	0	0	**3**	1	0	1	1	0	0	0	**50**	**47**	**1**
RS4	**ID_T, C_T**	1	1	0	0	0	**8**	0	0	0	0	0,5	0	0,5	**13**	**5**	**1**
RS5	**ID_T, C_T, $Sig_T(ID_T, C_T)$, $Cert_T$**	1	1	1	0	0	**49**	1	0,5	0	1	1	0,5	0,5	**88**	**39**	**1**
RS6	**ID_T, C_T, $MAC_T(ID_T, C_T)$**	1	1	0	1	0	**38**	0	0	0	1	1	0,5	0,5	**50**	**12**	**1**
RWRS1	**ID_T, ID_P, C_T, $Sig_P(ID_T, C_T)$, $Sig_T(<all>)$**	1	1	1	0	1	**69**	1	1	0	1	1	0,5	1	**100**	**31**	**3**
RWRS2	**ID_T, ID_P, C_T, $Time_P$, $Sig_P(ID_T, C_T, Time_P)$, $Sig_T(<all>)$**	1	1	1	0	1	**69**	1	1	0	1	1	0,6	1	**100**	**31**	**3**
RSWRS1	**ID_T, ID_P, N_S, $Sig_T(ID_T, ID_P, N_S, Sig_P(N_S))$**	1	0	1	0	1	**64**	1	1	0	1	1	1	1	**100**	**36**	**4**

Figure 4.2.: Evaluation of systems for electronic coupons using the framework from Chapter 3. The security target weights have been adjusted to reflect the coupon specific requirements (marked with a *). System RS5 offers the most reasonable tradeoff in regard to system security, tag cost and tag touch time.

operations, it is infeasible to generate coupons satisfying the security requirements. The systems RWRS1, RWRS2, and RSWRS1 provide optimal fulfilment of the desired security properties (and even more), but require a more complex tag communication to be carried out. Henceforth, the system RS5, which requires a counter and public key cryptography features on the tag, provides sufficient security for coupons and requires no advanced communication with the tag, but only a single NDEF read operation.

For the redemption process, it must be ensured that only phones that actually touch a redemption tag are able to communicate with the server infrastructure of the system operator. This procedure is denoted *secure trigger*. Henceforth, the tag must provide some kind of "ticket" to the phone, which proves proximity to the tag. For this redemption tag, no verification means for the phone are considered (M1-M3), as the focus is on secure access to the backend server. From the server's perspective, it must be possible to verify the request authenticity and integrity (R1). Furthermore, the requests must provide some order (R2), from which also the timeliness property (R3) can be derived, as a coupon should only be redeemable at the time a user touches the redemption tag in a store or at a vending machine.

An evaluation of systems for the secure trigger is depicted in Figure 4.2. The required security targets have been weighted with 1. In comparison to the coupon evaluation, no phone related security targets are considered, as the focus is only on server-side validation. In this case, system RS6 offers a suitable trade off between cost and security, as cheaper symmetric cryptography is sufficient compared to the issuing tag (in terms of tag hardware).



		tag features						security targets									
								phone			server						
		UID mirror	Counter	ECDSA (asymmetric-key crypt.)	MAC (symmetric-key crypt.)	RWR (read and write)	**Total cost [%]**	M1: NDEF authenticity and integrity	M2: Tag authenticity and integrity	M3: Location authenticity and integrity	R1: Request authenticity/integrity	R2: Request order/uniqueness	R3: Request timeliness	R4: User authenticity	**Total security [%]**	**Difference security - cost [-100...+100 %]**	**Touch time of tag [1-4 steps]**
	relative weight (white on black)	**2**	**4**	**32**	**24**	**16**	**100**	**0***	**0***	**0***	**1***	**1***	**1***	**0***	**100**		
System	MSG =																
RS1	ID_T	1	0	0	0	0	**3**	0	0	0	0	0	0	0	**0**	**-3**	**1**
RS2	$ID_T, Sig_T(ID_T), Cert_T$	1	0	0	0	0	**3**	1	0	0	1	0	0	0	**33**	**31**	**1**
RS3	$ID_T, Loc_T, Sig_T(ID_T, Loc_T), Cert_T$	1	0	0	0	0	**3**	1	0	1	1	0	0	0	**33**	**31**	**1**
RS4	ID_T, C_T	1	1	0	0	0	**8**	0	0	0	0	0,5	0	0,5	**17**	**9**	**1**
RS5	$ID_T, C_T, Sig_T(ID_T, C_T), Cert_T$	1	1	1	0	0	**49**	1	0,5	0	1	1	0,5	0,5	**83**	**35**	**1**
RS6	$ID_T, C_T, MAC_T(ID_T, C_T)$	1	1	0	1	0	**38**	0	0	0	1	1	0,5	0,5	**83**	**45**	**1**
RWRS1	$ID_T, ID_P, C_T, Sig_P(ID_T, C_T), Sig_T(<all>)$	1	1	1	0	1	**69**	1	1	0	1	1	0,5	1	**83**	**14**	**3**
RWRS2	$ID_T, ID_P, C_T, Time_P, Sig_P(ID_T, C_T, Time_P), Sig_T(<all>)$	1	1	1	0	1	**69**	1	1	0	1	1	0,6	1	**87**	**17**	**3**
RSWRS1	$ID_T, ID_P, N_S, Sig_T(ID_T, ID_P, N_S, Sig_P(N_S))$	1	0	1	0	1	**64**	1	1	0	1	1	1	1	**100**	**36**	**4**

Figure 4.3.: Evaluation of systems for securely accessing a server (secure trigger). Again, the framework from Chapter 3 has been used and the security target weights have been adjusted to reflect the coupon specific requirements (marked with a *). System RS6 offers the most reasonable tradeoff in regard to system security, tag cost and tag touch time.

4.2. Object oriented design and interface specification

This section describes *how* the prospective system works by defining its structure, interfaces and dynamic aspects, using an object oriented design design approach and UML (unified modelling language).

A *coupon* is a set of data fields, encoded inside an NDEF message. Its integral parts are the offer, which specifies the actual value of the coupon, and the counter, which makes each single coupon issued distinguishable from all others. In order to acquire a coupon, an *issuing tag* dynamically generates this data set, and transmits it to the phone. The *phone* is operated by the *user*, and has a software installed to receive, handle and redeem the collected coupons. Upon touching a *redemption tag*, a *secure trigger* NDEF message is received by the phone. This data set is a proof to the *server* that the user actually touched the redemption tag at this specific moment in time. Together with a previously acquired coupon, the secure trigger is then sent to the server for redemption. If both, the secure trigger and the coupon, are validated successfully at the server, it clears the coupon by informing the phone and its user. Furthermore, a terminal at the redemption location gets informed, in order to indicate to the cashier the successful cash-in. The terminal itself is represented in the demo context by a web fronted in the browser.

4.2.1. System components

The UML deployment diagram in Figure 4.4 depicts the software components, the physical layer and the physical connections in between. On coupon issuing, the phone communicates with the issuing tag via the NFC interface, in order to retrieve a coupon. For coupon cash-in, the redemption tag sends via NFC a secure trigger to the phone. Then, the server is accessed by transferring the coupon and the secure trigger. Finally, both the phone and the terminal (such as a vending machine, or cash desk computer) get informed about the successful redemption. A database provides the server with the necessary information to verify the validity of a coupon redemption procedure.

4.2.2. Protocol

The protocol in Figure 4.5 describes the whole lifetime of a single coupon. It is composed of the coupon issuing steps 1–4, and the redemption process in steps 5–12.

1. Once the user brings the phone into proximity of an issuing tag, the phone carries out the necessary NFC commands to initiate the process of reading an NDEF message.

2. The tag prepares the coupon NDEF message by updating the necessary fields, particularly the counter and the signature value.

3. The coupon NDEF message is transferred to the phone via NFC.

4. The phone receives the coupon and verifies its content using the second NDEF record, which carries the signature and the corresponding certificate (4a). To verify the certificate, the phone uses the pre-installed intermediate and root certificates. If the validation succeeds, the coupon is stored in the phone's memory, and displayed to the user (4b).

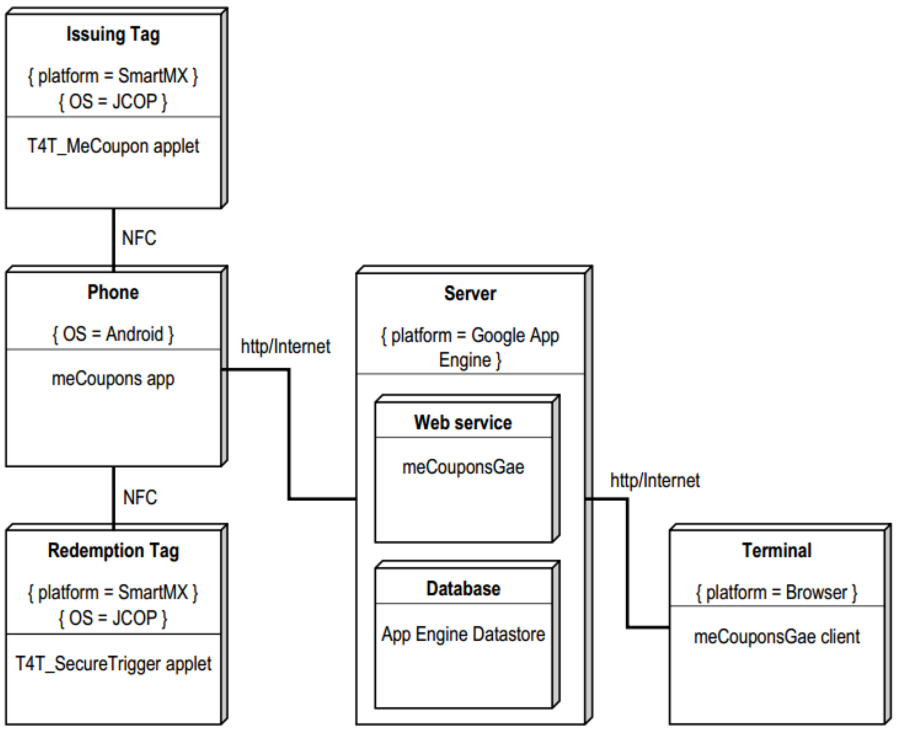


Figure 4.4.: The UML deployment diagram show the software components, the physical layer (hardware), and their physical connections.

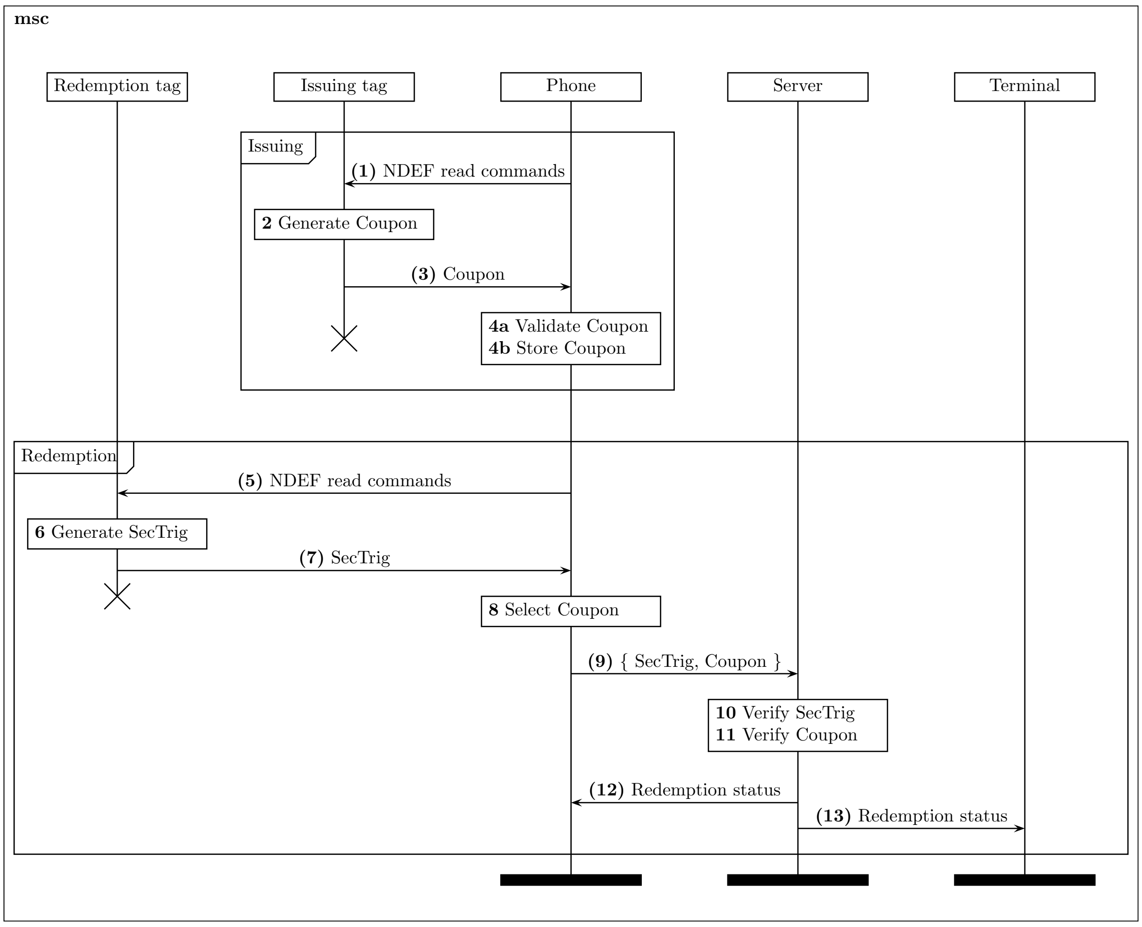


Figure 4.5.: This message sequence diagram depicts the dynamic aspect of the system. It shows the interaction of all components, from the issuing of a coupon to its redemption.

5. In order to cash-in the coupon, the user touches a redemption tag with his phone. This then carries out the necessary NFC commands to initiate the reading of an NDEF message.
6. The tag prepares the secure trigger NDEF message (SecTrig) by updating the necessary fields, particularly the counter and the MAC value.
7. The secure trigger NDEF message is transferred to the phone via NFC.
8. The phone selects a suitable coupon to be redeemed. This is done by matching the vendor ID field from the secure trigger with stored coupons. If multiple coupons match, the oldest, unredeemed one is selected. If no coupon matches, the process stops. Furthermore, if the user had pre-selected a coupon before touching the tag, this one is used instead of automatically selecting one.
9. A request to the server is sent, containing the just received secure trigger, and the selected coupon.
10. The server receives and verifies the incoming request. For the secure trigger, it must be ensured that the tag touch actually happened, and is timely:
    a) Check the tag ID and retrieve the associated secret key from the database.
    b) Verify the MAC using the secret key (to ensure an authentic and integer request).
    c) Check that the counter value is the greatest one known so far for this specific tag (to ensure timeliness).
    d) If any check fails, reject the request.
11. If the request is allowed, continue to verify, and eventually redeem the coupon.
    a) Verify the signature in the second NDEF record of the coupon to assert coupon authenticity and integrity, using the certificate from the signature record. Then validate this certificate using the intermediate and root certificate known to the server.
    b) Check the tag ID, the vendor ID, and the offer ID using the database.
    c) Verify the coupon whether it has already been cashed-in to avoid double-spending by querying the database for already redeemed coupons.
    d) If any check fails, reject the coupon redemption. On success, mark the coupon as redeemed in the database.
12. Send a response to the phone, indicating the status of the redemption.
13. Inform the terminal about the redemption status of the coupon.

