An Attack on Signed NFC Records and Some Necessary Revisions of NFC Specifications

Muhammad Qasim Saeed
ISG, Department of Mathematics
Royal Holloway University of London
Egham, UK
muhammad.saeed.2010@live.rhul.ac.uk

Colin D. Walter
ISG, Department of Mathematics
Royal Holloway University of London
Egham, UK
Colin.Walter@rhul.ac.uk

Abstract

The Signature Record Type Definition was released by the Near Field Communication (NFC) Forum to provide integrity and authenticity to the NFC Data Exchange Format (NDEF). It achieves this goal by adding a digital signature and corresponding certificates to the NDEF message. Although the Signature Record Type Definition (Signature RTD) specifies the use of strong cryptographic algorithms like RSA, DSA and ECDSA, a few vulnerabilities have been discovered in its implementation. A recently published Record Composition Attack by Roland et al. (2011) describes how data can be modified in an NDEF message by exploiting the Type Name Format (TNF) field even though the NDEF message is protected by a Signature Record.

This paper takes a close look at the attack and points out that, apart from the TNF value, a few other fields of the NDEF header must also be manipulated in order to implement this attack successfully. It is shown how to do this and some necessary modifications to the signature scheme are proposed in order to counter such attacks. Our main contribution is proposing a revision to the Signature specification by signing more fields but keeping the existing NDEF specification.

1. Introduction

The format of Near Field Communication (NFC) messages is covered by the NFC Data Exchange Format (NDEF) specification [1], published by the NFC Forum, and signatures on such messages are defined by the Forum's Signature Record Type Definition (Signature RTD) in [2]. This paper takes a critical look at the signature specification, explores some of its vulnerabilities and proposes countermeasures to two straightforward attacks on signed messages under the current NFC specifications.

The first part introduces technical aspects of Near Field Communication (NFC), including the format for NFC messages and the originally proposed digital signature scheme [2]. After this, two attacks on the signature scheme by Roland *et al*. [3] are presented and some missing critical detail is identified. The first novel contribution is a fuller presentation of their Record Composition attack which now certainly succeeds. More significantly, however, we need to propose a revision of the signature scheme which is essential to counter both of Roland's attacks. Some detailed discussion is required to justify the inclusion of various new elements and to explain their construction.

Lastly, we also need to discuss implementation issues arising from our proposed modifications in order to integrate them within the current infrastructure.

Our original conference report [12] on the attacks suggested the necessity for a revision of the NDEF specification. Here, we manage to avoid this, with its attendant implications for the existing NFC infrastructure, by making some further changes to the proposals for the signature scheme. However, this proposed revision to the signature specification is nevertheless a significant result because of its implications for extending and renewing the current NFC infrastructure.

2. Near Field Communication

Near Field Communication (NFC) is a short-range wireless technology compatible with contactless smart cards (ISO/IEC 14443) and radio-frequency identification (RFID) [4]. NFC is on the 13.56 MHz frequency band and operates at a distance of less than 4 cm. It uses magnetic field induction for communication and powering the chip.

NFC technology has a number of applications such as ticketing and payment, retrieving information from information kiosks or setting up connections between devices (so called *device pairing*). A wide variety of applications is possible using the

technology because of the different operation modes supporting both communication from device to device (peer-to-peer mode), communication between a device and a passive tag (read/write mode) and an emulation mode where a device can act like a contactless smart card [5].

NFC technology is now available on cell phones resulting in a sharp rise in its use. It makes life easier and more convenient for consumers around the world by making it simpler to make transactions, exchange digital content, and connect to electronic devices with a touch. Multiple modes allow consumers to perform contactless transactions, connect to peer devices for data sharing or interact with a variety of contactless "smart objects" through their mobile devices. For example, a customer could initiate the process of buying a cinema ticket by touching his NFC mobile phone against a smart movie poster. NFC technology provides the capability for ticketing, banking and other applications, which were historically installed on contactless security tokens, to be implemented on mobile devices. These functionalities allow NFC-enabled mobile phones to be used as if they were contactless smart tokens, e.g. for retail payments at point of sale (POS) terminals or swiping an e-ticket at a turnstile. They also provide the opportunity for a single device to contain multiple tokens [6]. This technology is highly suitable for monetary transactions (especially micro-payments) because of its shorter range and compatibility with contactless smart cards.

3. The NFC Forum

The NFC Forum was established in 2004 to standardize applications which use NFC [7]. The NFC Forum promotes sharing, pairing, and transactions between NFC devices or tags. In June 2006, the Forum formally outlined the architecture of NFC technology. One such use of NFC tags is in so-called *Smart Posters*. These contain information such as the Title, an SMS, and a URL or electronic business card. The user can access this information simply by touching his cell phone on such tags. Apart from displaying the information to the user, the smart poster can also trigger an action such as opening a specific website, calling the telephone number stored in the poster, etc [8]. (This should, of course, be subject to the consent of the user's security policy settings.)