4.2.3. Coupon and secure trigger NDEF message formats

The coupon, depicted in Table 4.2, is composed of two NDEF records. The first one uses an external type "nxp.com:mecoupon" and contains information about the coupon,

Type	*Type ID*	*Content of value field*
Offer	0x01	Encodes a textual description of the coupons offer as an UTF-8 encoded string.
Offer ID	0x02	Encodes the Offer ID (8 bytes).
Tag ID	0x03	Encodes the Tag ID (8 bytes).
Counter value	0x04	Encodes the value of the counter in a 4 byte field using big endian.
MAC value	0x05	Encodes the calculated MAC value. This field may only carry an SHA256 based HMAC.
Vendor ID	0x06	Encodes the vendor ID (8 bytes).

Table 4.1.: The TLVs used to encode coupon and secure trigger specific data inside the payload of the respective NDEF records.

the issuing tag, and the counter value. To protect the coupon's data, a signature record according to the Signature RTD Version 2 [37] follows, using an ECQV implicit certificate as depicted in Table 4.4.

For cash-in of a coupon, a secure trigger NDEF message acquired from a redemption tag is generated, as depicted in Table 4.3. The data fields contained are protected by a MAC, which constitutes the last field in the records payload.

The coupon and secure trigger specific fields are encoded as type-length-value (TLV) elements. Both, type and length are fields of size 1 byte. The length indicates then the size of the value field. Table 4.1 summarizes the defined TLVs necessary to encode the required information.

4.2.4. PKI and symmetric key setup

To enable the cryptographic protection mechanism, a system initialization is necessary. The following steps need to be performed in order to perform validation of coupons using a public key infrastructure (PKI). The resulting key infrastructure is depicted in Figure 4.6.

1. The system operator requires a root and an intermediate certificate, encoded according to X.509 [54]. For the demo system, a self-signed root certificate, denoted $\text{Cert}_{CA} = \{ \text{ID}_{CA}, \text{Q}_{CA} \}$, is generated. The certificate essentially binds the identity information ID of the CA to its public key Q. The corresponding private key d_{CA} is kept secret by the certificate authority.

2. Furthermore, an intermediate certificate hierarchy is created by signing a certificate dedicated to the system provider using d_{CA}, giving $\text{Cert}_{SP} = \{ \text{ID}_{SP}, \text{Q}_{SP} \}$. The corresponding private key d_{SP} is kept secret by the system provider.

3. For each issuing tag (IT), an ECC key pair is generated using the ECQV scheme. The corresponding ECQV implicit certificates are signed using d_{SP}, giving $\text{Cert}_{IT} = \{ \text{ID}_{IT}, \text{Q}_{IT} \}$. The certificates are stored on the issuing tags, and will be incorporated into each coupon NDEF message issued by the issuing tag, whereas the private key is stored securely inside the tag and used to dynamically generate the digital signature that is part of each coupon NDEF message.

Data field	Description	DC	DL
NDEF Record	Coupon	-	-
Flags	MB=1, ME=0, CF=0, SR=1, IL=0, TNF=0x04	n	n
Type length	Length of the "nxp.com:mecoupon" = 16 bytes.	n	n
Payload length	1 byte field indicating the length of the Payload.	n	n
Type	NFC Forum external type "nxp.com:mecoupon"	n	n
Payload	Coupon proprietary TLV structure	-	-
Offer	TLV field containing a textual description of the coupons offer.	n	n
Offer ID	TLV field of 8 bytes that is a unique identifier to relate the offer in the backend system.	n	n
Tag ID	TLV field of 8 bytes that uniquely identifies the tag that issued the offer.	n	n
Vendor ID	TLV field of 8 bytes that uniquely identifies the vendor who accepts this coupon.	n	n
Counter	TLV field with a 4 byte dynamic value that is incremented for each new coupon.	y	n
NDEF Record	Signature	-	-
Flags	MB=0, ME=1, CF=0, SR=1, IL=0, TNF=0x01	n	n
Type length	Length of the type "Sig" = 3 bytes.	n	n
Payload length	1 byte field indicating the length of the Payload, which is (70 + length of signature value).	y	n
Type	NFC Forum well known type "Sig"	n	n
Payload	Payload (according to [37])	-	-
Version	"2.0"	n	n
Signature	Signature and corresponding meta information.	n	n
Flags	URI present: no, Signature type: ECDSA-P192	n	n
Hash type	SHA256	n	n
Length	Length of the signature value as a 2 byte field.	y	n
Value	The actual signature value protecting the complete first NDEF record.	y	y
Certificate chain	Fields describing the certificate chain and meta information about it.	n	n
Flags	URI present: no, Format: ECQV, Number of certs: 1	n	n
Certificate 0	The first certificate of the chain.	-	-
Length	Length of the subsequent certificate.	n	n
Certificate	The ECQV implicit certificate in MES encoding to validate the signature.	n	n

Table 4.2.: The Coupon NDEF message format. The column *DC* indicates whether the content of the field changes for each coupon issued (dynamic content). The column *DL* indicates whether the content may also vary in length for different issued coupons (dynamic length).

Data field			Description	DC	DL
NDEF Record			Coupon	-	-
	Flags		MB=1, ME=1, CF=0, SR=1, IL=0, TNF=0x04	n	n
	Type length		Length of type field value.	n	n
	Payload length		Length of payload field value.	n	n
	Type		NFC Forum external type "nxp.com:sectrig"	n	n
	Payload		Secure trigger proprietary TLV structure	-	-
		Tag ID	TLV field of 8 bytes that uniquely identifies the tag that issued the offer.	n	n
		Vendor ID	TLV field of 8 bytes that uniquely identifies the vendor. Using this contextual information, a suitable coupon with matching vendor ID can be automatically selected.	n	n
		Counter	TLV field with a 4 byte dynamic value that is incremented for each new secure trigger.	y	n
		MAC	A HMAC-SHA256 based MAC that protects all previous fields (including complete NDEF record header) using a secret key shared between tag and server. The correct key is related via the tag ID at the server.	y	n

Table 4.3.: The secure trigger NDEF message format, consisting of a single NDEF record. The column *DC* indicates whether the content of the field changes for each coupon issued (dynamic content). The column *DL* indicates whether the content may also vary in length for different issued coupons (dynamic length).

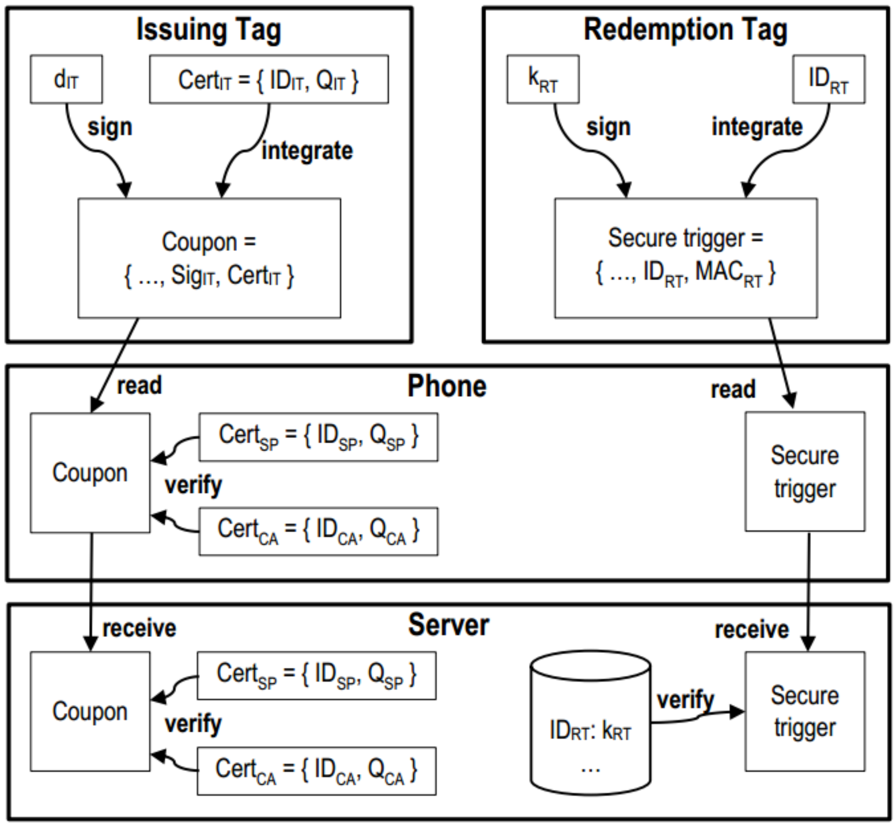


Figure 4.6.: The cryptographic key infrastructure as it is preinitialised in the system, and how the keys are used to verify the coupon and secure trigger data sets.

4. The trusted root certificate Cert_{CA} and the service operator certificate Cert_{SP} are distributed with the coupon software for the phone. Also, the server stores these two certificates for coupon validation.

The redemption tag protects the secure trigger message using a MAC, which is based on a symmetric key scheme. The following initialization steps are necessary:

1. A secret key k_{RT} is generated for each redemption tag, and associated with the tags IDs in the database.

2. The corresponding secret key is also stored securely inside each tag, and used to dynamically calculate the MAC for each secure trigger generated by the tag.

4.2.5. Software components

Java library 'sntbs': ECQV implicit certificate format and encoding

A Java library consolidates common functionality across the software components on phone and server. It is named 'sntbs', which means secure NFC tag based systems. It's core functionality is the handling of NDEF messages (parsing and generation), and the calculation and verification of the cryptographic protection measures.

To make the digital certificates used for signature verification in coupons space-compact, ECQV implicit certificates are used. As an encoding, the minimal encoding scheme (MES) is used, as it is currently favoured by the Security Working Group of the NFC Forum. The specification of MES is given in [23].

Signature and certificate verification. In order to verify a signed NDEF message, and its corresponding certificate, the following steps are carried out:

1. Extract the public key from the certificate according to the ECQV scheme.
2. Verify the digital signature protecting the NDEF message content, using the previously extracted public key.
3. Verify the certificate
   a) Check validity (time).
   b) Check key usage flag (allowed to sign content?)
   c) Check revocation status (has the certificate been revoked by an issuing authority)
   d) Check certificate signature: Not possible nor required for ECQV based certificates, as signature was already implicitly verified by steps (1) + (2)
4. Verify certificate chain, for each certificate check
   a) Validity (time)
   b) Key usage (allowed to issue certificate)
   c) Revocation (has the certificate been revoked by an issuing authority)
   d) Signature using public key of issuer
5. The Last certificate in chain must be the trust anchor (trusted root certificate), otherwise the signature over the NDEF message cannot be considered trusted.

Java Card applet 'T4T_MeCoupon' and 'T4T_SecureTrigger'

These applets construct the NDEF messages as depicted in Table 4.2 and Table 4.3.

Android app 'MeCoupons'

The Android application manages coupons. It communicates with issuing tags to receive new coupons, and it connects to redemption tags to initiate the coupon cash-in. Furthermore, it allows to view and delete coupons. The application consists of two particular views. One allows the user to see all issued and redeemed coupons in a list fashion (main view), the other view displays the details for a specific, selected coupon (coupon view).

Upon tag touch, the app is in either of three states: closed, main view, or coupon view. For best user experience, it does not matter in which state a user touches an issuing or redemption tag. Table 4.5 depicts in which state (closed, main view, coupon view) what functions are triggered, based on the context (issuing or redemption tag touch, or input via UI):

Field	Byte(s)	
Type	1	Type of MES certificate, defines its structure. Type 1 (no extensions) is represented by 0x00, and implies the fields as outlined in this table.
Serial number	8	A unique serial assigned from the CA, that allows to uniquely identify the certificate in context of the issuer.
Curve	1	The ECC curve used for the ECQV based key generation. For secp192r1, the corresponding value is 0x01.
Hash	1	The cryptographic hash function used for the ECQV based key generation. For SHA-256, the corresponding value is 0x01.
Issuer ID	8	Identifier of the issuing CA. This field is required to find the associated issuing certificate when verifying the certificate chain.
Valid from	5	Beginning of certificate validity in Unix time. Unix time is specified as the number of seconds elapsed since January 1, 1970.
Valid to	4	End of certificate validity in seconds since *Valid from*.
Subject ID	8	Identifies the owner of the private key corresponding to the public key in this certificate. For certificates used by tags to sign NDEF messages, this field holds the tag ID.
Key usage	1	Indicates for which purposes this certificate can be used. The value 0x01 denotes digital signatures, which is also the purpose in the context of the demo system.
Public key reconstruction value	25	Value computed by the ECQV algorithm. Allows to reconstruct the public key and to implicitly verify the certificate (when using the public key for signature verification).
Total	62	

Table 4.4.: The Minimal Encoding Scheme for ECQV implicit certificates, specified in [23]. The public key reconstruction value size applies to the EC domain secp192r1.

	Issuing tag touch	Redemption tag touch	Input via UI
App closed	StoreCoupon	RedeemAuto	Start
MainActivity	StoreCoupon	RedeemAuto	TestConnection
CouponActivity	StoreCoupon	RedeemSelected	BrowseCoupons DeleteCoupon ShowDetails

Table 4.5.: Depending on the current state the Android app is in, different actions are triggered by different interactions (tag touch, user input).

- *StoreCoupon*: Receive a coupon from an issuing tag, validate it, and store it in the phone's memory. Display the coupon to the user.
- *RedeemAuto*: Select a suitable coupon, depending on the context of the redemption tag. By matching the vendor ID from the secure trigger record to the vendor ID of the stored coupons, an appropriate coupon is selected. If multiple ones match, select the oldest coupon, with the longest time period since issuing. If no coupon matches, inform the user and discard the secure trigger record (stop). Only if a proper matching coupon was found, start the redemption process by sending the request with both, selected coupon and secure trigger, to the server with the `redeemCoupon(...)` API call.
- *RedeemSelected*: A coupon has already been selected by the user. In that way, it is being displayed in the coupon view upon tag touch. Transfer this coupon with the `redeemCoupon(...)` API call to the server.
- *Start* launches the app, if it is closed. Opens the main view displaying all stored coupons.
- *BrowseCoupons* opens the main view displaying all stored coupons.
- *TestConnection* sends an `hello(...)` API request to the server.
- *DeleteCoupon* removes the coupon from the phone's storage.
- *ShowDetails* displays the NDEF message structure of a coupon to the user.

Server 'MeCouponsGae'

Web service API. The interface via which the phone connects to the web service of the backend is defined in following listing. The request parameters are described in the given annotations.

```
public interface Api {
  /**
   * Redeems the specified coupon using the specified secure trigger.
   * @param couponNdefMessageBytes
   *   NDEF message bytes of a valid meCoupon NDEF message, which has
   *   not yet been redeemed.
   * @param secureTriggerNdefMessageBytes
   *   NDEF message bytes of a valid secure trigger record.
   * @return Indication of (un)successful redemption and details about a
   *         possible denial of redemption.
   */
  public RedeemCouponResponse redeemCoupon(
    byte[] couponNdefMessageBytes,
    byte[] secureTriggerNdefMessageBytes);

  /**
   * A simple hello to which the server replies. Use to test the connection to
   * the server.
   * @param senderName
   *   An arbitrary name of the requester.
   * @return A string containing the current time of the server and the name of
   *   the requester.
   */
  public HelloResponse hello(
    String senderName);
}
```

When the server receives an incoming request, it first verifies the secure trigger:

1. Reconstruct the secure trigger from the received byte stream representing the secure trigger.
2. Verify the secure trigger record:
    a) Verify the tag UID by querying the database table SecureTriggerTag whether it exists. If it exists, also retrieve the corresponding secret key.
    b) Verify the authenticity and integrity of the secure trigger using the MAC field, and the secret key.
    c) Verify the counter value: The value inside the current secure trigger must be larger than any other previously received counter value. Otherwise a newer secure trigger was already used from the same tag, and hence this request is not timely enough anymore.

If any step does not fully succeed, ignore the request and do not carry out any other steps. In any case, log the request and all its parameters. Given a successful verification, continue with validating the coupon:

1. Reconstruct the coupon NDEF message from the received byte stream containing the NDEF message.
2. Verify the coupon:
    a) Verify the coupon record using the signature record, which contains the required ECQV certificate and the signature value. Also verify the certificate chain using the intermediate and trusted root certificates known to the server (ensures that coupon is neither forged nor manipulated).
    b) Verify the tag ID by querying the database. Do the same for vendor ID and offer ID.

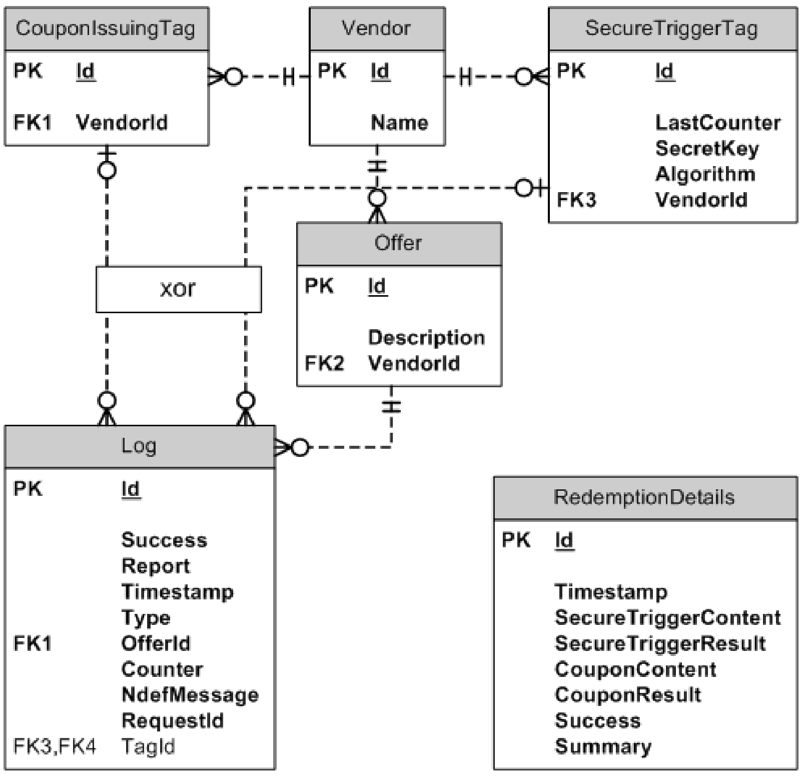


Figure 4.7.: The logical database model for the App Engine datastore.

c) Check the coupon for double spending. Query all previously redeemed offers from this tag, and check that no previously redeemed coupon matches the current tag ID, offer ID and counter value.

If any step does not fully succeed, consider the coupon invalid. In any case, log the coupon parameters and the redemption status to the database (tables `Log` and `RedemptionDetails`.

Database model. The database model depicted in Figure 4.7 serves two main purposes: Firstly, it is preconfigured with all issuing and redemption tags and corresponding information for validation, such as keys. Secondly, it logs all requests to the server, in order to keep track of redeemed coupons and the secure trigger records.

For the server to validate secure trigger records, the table *SecureTriggerTag* knows all tags configured as such. For each tag, the ID is used for lookup. The LastCounter value indicates which counter value was submitted with the latest secure trigger record from this tag, when accessing the server. Using the SecretKey and corresponding algorithm, the MAC field protecting the other secure trigger fields can be verified. Using the VendorId, coupons can be rejected that are redeemed at wrong places by the user.

All coupon issuing tags are configured in table *CouponIssuingTag*. Each issuing tag belongs to a single vendor, but a vendor can have an arbitrary number of offers served by an arbitrary number of issuing tags. A single tag could also issue different offers, e.g. by randomly selecting one.

The tables *Vendor* and *Offer* identify and describe the respective vendor and offer entities.

To keep track of previous requests to the server, a log entry in table *Log* is stored, for each secure trigger and for each coupon received. This table also allows to check whether a specific coupon has already been redeemed. Table *RedemptionDetails* store human readable information about a single redemption process and is used for displaying the redemption result, and its technical details, at the terminal.

5. Implementation of the prototype system

This chapter describes the implementation of the coupon system described in Chapter 4. First, the development tools and the target platforms are introduced. Then, implementation specific aspects of the software components are explained. Finally, the usage and operation of the prototype system is illustrated.

5.1. Development environment and process

5.1.1. Tools

The ADT Bundle, provided by Google[1], includes an Eclipse IDE with the Android development tools (ADT) and the Android SDK. Furthermore, the following plug-ins are required:

- NXP JCOP Plugins Generic 3.4.0 Target 1.4.0 (provided by NXP)
- Google Plugin[2] for Eclipse, which includes the tools and software development kit (SDK) for App Engine and Web Toolkit development

Software testing was carried out with a Samsung Galaxy S3 (Android 4.1) and a Galaxy Nexus (4.2).

5.1.2. Eclipse workspace

A single Eclipse workspace folder organizes all projects related to this thesis and the system implementation. Table 5.1 gives an overview of the code organization.

5.1.3. Development process

The development mainly consisted of three stages, which were carried out sequentially:

1. Using test driven development, the sntbs library (see Table 5.1) was developed with unit tests from the sntbs-test project. In a number of iterations, JUnit test cases were added to the test project for the next set of features to be coded. Then, the implementation in the actual library followed until all test passed, and then the next iteration started.

[1]Android Developers, `http://developer.android.com/sdk/index.html`, accessed on 2013-06-27
[2]Google Developers, `https://developers.google.com/appengine/docs/java/tools/eclipse`, accessed on 2013-06-27

Project/folder	*Description*
LibTestOnAndroid	Android test project to test whether the NDEF functions from the sntbs library provide the same output as the native Android implementation for NDEF messages and records.
meCoupons	Android application project for collecting, managing and redeeming coupons.
meCouponsGae	Google App Engine (GAE) project for the backend system that is hosted on GAE. Also includes the web frontend, developed with Google Web Toolkit (GWT).
sntbs	Java library for the Signature RTD with implicit ECQV certificates. Also implements a full NDEF message and record handling mechanism and the data structures for coupons and secure trigger records. ECQV support includes the generation and validation (public key extraction) of such certificates.
sntbs-test	Java test project for the sntbs library.
T4Tapplet	Java Card applet that implements the Type 4 Tag operation. (provided by NXP)
T4T_MeCoupon	Java Card applet that runs on the issuing tag (based on T4Tapplet).
T4T_SecureTrigger	Java Card applet that runs on the redemption tag (based on T4Tapplet).
TagAuthorEcqvDemo	Demo app that writes and verifies signed NDEF messages to/from arbitrary NFC tags.

Table 5.1.: The Eclipse workspace structure used for development, which contains all related projects.

2. Using the library, a simple tag author app for Android was implemented in an agile manner. Henceforth, the library could be tested in practice with actual NFC tags. The functionality was limited to writing and verifying signed NDEF messages.
3. Finally, the actual coupon demo system was implemented as an evolutionary prototype. First, the issuing use case involving the issuing tag and the Android app was implemented, as it did not yet require the server. Eventually, the redemption use case was implemented, beginning with the tag, and then the backend system.

5.2. Java library 'sntbs'

This library is implemented on the Java SE 1.6 platform, which allows it to be used on Google App Engine, on Android, and on the Desktop (mainly for unit testing during development). It covers a number of functions, which are grouped into their respective namespaces (beginning with `com.nxp.cas.sntbs.*`:

- Cryptographic functions for hashing, ECC, ECDSA, and ECQV are implemented in the namespace `crypto`.
- Classes that represent implicit ECQV certificates, their encoding and parsing, as well as the respective fields can be found in `implicitcerts`.
- A full NDEF parsing functionality is provided by the namespace `ndef`. Although Android provides a limited API to work with NDEF messages and records, this library implementation works on any Java compatible platform (also GAE), and provides more functionality and flexibility (e.g. reflection, see below).
- Some common NDEF types can be found in namespace `knownndef`, such as the URI record, and the Android application record (AAR).
- The specific NDEF message payloads for coupon and secure trigger are contained in the namespaces `mecoupon` and `sectrig`. To encode proprietary data in their payloads, the TLV fields are defined in `tlv`.
- The implementation of the Signature RTD [37], supporting ECQV implicit certificates [23], is given in `sigrtdv2`.
- To create or verify certificates and certificate chains for a PKI, `pki` provides the respective implementation.
- The namespace `reflection` provides a useful utility to inspect the content of data structures such as an NDEF message and their payloads.

5.2.1. ECC and ECQV scheme

The Java Cryptography Architecture (JCA) and the Java Cryptography Extensions (JCE) provide an abstraction layer for application code utilizing cryptography. A service provider is a java library with a specific implementation of cryptographic functions, and not directly visible to the application. Through the use of a factory pattern, instances of the service provider classes are generated. The separation of interface and actual implementation is a result of history and politics: On the one hand export control restrains implementations of certain cryptographic operations, and on the other hand some algorithms might be patented. [15].

BouncyCastle[3] is an open-source provider for JCA/JCE. It includes supports for ECC and ECDSA, and X.509 certificates. [39]

The ECQV related functions have been implemented on top of BouncyCastle: An ECC implementation can be viewed in layers: on the bottom, field operations over prime or binary finite fields build the foundation; next, operations for points on an EC curve, respectively point addition and multiplication, follow. The uppermost layer implements schemes such as ECDSA, which then allow the signing of data. The presented implementation of ECQV is based on the point operations layer given by BouncyCastle. The following code snippet demonstrates the public key reconstruction using point operations from the crypto library. In this case, the JCA/JCE interface had to be bypassed, and the internal provider classes were used to carry out the necessary computations (e.g. class `ECPoint`).

```java
public static ECPublicKey reconstructPublicKey(
                                ECNamedCurveParameterSpec domain,
                                String hash,
                                ECPublicKey publicKeyCa,
                                byte[] publicKeyReconstructionValue,
                                byte[] tbsCertificate) {
  BigInteger e = hash(domain, hash, tbsCertificate);

  ECPoint P_U = Ecqv.convertFromCompressedEncodedEcPoint(
                                domain,
                                publicKeyReconstructionValue);
  ECPoint Q_CA = publicKeyCa.getQ();

  ECPoint Q_U = P_U.multiply(e);
  Q_U = Q_U.add(Q_CA);

  return Ecc.instantiateEcPublicKey(domain, Q_U);
}
```

5.2.2. Reflection

A functionality denoted reflection allows to represent the content of an NDEF message in a tree-like structure in memory, and to convert this representation to a formatted string. Examples of a coupon and a secure trigger NDEF message are given in the appendix in Chapter A. They were generated from actual data using this functionality. The reflection is based on the composite design pattern[4], which is depicted in the adapted form in Figure 5.1. When an NDEF message is parsed, each of its fields get represented by an instance of respective class. Recursively, when the payload is parsed, it gets represented by a corresponding class. In case of a signature record, this payload itself is composed of a number of fields, such as the signature value or the ECQV certificate, each represented by

[3]Bouncy Castle, `http://bouncycastle.org/java.html`, accessed on 2013-07-01

[4]Wikipedia: Composite pattern, `http://en.wikipedia.org/wiki/Composite_pattern`, accessed on 2013-07-02

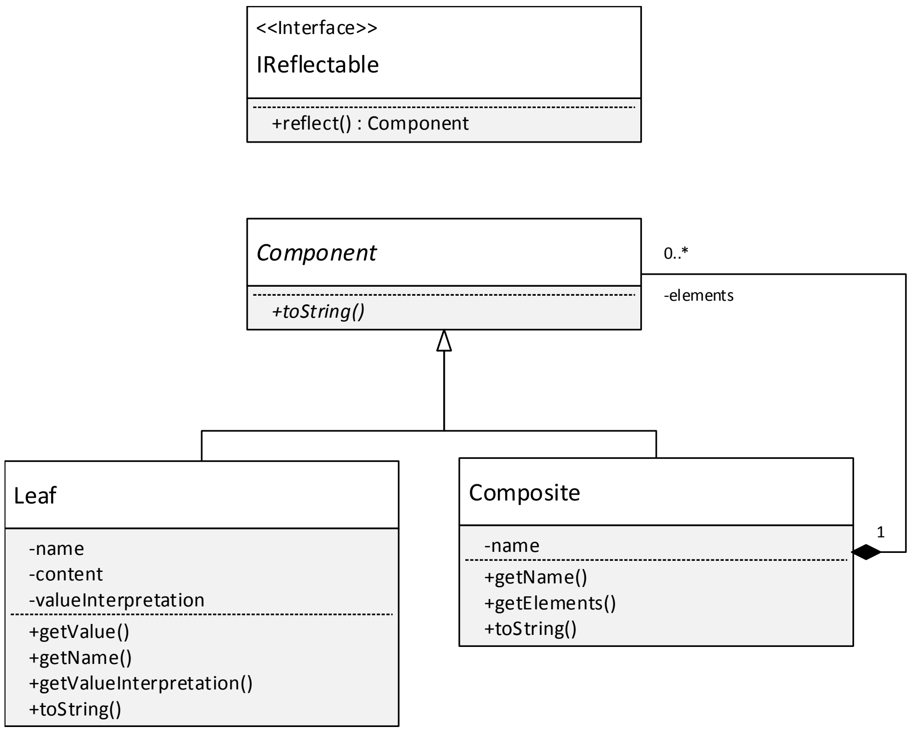


Figure 5.1.: The essential aspect of the reflection functionality in a UML class diagram.

an instance of a dedicated class. The certificate itself then is again composed of its specific fields. Each of these component classes implements the interface `IReflectable`. By calling its only method `reflect()`, a tree like structure of all its components, and recursively their sub-components is returned. Calling `toString()` on the root (or any other) instance extended by the abstract class `Component` gives a string representation of all data fields. This feature allows to easily display the contents of an NDEF message. A generic fallback solution for unknown payloads, as well as a plug-in structure for future extensions yields a flexible architecture to inspect NDEF messages and their specific payloads.

5.2.3. Testing

Test driven development facilitates the development and verification of the library implementation. Table 5.2 gives an overview of the implemented test cases. The test cases were implemented and carried out using JUnit 4.

5.3. JCOP applets

5.3.1. SmartMX and Java Card platforms

SmartMX is a smart card controller IC, which includes an ECC coprocessor. Java Card OpenPlatform (JCOP) is an operating system for smart cards developed by NXP, that runs on SmartMX. Through a Java Card virtual machine, applications written in Java Card

Test	Purpose
CertGenerator	Generates and verifies the prototype PKI with an ECQV end entity certificate, an intermediate certificate, and a root CA certificate (both X.509).
CertificateChainTest	Tests the correct verification of a complete certificate chain.
CouponWithCounterTest	Tests the correct encoding and decoding of a coupon NDEF record.
CryptoAvailability	Tests the platform support for various cryptographic functions.
EccTest	Tests the ECC related functions.
EcqvPublicKeyReconstruction	Tests the reconstruction of the public key from an ECQV certificate.
EcqvTest	Tests the ECQV based key pair generation and related functions.
HelperTest	Tests helper functions.
HmacTest	Tests the HMAC implementation and the secure trigger implementation.
MesCertificateTest	Tests the encoding and decoding of certificates in minimal encoding standard (MES).
NdefMessageTest	Test the construction and parsing of NDEF messages.
PkiSetupPlusSigning	Tests the generation and verification of a three level PKI (with ECQV end entity certificate), and tests the signing and verification of content.
SignatureRecordTests	Tests the signature record implementation.

Table 5.2.: The JUnit test cases to assert the 'sntbs' library.

(a subset of Java), so called *applets*, can be executed on this platform. SmartMX/JCOP is ISO/IEC 14443 compliant, supporting contactless communication, and thus ultimately also NFC.

5.3.2. Type 4 Tag Operation

The Type 4 Tag (T4T) Operation [35] describes three ISO/IEC 7816-4 commands which must be supported by an NFC tag in order to be read by an NFC device operating in reader/writer mode:

- *Select* for selection of applications or files,
- *ReadBinary* to read data from a file of the smart cards memory, and
- *UpdateBinary* to write data to a file

The typical command sequence to select an NDEF application, and read the NDEF message is:

1. Select the NDEF tag application by name `D2760000850101h`
2. Select the capability container (CC) with identifier `E103h`
3. ReadBinary from the selected CC, to get the NDEF file identifier
4. Select the NDEF file with the previously received identifier
5. ReadBinary from the selected NDEF file, to get the NDEF file length contained in the first two bytes (NLEN field)
6. ReadBinary from the selected NDEF file to retrieve the actual message (with offset two bytes)

To implement the coupon and secure trigger generation on top of an applet that supports the aforementioned T4T operation, a hook (see listing) in the NDEF tag application select is implemented. There, the specific NDEF message for the subsequent ReadBinary command is prepared. The following listing gives a simplified overview of a Type 4 Tag applet implemented in Java Card:

```java
public class T4Tapplet extends Applet implements ExtendedLength {
  // [... fields and constants ...]

  public static void install(byte[] bArray, short bOffset, byte bLength) {
    // [...]
    new T4Tapplet(bArray, t).register(bArray, (short) (bOffset + 1),
                                      bArray[bOffset]);
  }

  public T4Tapplet(byte[] bArray, short bOffset) {
    // [... code executed on applet installation ...]
    // initialization of coupon or secure trigger
  }

  public void process(APDU apdu)
```

```
    if (selectingApplet()) {
      // [...]
      // HOOK: prepare the specific coupon or secure trigger NDEF message
    }
    switch (buf[ISO7816.OFFSET_CLA] & (byte) 0xF0) {
      case 0x00: {
        switch (Ins) {
          case 0xA4: {
            // ISO instruction: SELECT FILE
            // change internal state to CC or NDEF file selected

          case 0xB0: {
            // ISO instruction: READ BINARY
            // return the data of selected (CC or NDEF) file

          case 0xD6: {
            // ISO instruction: UPDATE BINARY
            break;
          }
          break;
        }
        break;
      }
    }
  }
}
```

5.3.3. Issuing tag 'T4T_MeCoupon' applet

The ECC domain parameters for the *secp192r1* curve are specified in the class `EccSecp-192r1`. They are required to initialize the private ECC key, which is then required to initialize the `Signature` class that carries out the actual signature computation. The computed signature value is not of fixed length, hence the NDEF message needs to be constructed dynamically. The following steps are carried out by the applet on tag application selection:

1. Increment the counter.
2. Copy (mirror) the counter value into the specific position of the coupon record.
3. Calculate the ECDSA signature value over exactly the entire coupon record.
4. Update the NLEN field, the payload length field of the NDEF message, and the signature length field of the signature record payload.
5. Copy the signature value to the correct position in the signature record.
6. Append the certificate chain to the signature record.

5.3.4. Redemption tag 'T4T_SecureTrigger' applet

The secure trigger record is the only record in the NDEF message served by the redemption tag. Furthermore, the HMAC based on SHA-256 always generates an output of static size.

The process requires the following steps to be carried out by the applet upon each tag application selection.

1. Increment the counter.
2. Copy (mirror) the counter value into the specific position of the secure trigger record.
3. Calculate the MAC over all fields, except the last TLV, of the secure trigger record.
4. Copy the MAC value to the correct position in the secure trigger record.

As the provided JCOP smart card does not support the calculation of an HMAC, this functionality has been implemented manually, according to the definition in Section 2.2:

```
public class SecureTrigger {
  private static final short SHA256_BLOCK_SIZE = 64;
  private static final short SHA256_DIGEST_SIZE = 32;
  private static final byte IPAD = (byte) 0x36;
  private static final byte OPAD = (byte) 0x5C;

  private byte[] keyXorIpad;
  private byte[] keyXorOpad;
  private byte[] tmp;
  private MessageDigest sha256Digest;

  public SecureTrigger(byte[] dest, short offset, byte[] secretKey,
                       byte[] tagUid, byte[] vendorId) {
    // prepare for manual HMAC calculation
    this.keyXorIpad = new byte[SHA256_BLOCK_SIZE];
    this.keyXorOpad = new byte[SHA256_BLOCK_SIZE];
    Util.arrayCopy(secretKey, (short) 0, keyXorIpad, (short) 0,
                   (short) secretKey.length);
    Util.arrayCopy(secretKey, (short) 0, keyXorOpad, (short) 0,
                   (short) secretKey.length);
    SecureTrigger.xor(keyXorIpad, IPAD);
    SecureTrigger.xor(keyXorOpad, OPAD);

    this.tmp = JCSystem.makeTransientByteArray(SHA256_DIGEST_SIZE,
                                               JCSystem.CLEAR_ON_DESELECT);
    this.sha256Digest = MessageDigest.getInstance(MessageDigest.ALG_SHA_256,
                                                  false);
  }

  private void signHmacSha256Manually(final byte[] data,
                                      final short dataOffset,
                                      final short dataLength,
                                      final byte[] out,
                                      final short outOffset) {
    sha256Digest.reset();
    sha256Digest.update(keyXorIpad, (short) 0, (short) keyXorIpad.length);
    sha256Digest.doFinal(data, dataOffset, dataLength, tmp, (short) 0);
    sha256Digest.reset();
    sha256Digest.update(keyXorOpad, (short) 0, (short) keyXorOpad.length);
    sha256Digest.doFinal(tmp, (short) 0, (short) tmp.length, out, outOffset);
  }
  private static void xor(final byte[] data, final byte xor) {
    for (short i = 0; i < data.length; i++) {
      data[i] ^= xor;
    }
  }
}
```

5.4. Android client 'meCoupons'

5.4.1. Android platform

Android[5], currently in version 4.2, is a widespread smart phone operating system developed by Google. Application development can be carried out using the Android SDK and Java; or in C++ using the native development kit (NDK). Android applications are mainly distributed via Google Play, an application distribution platform, but can also be distributed as an Android application package file (APK) via any other channel.

For security purposes, applications are executed inside a sandbox, and must explicitly request permissions from the user to access system resources such as NFC. Upon installing an application, the user can decide whether he grants all required permissions, or denies the installation.

5.4.2. Android NFC API

Support for NFC is available in Android since version 2.3. It supports communication with NFC devices on different protocol layers, from high level NDEF message, to low level commands for receiving and transmitting raw bytes. Basically, as soon as an NFC tag is in proximity of the phone, the operating system decides what action to trigger. A user application utilizes the following APIs in order to facilitate NFC:

1. With the *tag dispatch system*, using a so called intent filter an application is registered for a specific category of data, and gets started automatically by the operating system if such data was read from a tag. If multiple applications register for the same category, a dialog to choose the desired application is shown to the user.

2. With the *Android application record* (AAR), an NDEF record of external type "android.com:pkg", which includes as payload the package name of an Android application, is part of the tags NDEF message. Upon reading a tag, the operating system detects the AAR, and if the application is installed, it is automatically started. Otherwise the Google Play page to install the app is shown to the user.

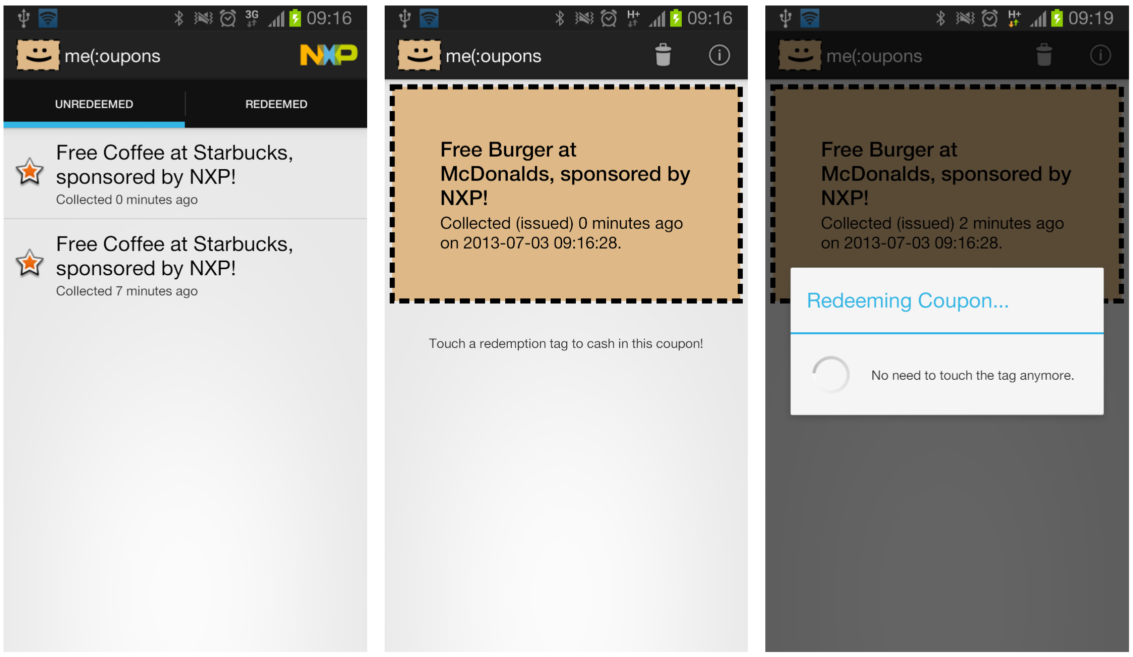
3. Using the *foreground dispatch system*, an active application (that is currently shown on the screen) can take full control of the communication to the NFC device in proximity. It overrides both of the previous two options.

For the presented demo application, the tag dispatch system is facilitated. As both, the coupon and the secure trigger NDEF messages, are of NFC Forum external type, an intent filter could be registered in the `AndroidManifest.xml` file:

[5]Android Developers, `http://developer.android.com/index.html`, accessed on 2013-06-26

```
<intent-filter>
  <action android:name="android.nfc.action.NDEF_DISCOVERED" />
  <category android:name="android.intent.category.DEFAULT" />
  <data
    android:host="ext"
    android:pathPrefix="/nxp.com"
    android:scheme="vnd.android.nfc" />
</intent-filter>
```

To deal with NFC data sets, the SDK provides the classes `NdefRecord` and `NdefMessage` to receive and parse NDEF message. No support for the Signature RTD is provided. Also, the functionality of these provided classes is limited, thus the previously described sntbs library implements a complete NDEF generation and parsing library on its own.



Free Coffee at Starbucks, sponsored by NXP!
Collected 0 minutes ago

Free Coffee at Starbucks, sponsored by NXP!
Collected 7 minutes ago

Free Burger at McDonalds, sponsored by NXP!
Collected (issued) 0 minutes ago on 2013-07-03 09:16:28.

Touch a redemption tag to cash in this coupon!

Free Burger at McDonalds, sponsored by NXP!
Collected (issued) 2 minutes ago on 2013-07-03 09:16:28.

Redeeming Coupon...

No need to touch the tag anymore.

Figure 5.2.: Screenshots of the Android client. In the first screenshot, the main view listing all coupons is shown. Secondly the details for a selected coupon, and finally the process of redeeming a coupon.

5.5. Server 'meCouponsGae'

5.5.1. Google App Engine and Google Web Toolkit

Google App Engine (GAE)[6] is a cloud computing platform offered by Google. It enables the development and hosting of web applications written in Java, Python, and other languages, which are executed in a sandbox environment. Persistence is provided by the App Engine Datastore, which is accessed using Java Data Objects (JDO). Google Web Toolkit (GWT)[7]

[6]Google App Engine, `https://developers.google.com/appengine/`, accessed on 2013-07-02
[7]Google Web Toolkit, `https://developers.google.com/web-toolkit/`, accessed on 2013-07-02

facilitates the front-end development of GAE applications. The web front-end is developed using the Java language, and then cross-compiled to JavaScript to run in the Browser. Remote procedure call (RPC) enables communication with the GAE backend from the browser client.

5.5.2. Web front-end

Using GWT, a web interface represents the cash-in terminal and a management interface. A login mechanism protects the demo system from access, as the URL is public. After login, the following screens can be accessed:

- *Receive coupon* simulates a cash terminal, where as soon as a coupon is redeemed, it is displayed. This view is depicted in Fig.5.3.
- *Log/History* displays a log of all requests sent to the server, and details about the validation of secure trigger and coupon message.
- *Coupon Issuing Tags* lists all tags configured as issuing tags, and their corresponding vendor.
- *Secure Trigger Tags* shows all tags configured as redemption tags, and their respective properties.
- *Offers* gives all offers that are served by issuing tags.
- *Vendors* lists all vendors registered with the system.
- *Settings* allows to reset the system into its initial state by deleting the information about all redeemed coupons, and the history.
- *Log out* leaves the demo front-end and presents the login screen again.

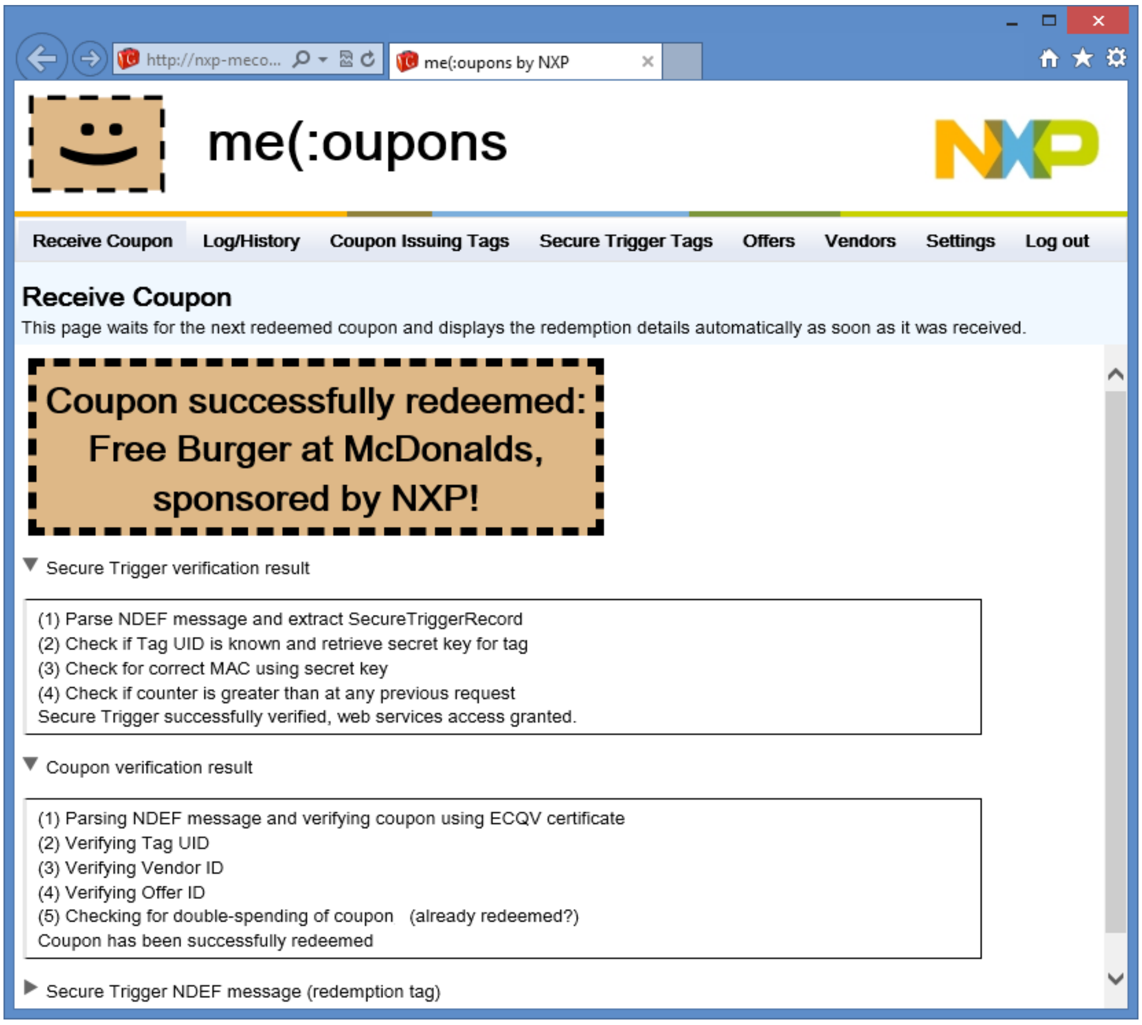
For the receive coupon page, a polling mechanism is implemented. The backend cannot push updates, such as a newly redeemed coupon, to the front-end GUI. Hence, the screen must continuously poll the backend whether a coupon has been received. When the screen is opened, it requests the current time from using the RPC function `getTime()`, which returns the current time. Then, by periodically calling `getRedemp tionDetails(sinceTimestamp)`, the server returns information about the most recent coupon cash-in (whether successful or not), if any occurred since the given time.

5.5.3. Backend system (GAE)

The App Engine backend serves three main purposes. Firstly, with the `DataService`, it provides an interface to the web front-end to retrieve the information it needs to display. Secondly, its `Api` interface can be accessed via Internet to redeem coupons. Finally, it uses the datastore to store and retrieve information about tags, offers, vendors, redeemed coupons, and used secure triggers.

Communication between the backend API and the Android smart phone is facilitated by the Hessian[8] binary web service protocol. The main benefits of Hessian are that it only

[8]Hessian, `http://hessian.caucho.com/`, accessed on 2013-07-02



Receive Coupon

This page waits for the next redeemed coupon and displays the redemption details automatically as soon as it was received.

Coupon successfully redeemed: Free Burger at McDonalds, sponsored by NXP!

▼ Secure Trigger verification result

```
(1) Parse NDEF message and extract SecureTriggerRecord
(2) Check if Tag UID is known and retrieve secret key for tag
(3) Check for correct MAC using secret key
(4) Check if counter is greater than at any previous request
Secure Trigger successfully verified, web services access granted.
```

▼ Coupon verification result

```
(1) Parsing NDEF message and verifying coupon using ECQV certificate
(2) Verifying Tag UID
(3) Verifying Vendor ID
(4) Verifying Offer ID
(5) Checking for double-spending of coupon   (already redeemed?)
Coupon has been successfully redeemed
```

▶ Secure Trigger NDEF message (redemption tag)

Figure 5.3.: The server web front-end, developed using GWT. Currently, the *Receive Coupon* view is being displayed, which simulates the cash terminal where the cashier gets informed that a coupon was redeemed. For the demo system, also the details about the secure trigger, and subsequent coupon verification, are shown in the lower half, to demonstrate what is going on behind the scenes.

requires very little boilerplate code, and it allows an easy transfer of serializable objects between client and server.

5.6. Typical operation of the system

As the system is a prototype and intended for demonstration purposes, it is used by a single operator in the following way.

Set-up. The issuing tags are mounted behind smart posters, represented by two sheets of paper such as depicted in Figure 5.6. These smart posters will denote the advertisements that issue coupons. For the cash-in terminal, a PC with a browser is required, where the web-frontend of the terminal is displayed. Also, the redemption tags are placed close to this terminal. Finally, on the NFC enabled Android smart phone, the meCoupons Android application has to be installed.

Operation. To collect coupons, the user brings his smart phone into proximity of one of the issuing smart poster with the issuing tag attached. The received coupon will be displayed right after the successful touch, such as shown in Figure 5.4.2 in the second screenshot. Multiple coupons can be collected from different issuing smart posters. An overview of all coupons is given in the application's main view, as depicted in the first screenshot.

To redeem a coupon, the user brings his smart phone into proximity of a suitable redemption tag. Coupons of a specific vendor (identified by the vendor ID in the coupon, see Table 4.2), can only be redeemed at redemption tags of the same vendor (vendor ID in the secure trigger, see Table 4.3). The user can either redeem a specific coupon manually by displaying it and then touching the redemption tag. Or, if the app is closed or no specific coupon is selected, the system will automatically redeem a suitable coupon with matching vendor ID. The browser in the terminal will indicate the successful (or unsuccessful) redemption of a coupon, as depicted in Fig 5.3.

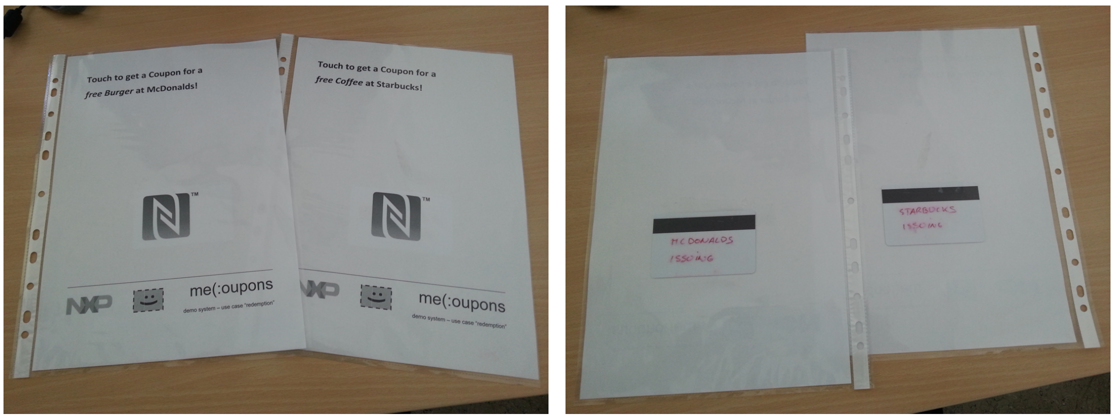

Touch to get a Coupon for a
free Burger at McDonalds!
Touch to get a Coupon for a
free Coffee at Starbucks!
NXP
me(:oupons
demo system – use case "redemption"

Figure 5.4.: Smart posters for two issuing tags based on JCOP contactless smart cards (left). The tags have been attached on the back, right behind the NFC logo to indicate the touch point (right).

6. Results and evaluation

In the following sections, the results of this thesis are evaluated from different perspectives. This chapter relates to the theoretical system study in Chapter 3, the design of the coupon prototype system in Chapter 4, and its implementation given in Chapter 5. Where applicable, future work is elaborated.

6.1. Phone-related security targets require asymmetric cryptography (digital signatures)

From the identified security issues regarding NFC tags given in Chapter 2, the question how the phone can assert the authenticity and integrity of a tag has been concluded. Three security targets, namely the tag content (M1), the tag itself (M2), and the location of the tag (M3), have been identified, and means presented to achieve them (see Table 3.1).

Furthermore, from the requirements of use cases where the backend information system requires a certain proof, e.g. about a users location (see Section 3.1), the question how the server can assert an incoming request has been derived. Here, sub-questions are the authenticity and integrity of the request, its uniqueness and timeliness, and the authentication of the user to the server. These security targets prove to the server that a tag touch took place, at which location it took place, at which time, and by which user (see Table 3.1).

The main finding implies that to achieve the targets summarized in Section 3.6 to at least a certain security level, the tag must support cryptographic operations. These tag capabilities can be divided into two categories. The main differentiation is between cheaper symmetric operations, or more complex asymmetric procedures, and can be summarized as:

- With *symmetric cryptography* the same shared secret key must be available to the proofing as well as all verifying parties. As the insecure phone platform cannot be used to store these keys, no verification is possible there. Yet, the server which knows the key can carry out the necessary verification operations. Hence, symmetric cryptography is suitable if only the backend system needs to verify the request.

- Using *asymmetric cryptography*, parties that verify a digital signature can be given a public key that must not be kept secret. Furthermore, the Signature RTD standardizes how to include a signature value inside a tag's NDEF message, hence the phone can verify the tag content.

6.2. The electronic coupon prototype system

The prototype of a mobile, electronic coupon system designed and implemented (see Chapters 4 and 5) during this thesis leads to a number of findings. Some of these discoveries are advantages to previous systems, while others depend on the intended use case. Also, not all issues can be tackled by the proposed security mechanisms.

Coupon security. From security perspective, coupons are protected against unauthorized generation and manipulation, during their complete lifetime. Before a coupon is transferred from an issuing tag to the phone, its data gets protected by a digital signature, based on asymmetric cryptography and a public key infrastructure. Henceforth, not only the server, but also the phone can verify the coupon. By a counter that is incorporated into the coupon, the backend system can distinguish them from all others issued, and prevent double spending. Yet, a limitation of the current system is that it is still possible to acquire an arbitrary number of coupons from an issuing tag (by consecutively touching it multiple times). Each coupon itself is then unique, and the server cannot detect that a single user redeems multiple coupons of the same offer. Therefore, user authentication might be required, depending on the use case. This limitation is subject to further research.

Server security. The interface through which the phone accesses the server is publicly available via the Internet. This potentially opens a number of attack vectors. As a countermeasure, the redemption process requires a certain kind of cryptographic proof, which the phone must present to the backend system in order to redeem a coupon. This proof is a cryptographically protected piece of data (secure trigger), incorporating a counter for uniqueness. Any request that has an invalid or no secure trigger, is rejected by the system. Implied by this, malicious requests are limited to those attackers that have physical presence to a redemption tag.

Coupon issuing. The issuing process of coupons involves solely the dedicated tag, and the phone of the user. Implicitly, no online connection for neither tag nor the phone is required. Henceforth, this use case is technically less complex, not requiring a backend system. Also, the communication protocol between phone and tag is boiled down to receiving a single NDEF message representing the coupon. This enables novel use cases where coupons are issued at remote locations, such as in a ski resort, where mobile network coverage might be limited. Anyhow, a downside of this approach comes with tag management. The system operator needs to visit each issuing tag in order to configure it for a new coupon, or to disable it.

User anonymity. Users of the system are anonymous. This means, that anyone who installs the application can immediately collect coupons, without registering at the system operator. As privacy and protection from tracing is a desirable aspect for an increasing number of users, the presented approach considers this fact. An implication of this is the anonymity of coupons. The piece of data generated by a tag is not cryptographically connected with a specific user. This association is based purely on the storage of a coupon in the user's phone. If an attacker transfers coupons to another phone, the backend system cannot recognize this circumstance. Yet, in the given system, the coupon can still be redeemed only once, hence this issue is of limited danger to the system operator.

Usability. Considering the usability of the system, the flexibility of the Android platform has been fully exploited. Coupons can be collected, no matter in which state the smart phone is, as long as it is turned on and the screen is unlocked. The same applies for the redemption. Using context information provided by redemption tags, coupons are automatically selected, henceforth a user is not required to start the coupon software manually in order to redeem a coupon. Anyhow, full control is still given to the user, as he can choose to select a specific coupon first, and then touch the tag to cash-in a specific, desired coupon alternatively.

Deployment. The presented prototype eases the deployment and interoperability of an electronic system in various ways. Firsthand, the secure element (SE) of the phone is not used. Henceforth, the system operator does not need to get approval to distribute its application into the SE, which is in practice owned by many different parties. Still, the digital signature protects coupons from manipulation while stored in the phone. Furthermore, the redemption process does not require the tag to communicate with the cash-in terminal directly. A system operator can use rather cheap, passive tags, instead of connecting an NFC interface to its cash terminal, which might even require the re-certification of all affected terminals. Instead, a single web service interface is required, to which the smart phone connects. This interface is then connected to a backend system of the system operator. In case of a retailer, such a system in practice already exists. Furthermore, this approach allows to create vendor or retailer independent system, where a single web service informs the backend systems of different retailers that participate in such a system.

Relay attacks. Finally, a typical NFC based attack cannot be mitigated by any proposed means: relay attacks. As discussed in Section 2.3, no application layer countermeasures can be deployed to hinder such attacks. Investigating this issue is open to future research.

6.3. Recommended NFC tag features

The systems presented in Chapter 3 fulfill certain security targets to secure NFC tag based systems. Therefore, the tag must be capable of dynamically constructing the NDEF message, which enables the security mechanisms.

Generally speaking, three features must be supported by the tag in order to prepare the NDEF message to be read:

1. A tag ID, which is mirrored into a specific part of the NDEF message. This feature is already supported by the latest NXP tag products (see Section 2.1).
2. A counter value, which is automatically mirrored into a specific part the NDEF message. This feature is also already available.
3. A cryptographic mechanism that allows to calculate some value protecting tag ID, counter, and other parts of the NDEF message. This value must also be integrated by the tag at a specific location inside its NDEF message.

The cryptographic function might either be based on symmetric or asymmetric cryptography. The difference, as already presented, is which party is able to verify the integrity: only the backend system, or also the phone of the user. But further practical considerations, need to be accounted for. In most cases, a symmetric primitive produces an output of constant length, such as the HMAC used in the prototype system. This allows a simple mirror functionality to incorporate this data into the desired part of the NDEF message. Using asymmetric cryptography such as ECDSA, the resulting signature varies, for a P-192 curve from 54-56 bytes. Henceforth, the tag must be capable of not just mirroring the value into the NDEF message, but actually constructing the message, and settings length fields accordingly.

As an example, the following scenario (similar to the coupon issuing tag) is given. A tag serves an NDEF message composed of two NDEF records: a URI, and a subsequent signature record (according to the Signature RTD Version 2). Tag ID and counter are then mirrored into specified parts of the URI, so they are transferred to the backend system upon access of the URI by the phone. The signature value must be incorporated into a specified position of the signature record, which requires a number of steps:

1. Calculate the signature value over the first NDEF record.
2. Update the payload length field of the signature record according to the signature length and the static length of the other fields.
3. Update the signature length field of the signature record according to the signature length.
4. Mirror the signature value into the payload of the signature record.
5. Append the certificate chain after the signature value, inside the signature record.
6. Depending on the Type Tag Operation, possibly additional fields such as the NLEN field of a Type 4 Tag, need to be updated, because the signature value also affects the total length of the NDEF message.

Table 6.1 summarizes the possible tag configurations and the resulting verification options, depending on whether the tag is capable of either MACing, or calculating a digital signature using ECDSA. As explained in Chapter 3, the NDEF message received from the tag is then used by the phone to initiate a request to the server, either by accessing a URL (if a URI record is contained), or by using the data contained in the NDEF message to construct a request. Two aspects are of relevance:

1. How is the NDEF message received from the tag verified by the phone?
   a) If a signature record (SR) is contained, verification is carried out according to this specification. Given a future, successfully established ecosystem, this function might be carried out by the operating system of the phone and thus does not require a dedicated application to be installed.
   b) If no particular signature record is contained, a dedicated application (of the service provider) must be installed on the smartphone in order to validate the NDEF message using the contained signature value.

NFC tag				verification			
crypto		NDEF message configuration		phone		server	
type	DL	record 1	record 2	1a	1b	2a	2b
MAC	n	R={ID, C_T, MAC(ID, C_T)}	-	n	n	n	y
	n	UR={ID, C_T, MAC(ID, C_T)}	-	n	n	y	y
ECDSA	y	UR={ID,C_T,Sig(ID, C_T)}	-	n	y	y	y
	y	UR={ID, C_T}	SR={Sig(UR)}	y	y	n	y
	y	R={ID, C_T}	SR={Sig(R)}	y	y	n	y
	y	R={ID, C_T, Sig(ID, C_T}	-	n	y	n	y

Table 6.1.: Different crypto features and tag configurations and the implied verification options (1a-2b). U denotes an NDEF record of any type, while UR is a record of type URI, and SR is record according to the Signature RTD. Column (DL) indicates, whether the resulting signature/MAC value is of dynamic length, and hence makes the dynamic NDEF message construction more complex for the tag. Columns (1a-2b) indicates implications for the signature verification on phone and server side (see text).

2. How are the NDEF message data fields (ID, C_T, Sig/MAC) transferred to the server as part of the request, so that the server can verify the transferred parameters using the signature value?
   a) If the data is already encoded inside a URL as part of an URI record (UR), the default browser of the phone can access the URL, and all the parameters get transferred implicitly as part of the request to the server. No dedicated application is required on the phone.
   b) If the data is encoded in vendor specific fields in the NDEF message, a dedicated application from the service provider is required. This software would then parse the data fields from the NDEF message, and build a request containing them, in order to transmit them to the server.

In order for the phone to verify an NDEF message according to the Signature RTD [37], a certificate must be included in the signature record. A novel approach, introduced with version 2 of the RTD, are ECQV implicit certificates which provide the advantage of greatly reduced sizes (see 6.6.1). Still, it cannot be said to what extent future phones will support this standard, and how the signing ecosystem around will evolve.

6.4. Advantages and limits of implicit NFC tag authentication

To achieve the property of content authenticity (see M1 in Table 3.1) for an NFC tag which serves static content, the tag does not require cryptography. An already signed NDEF message can be written to the tag, and verified by any reader aware of the corresponding public key or certificate chain. But an attacker can easily copy this message to arbitrary tags, which may in certain applications break the desired system security. Hence, tag authenticity itself is also of interest (M2 in Table 3.1). In [46], a solution for tag authentication based on a challenge response protocol is proposed. In contrast, this thesis considers a less complex approach: Through the counter inside the tag, which is incorporated into its NDEF message, the content and hence an attached signature changes upon each tag read (see system RS5 in Figure 3.12). Assuming a tag is sufficiently tamper proof and an attacker cannot extract the private key, he cannot imitate or forge such a tag. The phone verifying the content can conclude the tag is authentic to a certain extent. A possible attacker can still copy the tag (reading a number of NDEF message with increasing counter value), and put this content on a number of tags. But in the context of many systems, this attack has a rather limited scope: The attacker cannot manipulate the NDEF message due to the digital signature. And secondly, a backend system would detect such a behavior rather soon, as multiple requests with the same counter value would be produced. Summarizing, this proposed and implemented (on the issuing tag) ”implicit” tag authentication is not based on challenge response, but still satisfies the security requirements in most cases. In other cases, two-way tag authentication, such as presented by Saeed and Walter [46], should be considered.

6.5. Challenges of tag serialization and system set-up

The implemented prototype (Chapter 5), as well as the security framework (Chapter 3), only partially consider the deployment in real world scenarios, regarding the system set-up, the initialization, and management of the system. Especially the distribution of the required cryptographic key material to tags and user phones requires further elaboration.

From system perspective, the prototype uses a certificate hierarchy with three levels, consisting of a trusted root certificate authority (CA), an intermediate CA, and the certificate for a tag. In practice, the retailer probably is not the only involved party. As depicted in Figure 6.1, the term service operator which was used for the technical discussion, is composed of many parties. The picture is an extension on the "NFC signing eco-system" [18, 44], extended by possible further involved roles: A business decides to utilize such a system and hence contracts an advertising company that carries out the ad campaign, and a system provider that takes care of the technical implementation. The advertiser may contract a company that authors the tags, by putting the ad content on them. These tags are then mounted onto billboards in the public, which might be owned by another different party, the billboard owner. On the other hand, the service provider possibly outsources the application development for the smart phone application. An independent CA issues certificates for tag authoring, and takes care of distributing root certificates to all devices supported by the ecosystem. The prototype certificate hierarchy respects this by using a root level for the CA, an intermediate level for the tag author, and the end entity certificate for the actual tag (and the corresponding private keys).

Regarding a potential tag product, the serialization is an important aspect of mass customization. Current tag products, as presented in Section 2.1, provide a feature called ”mirror” that allows to incorporate tag ID and counter value into a specific part of a tags NDEF message. Using also cryptography, more such mechanisms are required, including the initialization of the cryptographic keys, deployment of certificates, and initialization of the routines where to ”mirror” the results inside the NDEF message.

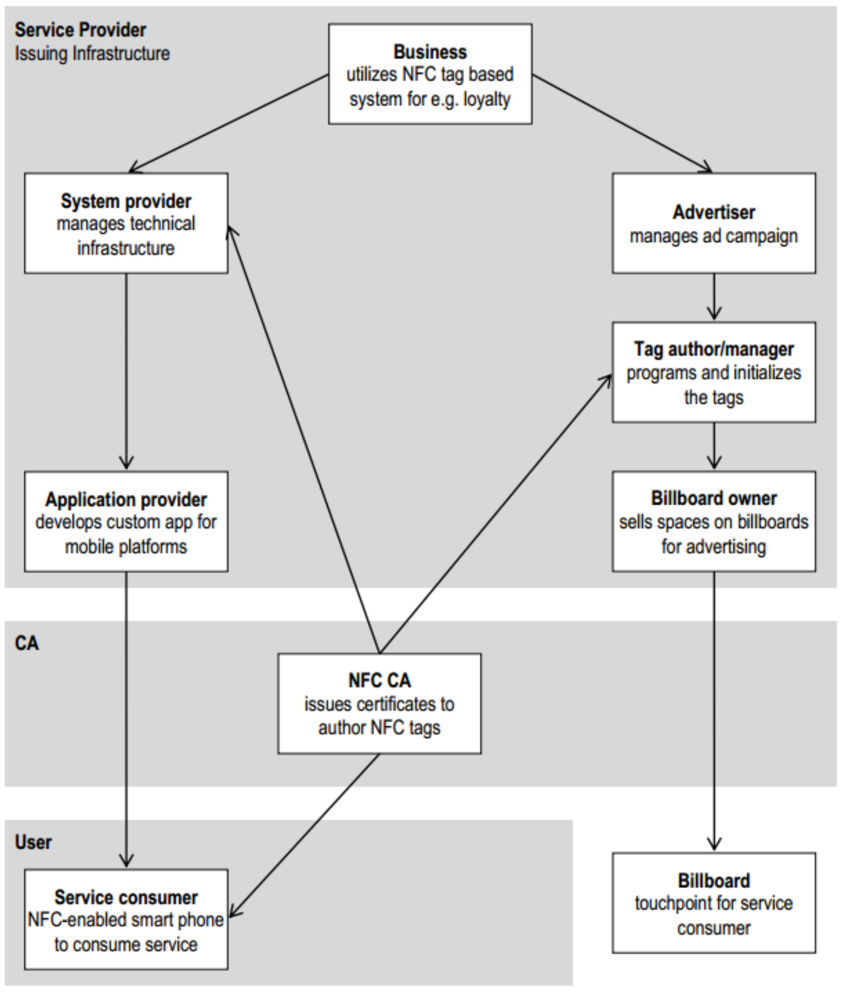

Service Provider
Issuing Infrastructure
Business
utilizes NFC tag based system for e.g. loyalty
System provider
manages technical infrastructure
Advertiser
manages ad campaign
Tag author/manager
programs and initializes the tags
Application provider
develops custom app for mobile platforms
Billboard owner
sells spaces on billboards for advertising
CA
NFC CA
issues certificates to author NFC tags
User
Service consumer
NFC-enabled smart phone to consume service
Billboard
touchpoint for service consumer

Figure 6.1.: Parties involved in a real-world system.

Signature NDEF record sizes in bytes

		signature	certificate	overhead	total
secp112r1	ECQV	35	52	13	100
	X.509	35	229	13	277
secp128r1	ECQV	40	54	13	107
	X.509	40	236	13	289
secp160r1	ECQV	47	58	13	118
	X.509	47	252	13	312
secp192r1	ECQV	54	62	13	129
	X509	54	272	13	339
secp224r1	ECQV	63	66	13	142
	X.509	63	285	13	361
secp256r1	ECQV	72	70	13	155
	X.509	72	304	13	389
secp384r1	ECQV	104	86	13	203
	X.509	104	365	13	482
secp521r1	ECQV	138	104	13	255
	X.509	138	440	13	591

Table 6.2.: Size of a complete signature record [37] for different ECC domain parameters and certificate formats. The values were acquired from the prototype implementation. Overhead accounts for other fields of signature record payload or NDEF record header.

6.6. Digital signatures on NFC tags

Digital signatures enable authentication and integrity for NFC tags and their content and thus allow to fullfill the security targets M1 and M2 (see Table 3.1). In order for the phone to verify the signature, a hierarchy of trust is required that traces back to an ultimately trusted root certificate. This chain of certificates and the verification rules form a public key infrastructure (PKI). From a technical viewpoint, such systems are feasible and sound. yet, a number of organizational issues need to be addressed in the design and deployment.

6.6.1. Advantages and practical considerations for ECQV implicit certificates

In Table 6.2, a comparison of ECQV implicit certificates in minimal encoding scheme (MES, [23]), and traditional X.509 certificates in DER encoding [54] is given. Due to the different certificate encoding, as well as the combination of public key and signature into a single value, an obvious reduction in size can be achieved by ECQV based certificates. The values for the certificates are observations from the prototype implementation. The X.509 certificates do not include any extensions here, hence the given numbers can be considered a lower bound.

Practical considerations. The implementation of the ECQV implicit certificate scheme reveals a number of implications.

1. The public key of the issuer is required, in order to extract the public key of the ECQV certificate. This fact is implicitly given by the public key reconstruction $Q_S = h(Cert_S) \times P_U + Q_{CA}$.

2. Following, the same ECC domain parameters are necessary for both, the ECQV certificate, and the corresponding issuer certificate (of any type). As a consequence, the security of the issuer certificate is of the same security level as the subject certificate, imposed by the domain parameters.

3. Each tag requires its own ECQV certificate. Multiple tags can neither share the same certificate, nor the same public/private key pair. The reason for this is that the tag ID is used as the subject identifier in the certificate. As for each tag, this ID is unique, so each certificate and hence its hash differs. As the public key is reconstructed using this hash, a different public key results.

4. If the tag provides ECC or ECDSA functionality (such as JCOP contact-less smart cards), ECQV implicit certificates are automatically supported as well. For ECQV, key generation and public key extraction are different to traditional certificates, but both of these operations are not carried out on the tag itself. The signing operation purely uses an EC private key, which is not different from a non-ECQV private key. Verification is also carried out by another party than the tag (such as the phone). The only potential issue is that the tag alone cannot generate the private key for the ECQV certificate (so that it never leaves the tag), as this procedure must be carried out in conjunction with the issuing authority.

6.6.2. Signing ecosystem: who is allowed to sign, and how?

The issuance of certificates to sign tag contents, or to equip tags with private keys for dynamic signing, requires careful consideration to whom to issue the certificate. The identification of the subject need to be verified. Strategies to decide who may receive a certificate, and rules about the protection of the associated private key are necessary.

The protection of private keys can also be supported by technical means. A potential novel approach to this issue to be further investigated is a JCOP based signing card. The general idea is based on the assumption that any NFC enabled smart phone is used to author the (static) content of tags. Storing the secret and private signing keys on the smart phone can be considered critical. Tight control and protection of these keys is necessary, but current platforms cannot guarantee this. The issue of malware is well known, which could steal such keys. Instead, the cryptographic material could be stored inside a contactless smart card based on JCOP. When a tag needs to be authored with signed content, the NDEF message is first generated on the smart phone. Then, the signing card is touched. The NDEF message gets transferred to the card, and the card returns the corresponding signature. The phone adds the signature to the NDEF message (as specified by the Signature RTD) and now holds a signed NDEF message. Then, this message is written to the desired tag(s).

Using this approach, private keys need not to be known by tag authors, but are stored securely inside a tamper resistant smart card. Only persons that are in physical possession of this signing card, are able to author tags. Such cards would be issued by the certificate authority, and could employ further protection means such as an internal counter that limits the maximum possible number of singing operations. Also a password protection of the signing card is possible.

In the end, this approach does not require protecting a digital good (the keys), but a physical one (the signing card), which could be easier to achieve.

6.6.3. How does an NFC reader, e.g. a phone, act upon signed/unsigned tags?

Already a lot of NFC tags are available in the field, and many are used for personal purposes. If signed tags once become widespread, all previously deployed unsigned tags become a legacy to be dealt with. Furthermore, tags in personal use cannot be signed as not every individual will have a certificate. Considering a NFC Forum standard for signed tags, the way how an NFC phone acts upon unsigned, unsuccessfully verified and signed tags, must be considered. The following classification in three scenarios is possible:

1. In a *totalitarian* ecosystem, unsigned (or unsuccessfully verified) tags are ignored by the phone. On system level, such as the NFC driver, protocol stack, or the operating system, content received from unauthentic tags is discarded. Neither apps installed, nor system applications can act upon the content received. This approach excludes all current NFC tags from NFC readers that comply to these rules. Furthermore, only tags authored by certified parties can be brought into circulation.

2. In an ecosystem where verification is *optional*, reading devices verify a signature if present, but do not consider unsigned tags harmful. This provides compliance to legacy tags, and allows tags to be authored by anyone, such as for personal use. But it renders the whole signing ecosystem inferior, as attackers could easily bypass the signature verification with unsigned tags.

3. The *hybrid* approach represents a trade-off between the aforementioned two. It differs between (i) signed and valid, (ii) signed but invalid (i.e. unsuccessful verification of signature or certificate), and (iii) completely unsigned tags. Specific ways to differentiate are:

    a) A permission based operating systems like Android (see Section 5.4.1) requires the user to acknowledge the specific permission for each app before it gets installed. By that, two NFC permissions could be provided: (X) read signed tags, (Y) read unsigned tags. An app with permission (X) will not be allowed to handle unsigned or unsuccessfully verified content (as the OS blocks this), whereas an app with permission (Y) can interact arbitrarily via NFC.

    b) A UI indication could inform a user before communicating with an unsigned or unsuccessfully verified tag. Each time this happens, the user must explicitly acknowledge that he wants to continue interacting with this tag. With successfully verified tags, no such warning is shown. This behavior is similar to that of modern browsers and SSL certificates.

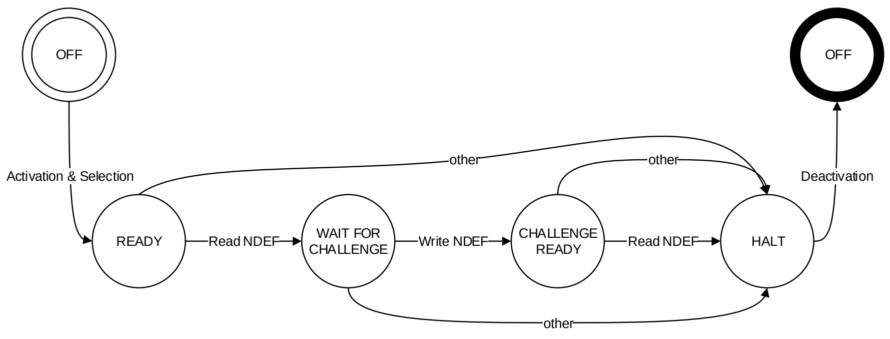


Figure 6.2.: The state machine highlights how, using high level NDEF read and write commands, information is exchanged with an NFC tag.

c) If the operating system differs NFC communication between foreground dispatching and tag dispatch using intent filters (see Section 5.4.2), then unsigned or unsuccessfully verified tags could be ignored with tag dispatch, but allowed in foreground dispatch. This would allow developers of apps to define their own security mechanisms, but rely on the OS if tag dispatch is used.

d) Finally, a differentiation by NDEF record type could be drawn. Well known types, such as a smart poster that includes a URL, must be signed and successfully verified, while proprietary, binary payloads that are only handled by a proprietary app (instead of a system application like the browser), could be unsigned.

With the approaches 3a–3d, both, tags for personal use that are unsigned, and tags used by business (e.g. public ads) could be dealt with in a meaningful manner.

6.7. Challenge-response communication using NDEF message read and write operations

In Chapter 3, some systems required a bidirectional interaction between phone and tag. This two-way communication allows the phone to provide authentication to the tag, and it helps the phone reason about the tag authenticity by providing a challenge. Albeit these ideas have been presented theoretically, they have not been implemented with the prototype. Further research needs to be conducted to explore and study the feasibility of enabling such a bidirectional communication.

The NFC protocol stack offers many layers to build a two-way communication between smartphone and tag. With Type 4 Tags [35], specific APDUs could be used. Also, multiple NDEF files for each protocol step could be defined in the tag's capability container. But in order to provide maximum compatibility with APIs in different smart phone platforms, the communication should take place on NDEF level in means of reading and writing operations onto a single NDEF message (from phone's perspective). Most current smart phone NFC APIs at least support to read and write NDEF messages.

Thus, a mechanism that can be implemented on all NFC Forum Tag Types (1-4), as well as possible future upcoming Tag Types is desirable. A possible implementation on top of a Type 4 Tag is proposed as following: The tags internal state, and the allowed operations are depicted in the state diagram in Figure 6.2. Upon activation and application selection, the mandatory NDEF message can be read (*READY* state). A read operation changes the tags internal state to *WAIT FOR INPUT*. The phone writes the challenge as an NDEF message to the tag, which signs the challenge and waits for it to be read (*CHALLENGE READY*). Then the generated response (e.g. signed challenge) can be read by the phone, before the tag deactivates itself again to run this procedure again.

6.8. Further use cases for advertising, loyalty and coupons

The business scope of this thesis relates to advertising and loyalty marketing. With the prototype implementation (Chapter 5), the specific use case of mobile electronic coupons has been exemplified as feasible. As the focus was primarily on security and usability, different further aspects could be built on top of the presented system solution. The system could be extended in various dimensions:

- *Loyalty card*: The phone application could not just serve as a management software for coupons, but also as replacement for the plastic card that customers carry nowadays. Using a secure trigger tag at the cash desk, and adding user registration to the system, physical cards no longer need to be carried and shown at check out. The smart phone would unify the shopping experience. In detail, the smart phone is used as a (loyalty) card, by acting as a reader on an NFC tag, similar to the system called ”inverse reader mode” in [47]. The customer installs the app of the retail (chain). Every time at the cashier’s desk he touches a specific NFC tag, that triggers a secure process that indicates to the backend system that customer X is present at cashier desk Y in store Z. The backend system immediately informs the computer at the cashier’s desk which customer is present and rewards it with loyalty points. Thus, instead of a loyalty card, users use their smart phone as a proof of membership.
- *Product information*: The concept of "location restricted content delivery" [49] constrains content access to visitors of a certain location. Only where an NFC tag is touched, access to digital content is granted. Thus, in a shop, tags attached to products point the user to a website with additional product information, or multimedia content, to enhance the shopping experience. To limit the distribution of this additional content, users must be physically present in the store, which is ensured by the secure trigger concept.
- *Store discounts for loyal customers*: [49] describes another use case of NFC tags that allow the backend system to ensure physical presence of a user:

  “Retaining customers offer many benefits to a store. Loyal customers are more likely to recommend the store to others, they are more willing to try new products and to spend more money, and their feedback is often more helpful. Thus, many stores are actively looking for new ways to retain their loyal customers by providing them with discounts, coupons, or with

> other rewards. One way to build a loyal customer base is to offer discounts to the customers who visit the store repeatedly or who spend a longer time in the store. With location proofs, customers' devices can gather the location proofs from inside the store; when a discount is available, each customer can prove their loyalty to the store by presenting their set of location proofs collected over time. Similarly, restaurants could offer priority seating for frequent customers. The key benefit of location proofs is that it vastly simplifies the process of keeping track of customers on behalf of the business owner."

  The presented and implemented secure trigger concept facilitates such location proofs.

- *Push coupons*: The system could be further extended by the concept of "push coupons" [5]. Compared to pull coupons such as demonstrated with the prototype, push coupons are sent from the service provider to the users, e.g. via Internet.

Furthermore, the interaction with NFC produces a certain amount of touches. Keeping track of such touches allows system operators to gather usage information about their users and customers, for example when and where coupons were collected: the location can be implicitly derived from tag ID, whereas the time could be provided by the smart phone and stored with the coupon. This temporal information would not be cryptographically protected, but would probably still suffice for marketing analysis.

The integration into the current systems of retailers requires further consideration. In the prototype, a coupon encodes an offer ID and a vendor ID, which are both used by the backend system to relate the issued coupon to an item or service in its data base. How this can be carried out in conjunction with deployed systems needs to be considered.

7. Conclusion and future work

7.1. Conclusion

The theoretical study of NFC tag based systems in Chapter 3 led to the identification of a total of seven security targets. Table 3.1 summarizes the security targets for the phone (M1-M3) and the server (R1-R4). In order to meet these targets, nine possible system solutions have been proposed, and evaluated in terms of security, tag features, and tag touch time. Figure 3.12 compares the systems in regard to their security mechanisms. As a result, the systems essentially require three features on the NFC tags: a counter, a digital signing mechanism based on ECC, and a message authentication code calculator. Thus, a recommendation for future tag products given in Section 6.3 facilitates either symmetric-key (MAC) or asymmetric-key (digital signature) cryptographic capabilities. Whereas a digital signature might be verified by both, phone and server, a symmetric mechanism only allows verification on server side (only R1-R4). In the end, if verification means for the phone (M1-M3) are necessary as well, a digital signature based on asymmetric cryptography must be calculated by the tag.

The design of a prototype system for mobile, electronic coupons utilizing NFC has been explained in Chapter 4. The prototype is based on two of the aforementioned possible system solutions from Chapter 3. The system satisfies coupon specific security such as prevention of double spending and protection against forgery. Furthermore, it facilitates the means for secure access of a backend system, the protection of a user's privacy, and the inference of context. Finally, a benefit of the presented solution is its inverse reader mode, which solely requires NFC tags, but no NFC readers at the system operator side.

The implementation of the prototype system in Chapter 5 exemplifies the feasibility of the previously given design. Furthermore, an implementation of the Signature RTD Version 2, including ECQV implicit certificates, has been carried out and utilized by the prototype system. As digital signatures require digital certificates and a public key infrastructure in order to verify them, ECQV certificates greatly reduce the size of certificates stored on NFC tags, as shown in Table 6.2.

7.2. Future work

In the theoretical analysis of NFC tag based systems in Chapter 3, certain proposed systems require the bidirectional exchange of NDEF messages between NFC enabled phone and NFC tag. As device interoperability is important, a possible approach on top of the NFC Forum reader/writer mode has been given in Section 6.7, but has not been implemented. Also, the feasibility across existing and future smart phone platforms needs further investigation.

For the coupon system prototype designed in Chapter 4, various future enhancements are discussed in Section 6.2. Adding user authentication removes the feature of user anonymity and privacy, but might provide even stronger security for certain use cases. Furthermore, for real world deployment, any aspects of system initialization and tag set-up need to be further investigated, especially in regard to distribution of the cryptographic key material as discussed in Section 6.5.

Furthermore, Section 6.8 lists a number of advertising and loyalty related business cases, which might be added to the given system and which exploit the implemented security mechanisms: loyalty card, product information, store discounts for loyal customers, and the principle of push coupons.

As digital signatures require a public key infrastructure, the signing and issuing of certificates is a central organizational topic discussed in Section 6.6. The two key questions to be further investigated are: firstly, the signing ecosystem, that determines who is allowed to sign and how. From a technical perspective, the idea of a signing card has been proposed, which helps to protect the cryptographic key material at end entities. Secondly, the question how an NFC reader acts upon unsigned or unsuccessfully verified tags was answered with three possible approaches. Further work is required to evaluate the technical and organizational feasibility of these proposed solutions.

A. Example data structures for coupon and secure trigger

This chapter presents two complete NDEF message of a coupon and of a secure trigger. They have been both generated by the respective issuing and redemption tags, as described in Chapter 4. The formatted output has been produced using the reflection functionality of the 'sntbs' library. A `[+]` denotes node, or a composite, but not an actual data field. The actual data is carried by field annotated with `[>]`, where the field name, its associated value, and the interpretation of the value according to the field's context is given.

A.1. Coupon NDEF message

```
[+] NDEF Message
  [+] NDEF Record
    [+] Header
      [+] Flags = 0x94
        [>] MB
              = true
        [>] ME
              = false
        [>] CF
              = false
        [>] SR
              = true
        [>] IL
              = false
        [>] TNF
              = NFC Forum external type: 0x4
      [>] Type length
            = 16
      [>] Payload length
            = 81
      [>] ID length
            = 0
      [>] Type (16 Bytes)
            = "nxp.com:mecoupon": [0x6E78702E636F6D3A6D65636F75706F6E]
      [>] ID (0 Bytes)
            = : [0xEmpty]
    [+] Coupon with counter payload
      [+] TLV element
        [>] Header (2 Bytes)
              = Type = 1, Length = 43 Bytes: [0x012B]
        [>] Value (43 Bytes)
              = Offer = Free Burger at McDonalds, sponsored by NXP!:
                          [0x467265652042757267657220617420D63446F6E616C6473
                             2C2073706F6E736F72656420627920E585021]
      [+] TLV element
        [>] Header (2 Bytes)
              = Type = 2, Length = 8 Bytes: [0x0208]
        [>] Value (8 Bytes)
              = Offer ID: [0x111111111111BEEF]
      [+] TLV element
        [>] Header (2 Bytes)
              = Type = 3, Length = 8 Bytes: [0x0308]
        [>] Value (8 Bytes)
              = Tag UID: [0x000000000000BEEF]
      [+] TLV element
        [>] Header (2 Bytes)
              = Type = 4, Length = 4 Bytes: [0x0404]
        [>] Value (4 Bytes)
              = Counter Value = 42: [0x0000002A]
      [+] TLV element
        [>] Header (2 Bytes)
              = Type = 6, Length = 8 Bytes: [0x0608]
        [>] Value (8 Bytes)
              = Vendor ID: [0x222222222222BEEF]
  [+] NDEF Record
    [+] Header
      [+] Flags = 0x51
        [>] MB
              = false
        [>] ME
              = true
        [>] CF
              = false
        [>] SR
              = true
        [>] IL
              = false
        [>] TNF
              = NFC Forum well-known type: 0x1
      [>] Type length
            = 3
      [>] Payload length
            = 126
      [>] ID length
            = 0
      [>] Type (3 Bytes)
            = "Sig": [0x536967]
      [>] ID (0 Bytes)
            = : [0xEmpty]
    [+] Signature Record Payload
      [>] Version (1 Bytes)
            = 2.0: [0x20]
      [+] Signature
        [>] URI present | Sig. type (1 Bytes)
              = 0 | ECDSA P-192: [0x04]
        [>] Hash type (1 Bytes)
              = SHA1: [0x01]
        [>] Signature/URI length (2 Bytes)
              = 56 Bytes: [0x0038]
        [>] Signature (56 Bytes)
```

```
              = ...:
                            [0x3036021900C44E3D49ED95D6608E22836133A7CAAC7FD0D382C91DDE37021
                               900E507D6577A44F434AE6464A509DB156172A90AEB44E1E990]
        [>] URI present | Cert format | Nbr. of certs (1 Bytes)
              = 0 | 0x2 = ECQV | 1: [0x21]
      [+] Certificate chain
        [+] Certificate 0
          [>] Length (2 Bytes)
                = 62 Bytes: [0x003E]
          [+] MES Certificate (fixed-length encoding)
            [>] Type (SEC4) (1 Bytes)
                  = Type 1 (no extensions): [0x00]
            [>] Serial (8 Bytes)
                  = Number: [0x44084B347D18E707]
            [>] Curve (1 Bytes)
                  = secp192r1: [0x01]
            [>] Hash (1 Bytes)
                  = SHA256: [0x01]
            [>] Issuer (8 Bytes)
                  = CN=9F0D932F6E1E9843: [0x9F0D932F6E1E9843]
            [>] Valid from (5 Bytes)
                  = Wed May 29 13:48:13 UTC 2013: [0x0051A6071D]
            [>] Valid duration (4 Bytes)
                  = 157766396 seconds (= 1826.00 days) since 'Valid from' field:
                                  [0x096752FC]
            [>] Subject (8 Bytes)
                  = CN=000000000000BEEF: [0x000000000000BEEF]
            [>] Key usage (1 Bytes)
                  = digital signature, : [0x01]
            [>] Public key reconstruction value (25 Bytes)
                  = ECC Point:
                                  [0x03FC1E5E9EBEE9F2C1D05B14FFC30799042B0EA36D4AADC274]
```

A.2. Secure trigger NDEF message

```
[+] NDEF Record
  [+] Header
    [+] Flags = 0xd4
      [>] MB
            = true
      [>] ME
            = true
      [>] CF
            = false
      [>] SR
            = true
      [>] IL
            = false
      [>] TNF
            = NFC Forum external type: 0x4
    [>] Type length
          = 15
    [>] Payload length
          = 60
    [>] ID length
          = 0
    [>] Type (15 Bytes)
          = "nxp.com:sectrig": [0x6E78702E636F6D3A73656374726967]
    [>] ID (0 Bytes)
          = : [0xEmpty]
  [+] Secure Trigger payload
    [+] TLV element
      [>] Header (2 Bytes)
            = Type = 3, Length = 8 Bytes: [0x0308]
      [>] Value (8 Bytes)
            = Tag UID: [0x0123456789ABCDEF]
    [+] TLV element
      [>] Header (2 Bytes)
            = Type = 6, Length = 8 Bytes: [0x0608]
      [>] Value (8 Bytes)
            = Vendor ID: [0x222222222222BEEF]
    [+] TLV element
      [>] Header (2 Bytes)
            = Type = 4, Length = 4 Bytes: [0x0404]
      [>] Value (4 Bytes)
            = Counter Value = 50: [0x00000032]
    [+] TLV element
      [>] Header (2 Bytes)
            = Type = 5, Length = 32 Bytes: [0x0520]
      [>] Value (32 Bytes)
            = HMAC SHA256: [0x7AE10256E1F964F8437176
                                        5AF3ACA897FF287E2AF74C6D48E80A28B4AAAFDF62]
```

Glossary

card emulation is the NFC Forum mode where an NFC Forum device emulates a passive NFC tag that responds to reader/writer mode communication.

NDEF message is a common data format for NFC Forum tags and NFC Forum devices specified by the NFC Forum.

NFC device is a NFC-enabled device that operates in either of the three NFC Forum modes reader/writer, peer to peer, or card emulation.

NFC Forum device denotes an NFC device. See NFC device.

NFC Forum tag see NFC tag.

NFC tag is a passive, small and rather cheap NFC device with memory. It is passive as it is not physically connected to a power supply. Power is provided by an NFC device operating in reader/writer mode, which generates a magnetic field. An NFC Forum tag is compatible to one of the NFC Forum tag operation specifications.

peer to peer is the NFC Forum mode where two NFC Forum devices communicate and either device can initiate the communication.

reader/writer is the NFC Forum defined mode where an NFC device initiates communication with an NFC tag.

tag is in the context of this thesis an NFC tag, if not indicated otherwise in the text. See NFC tag.

Acronyms

AA active authentication.

AAR Android application record.

ADT Android development tools.

APDU application protocol data unit.

API application programming interface.

APK Android package.

ASK amplitude shift keying.

CA certificate authority.

CC capability container.

DoS denial of service.

ECC elliptic curve cryptography.

ECDSA elliptic curve digital signature algorithm.

ECQV elliptic curve Qu-Vanstone.

eWOM electronic word-of-mouth.

GAE Google app engine.

GUI graphical user interface.

GWT Google web toolkit.

HMAC keyed-hash message authentication code.

IC integrated circuit.

ICC integrated circuit card.

IDE integrated development environment.

IEC international electrotechnical commission.

ISO	international organization for standardization.
JCA	Java cryptography architecture.
JCE	Java cryptography extension.
JCOP	Java card OpenPlatform.
JDO	Java data objects.
LBS	location based services.
LLCP	NFC logical link control protocol.
MAC	message authentication code.
MES	minimal encoding scheme.
MIME	multipurpose internet mail extensions.
NDEF	NFC data exchange format.
NDK	native development kit.
NFC	near field communication.
NPP	NDEF push protocol.
OS	operating system.
PKI	public key infrastructure.
RFC	request for comments.
RFID	radio frequency identification.
RPC	remote procedure call.
RSA	Rivest, Shamir and Adleman.
RTD	record type definition.
SDK	software development kit.
SHA	secure hash algorithm.
SMS	short message service.
SNEP	simple NDEF exchange protocol.
SNTBS	secure NFC tag based system(s).

T4T	type 4 tag.
TLV	type-length-value.
TTP	trusted third party.
UML	unified modelling language.
URI	uniform resource identifier.
URL	uniform resource locator.
UTF-8	universal character set transformation format – 8 bit.
WKT	well-known type.

Bibliography

[1] M. Aigner, S. Dominikus, and M. Feldhofer. A system of secure virtual coupons using NFC technology. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops' 07. Fifth Annual IEEE International Conference on*, pages 362–366. IEEE, 2007. (Cited on pages 31, 32, and 50.)

[2] C. Blundo, S. Cimato, and A. De Bonis. A lightweight protocol for the generation and distribution of secure e-coupons. In *Proceedings of the 11th international conference on World Wide Web*, pages 542–552. ACM, 2002. (Cited on page 33.)

[3] S.Z. Chiou-Wei and J.J. Inman. Do shoppers like electronic coupons?: A panel data analysis. *Journal of Retailing*, 84(3):297–307, 2008. (Cited on page 29.)

[4] Daniel R. L. Brown. SEC1: Elliptic Curve Cryptography. Technical specification, Certicom Research, May 2009. Version 2.0, `http://www.secg.org/`. (Cited on page 11.)

[5] A. Dickinger and M. Kleijnen. Coupons going wireless: Determinants of consumer intentions to redeem mobile coupons. *Journal of Interactive Marketing*, 22(3):23–39, 2008. (Cited on pages 29, 49, and 94.)

[6] S. Dominikus and M. Aigner. mcoupons: An application for near field communication (NFC). In *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*, volume 2, pages 421–428. IEEE, 2007. (Cited on pages 31 and 32.)

[7] Klaus Finkenzeller. *RFID handbook 3rd Edition*. Wiley, 2010. (Cited on pages 9, 14, and 15.)

[8] L. Francis, G. Hancke, K. Mayes, and K. Markantonakis. Practical NFC peer-to-peer relay attack using mobile phones. *Radio Frequency Identification: Security and Privacy Issues*, pages 35–49, 2010. (Cited on pages 14 and 17.)

[9] F. Gallo and C. Lesjak. Secure smart poster, August 15 2012. EP Patent 2,487,629. (Cited on pages 26 and 27.)

[10] G.P. Hancke. A practical relay attack on iso 14443 proximity cards. *Technical report, University of Cambridge Computer Laboratory*, 2005. `http://www.rfidblog.org.uk/hancke-rfidrelay.pdf`. (Cited on pages 14 and 17.)

[11] G.P. Hancke, KE Mayes, and K. Markantonakis. Confidence in smart token proximity: Relay attacks revisited. *Computers & Security*, 28(7):615–627, 2009. (Cited on pages 14 and 17.)

[12] E. Haselsteiner and K. Breitfuß. Security in near field communication (NFC). In *Workshop on RFID Security RFIDSec*, 2006. (Cited on pages 13, 14, 16, and 17.)

[13] C.M. Henderson, J.T. Beck, and R.W. Palmatier. Review of the theoretical underpinnings of loyalty programs. *Journal of Consumer Psychology*, 21(3):256, 2011. (Cited on page 28.)

[14] Paul Holleis, Gregor Broll, and Sebastian Böhm. Advertising with NFC. In *Workshop on Pervasive Advertising and Shopping, in conjunction with the 8th International Conference on Pervasive Computing (Pervasive 2010), Helsinki, Finland*, 2010. (Cited on page 28.)

[15] David Hook. *Beginning cryptography with Java.* John Wiley & Sons, 2005. (Cited on page 71.)

[16] Han-Cheng Hsiang. A simple and secure mobile coupons scheme. In *Conference on Information Technology and Applications in Outlying Islands*, 2009. (Cited on pages viii and 34.)

[17] S.C. Hsueh and J.M. Chen. Sharing secure m-coupons for peer-generated targeting via ewom communications. *Electronic Commerce Research and Applications*, 9(4): 283–293, 2010. (Cited on page 30.)

[18] M. Kilas. Digital signatures on NFC tags. *Unpublished master's thesis, KTH Royal Institute of Technology, Sweden*, 2009. (Cited on page 87.)

[19] H.S. Kortvedt. Securing near field communication. Master's thesis, Norwegian University of Science and Technology, 2009. `http://ntnu.diva-portal.org/smash/get/diva2:347744/FULLTEXT01`. (Cited on pages 14 and 16.)

[20] Mikko Lehtonen, Thorsten Staake, and Florian Michahelles. From identification to authentication–a review of RFID product authentication techniques. In *Networked RFID Systems and Lightweight Cryptography*, pages 169–187. Springer, 2008. (Cited on pages 24 and 25.)

[21] Gerald Madlmayr, Josef Langer, Christian Kantner, and Josef Scharinger. NFC devices: Security and privacy. In *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, pages 642–647. IEEE, 2008. (Cited on pages 14 and 21.)

[22] Konstantinos Markantonakis, Michael Tunstall, Gerhard Hancke, Ioannis Askoxylakis, and Keith Mayes. Attacking smart card systems: Theory and practice. *Information Security Technical Report*, 14(2):46–56, 2009. (Cited on pages 14 and 16.)

[23] Matthew Campagna. SEC4: Ellipitc Curve Qu-Vanstone Implicit Certificate Scheme (ECQV). Technical specification, Certicom Research, Jan 2013. Version 1.0, `http://www.secg.org/`. (Cited on pages 11, 62, 63, 70, and 89.)

[24] A.J. Menezes, P.C. Van Oorschot, and S.A. Vanstone. *Handbook of applied cryptography.* CRC, 1996. (Cited on page 13.)

[25] Charlie Miller. Exploring the NFC attack surface. In *Black Hat Briefings*. Accuvant Labs, August 2012. `http://media.blackhat.com/bh-us-12/Briefings/C_Miller/BH_US_12_Miller_{NFC}_attack_surface_WP.pdf`. (Cited on pages 5, 14, and 20.)

[26] Collin Mulliner. Vulnerability analysis and attacks on NFC-enabled mobile phones. In *Availability, Reliability and Security, 2009. ARES'09. International Conference on*, pages 695–700. IEEE, 2009. (Cited on pages 14, 19, and 20.)

[27] Avinash Nandwani, Paul Coulton, and Reuben Edwards. Using the physicality of nfc to combat grokking of the check-in mechanic. In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, MindTrek '11, pages 287–290, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0816-8. doi: 10.1145/2181037.2181087. URL `http://doi.acm.org/10.1145/2181037.2181087`. (Cited on page 14.)

[28] B Clifford Neuman and Theodore Ts'o. Kerberos: An authentication service for computer networks. *Communications Magazine, IEEE*, 32(9):33–38, 1994. (Cited on pages 13 and 38.)

[29] NFC Forum. NFC Data Exchange Format (NDEF). Technical specification, NFC Forum, July 2006. Version 1.0, `http://www.{NFC}-forum.org/specs/`. (Cited on pages 6 and 7.)

[30] NFC Forum. NFC Record Type Definition (RTD). Technical specification, NFC Forum, July 2006. Version 1.0, `http://www.{NFC}-forum.org/specs/`. (Cited on page 6.)

[31] NFC Forum. Smart Poster Record Type Definition. Technical specification, NFC Forum, July 2006. Version 1.0, `http://www.{NFC}-forum.org/specs/`. (Cited on page 6.)

[32] NFC Forum. Signature Record Type Definition. Technical specification, NFC Forum, Nov 2010. Version 1.0, `http://www.{NFC}-forum.org/specs/`. (Cited on page 6.)

[33] NFC Forum. NFC Activity Specification. Technical specification, NFC Forum, Nov 2010. Version 1.0, `http://www.{NFC}-forum.org/specs/`. (Cited on page 6.)

[34] NFC Forum. NFC Digital Protocol. Technical specification, NFC Forum, November 2010. Version 1.0, `http://www.{NFC}-forum.org/specs/`. (Cited on page 6.)

[35] NFC Forum. Type 4 Tag Operation Specification. Technical specification, NFC Forum, June 2011. Version 2.0, `http://www.{NFC}-forum.org/specs/`. (Cited on pages 6, 74, and 92.)

[36] NFC Forum. NFC Analog Specification. Technical specification, NFC Forum, July 2012. Version 1.0, `http://www.{NFC}-forum.org/specs/`. (Cited on page 6.)

[37] NFC Forum. Signature Record Type Definition. Candidate technical specification, NFC Forum, April 2013. Version 2.0, `http://www.{NFC}-forum.org/specs/candidate_spec_list/`. (Cited on pages 6, 18, 58, 59, 70, 86, and 89.)

[38] NFC Forum. Simple NDEF exchange protocol. Technical specification, NFC Forum, May 2013. Version 1.0, `http://www.{NFC}-forum.org/specs/`. (Cited on page 26.)

[39] The Legion of the Bouncy Castle. Bouncy castle crypto apis, 2013. `http://www.bouncycastle.org/`, accessed on 2013-03-05. (Cited on page 71.)

[40] C. Paar and J. Pelzl. *Understanding cryptography: a textbook for students and practitioners.* Springer, 2010. (Cited on page 10.)

[41] M. Roland. Effects of android's sms-uri parsing bug on NFC applications, May 2011. `http://www.mroland.at/fileadmin/mroland/papers/201105_SMSURIBug.pdf`. (Cited on pages 14 and 18.)

[42] 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*, pages 65–70. IEEE, 2011. (Cited on pages 14, 18, and 19.)

[43] Michael Roland, Josef Langer, and Josef Scharinger. Applying relay attacks to google wallet. In *Near Field Communication (NFC), 2013 5th International Workshop on*, pages 1–6. IEEE, 2013. (Cited on page 17.)

[44] T. Rosati and G. Zaverucha. Elliptic curve certificates and signatures for NFC signature records, 2011. `http://www.{NFC}-forum.org/resources/white_papers/Using_ECQV_ECPVS_on_{NFC}_Tags.pdf`. (Cited on page 87.)

[45] M.Q. Saeed and C.D. Walter. A record composition/decomposition attack on the NDEF signature record type definition. In *Internet Technology and Secured Transactions (ICITST), 2011 International Conference for*, pages 283–287. IEEE, 2011. (Cited on pages 14 and 19.)

[46] Muhammad Qasim Saeed and Colin D Walter. Off-line NFC tag authentication. In *Internet Technology And Secured Transactions, 2012 International Conferece For*, pages 730–735. IEEE, 2012. (Cited on pages 24, 86, and 87.)

[47] C Saminger, S Grunberger, and J Langer. An NFC ticketing system with a new approach of an inverse reader mode. In *Near Field Communication (NFC), 2013 5th International Workshop on*, pages 1–5. IEEE, 2013. (Cited on pages 25, 40, and 93.)

[48] J.J. Sánchez-Silos, F.J. Velasco-Arjona, I.L. Ruiz, and MA Gomez-Nieto. An NFC-based solution for discount and loyalty mobile coupons. In *Near Field Communication (NFC), 2012 4th International Workshop on*, pages 45–50. IEEE, 2012. (Cited on page 30.)

[49] S. Saroiu and A. Wolman. Enabling new mobile applications with location proofs. In *Proceedings of the 10th workshop on Mobile Computing Systems and Applications*, page 3. ACM, 2009. (Cited on pages 22, 37, and 93.)

[50] Bruce Schneier. *Secrets & lies.* dpunkt-Verlag, 2001. (Cited on page 9.)

[51] Bruce Schneier. *Liars and outliers: enabling the trust that society needs to thrive.* Wiley, 2012. (Cited on page 9.)

[52] P. Schoo and M. Paolucci. Do you talk to each poster? security and privacy for interactions with web service by means of contact free tag readings. In *Near Field Communication, 2009. NFC '09. First International Workshop on*, pages 81 –86, feb. 2009. doi: 10.1109/{NFC}.2009.20. (Cited on pages 14, 15, and 18.)

[53] B. Sharp and A. Sharp. Loyalty programs and their impact on repeat-purchase loyalty patterns. *International Journal of Research in Marketing*, 14(5):473–486, 1997. (Cited on page 28.)

[54] Telecommunication Standardization Sector. X.509 : Information technology – Open systems interconnection – The Directory: Public-key and attribute certificate frameworks. Recommendation, ITU, Nov 2008. `http://www.itu.int/rec/T-REC-X.509-200811-I/en`. (Cited on pages 11, 58, and 89.)

[55] Peter Teufl, Thomas Zefferer, Sandra Kreuzhuber, and Christian M. Lesjak. Trusted location based services. In *Internet Technology And Secured Transactions, 2012 International Conferece For*, pages 185–192. IEEE, 2012. (Cited on pages 14, 21, 23, 37, and 45.)

[56] R. Verdult and F. Kooman. Practical attacks on NFC enabled cell phones. In *Near Field Communication (NFC), 2011 3rd International Workshop on*, pages 77–82. IEEE, 2011. (Cited on page 20.)

[57] D. Volland, K. Noyen, O. Kayikci, L. Ackermann, and F. Michahelles. Switching the role of NFC tag and reader for the implementation of smart posters. In *Near Field Communication (NFC), 2012 4th International Workshop on*, pages 63–68. IEEE, 2012. (Cited on page 25.)

[58] Zhao Wang, Zhigang Xu, Wei Xin, and Zhong Chen. Implementation and analysis of a practical NFC relay attack example. In *Instrumentation, Measurement, Computer, Communication and Control (IMCCC), 2012 Second International Conference on*, pages 143 –146, dec. 2012. doi: 10.1109/IMCCC.2012.40. (Cited on pages 14 and 17.)

[59] Thomas Wiechert, Andreas Schaller, and Frederic Thiesse. Near field communication use in retail stores: Effects on the customer shopping process. In *Proceedings*, volume 4, pages 137–141, 2008. (Cited on page 28.)