With the increasing number of available applications of NFC technology, threats of its abuse are also emerging in parallel. In the case of abuses related to smart posters, an attacker may replace the URL address or the telephone number with malicious content. Consequently, it is essential to guarantee the integrity and authenticity of NFC data.

The NFC Forum developed the Signature Record Type Definition (Signature RTD) in 2010 to fix such problems [2]. Its main objective is to digitally sign the data fields of an NDEF message, thus providing integrity and authenticity.

3.1. NFC Data Exchange Format

The NFC Data Exchange Format (NDEF) specification defines a common format and rules to exchange information in the NFC environment. NDEF is a lightweight, binary message format that can be used to encapsulate one or more application-defined payloads of arbitrary type and size into a single message construct. Each payload is described by a type, its length, and an optional identifier. A record is the unit for carrying the payload within an NDEF message. An NDEF message contains one or more NDEF records [1]. The structure of an NDEF record is shown in Figure 1.

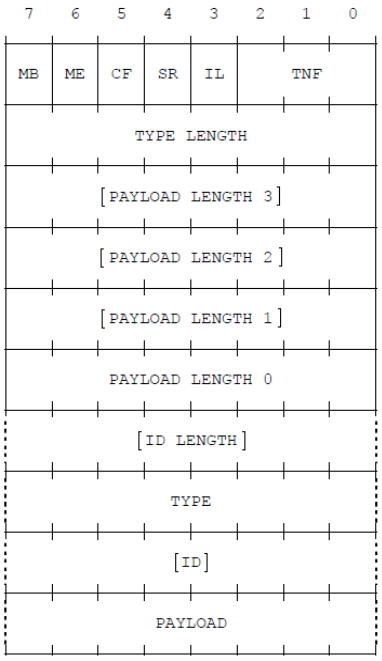

Figure 1. NDEF Record Layout ([1], fig. 3)

The Message Begin *(MB)* and Message End *(ME)* bits indicate the first and the last record of an NDEF message respectively. The Chunk Flag *(CF)* specifies that the payload of that record is continued in the next record. Short Record *(SR)* is a 1-bit flag which, if set, indicates that the size of the *Payload-Length* field is one byte. In this case, the size of *Payload* is restricted to between 0 and 255 bytes. Otherwise, the *Payload-Length* field consists of 4 bytes (as shown in Figure 1) and it determines a *Payload* size ranging from 0 to 2^{32}-1 bytes. The flag *IL* determines whether or not the optional *ID* field and corresponding *ID-Length* field are present.

The Type Name Format *(TNF)* is a 3-bit field indicating the structure of the *Type* field which gives the type of *Payload*. Its value ranges between 0 and 7 with meaning as shown in Table 1. *Type-Length* and

ID-Length are unsigned 8-bit integers that specify the length in octets of the *Type* field and *ID* field respectively. The *Type* field describes the type of *Payload*, and the optional *ID* field is a URI reference.

Table 1. Type Name Format (TNF) Description (cf [1], table 1)

TNF	Meaning
0	The record is empty and there is no payload or type associated with this record. The corresponding length fields are set to zero. This TNF value can be used whenever an empty record is needed.
1	Indicates that the *Type* field contains a value that satisfies the RTD type name format defined in the NFC Forum RTD specification, such as Smart poster RTD, Signature RTD, URL RTD etc.
2	*Type* is a MIME media type identifier (RFC 2406).
3	*Type* is an absolute URI (RFC 3986).
4	*Type* is an NFC Forum external type.
5	*Type* is of unknown format. It is used when the type of the payload is unknown. When used, the *Type-Length* field must be zero and thus the *Type* field is omitted. In this case, the payload is stored but not processed.
6	The record continues the payload of the preceding chunked record. When used, the *ID-Length* and *ID* fields are omitted, the *Type-Length* field must be zero and there is no *Type* field.
7	Reserved for future use.

3.2. Record Chunks

A record chunk carries a chunk of a payload. It can be used to partition dynamically generated contents or very large entities into multiple sequential record chunks within an NDEF message. Every chunked payload is encoded as an initial record chunk followed by zero or more middle record chunks and concluded with a terminating record chunk [1].

The initial record chunk has its *CF* flag set. The *Type* field and the *ID* field (if present) indicate the *Type* and *ID* of the entire payload respectively. The *Payload-Length* field indicates the size of payload of the initial record only.

The *middle* and *terminating* record chunks do not have *Type* and *ID* fields as these are already indicated in the initial chunk. Their *TNF* field value is 6, indicating that the *Type* and *ID* are the same as for the initial record chunk. Their *Type-Length* and *ID-Length* fields are zero and absent respectively. The *CF* is set for *middle* chunks and is clear for the *terminating* chunk.

4. The Signature Record Type Definition

The Signature Record Type Definition specifies the format used when signing single or multiple NDEF records [2]. It defines a list of suitable algorithms and certificate types that can be used to create the signature. It provides users with the possibility of verifying the authenticity and integrity of the data within the NDEF message.

4.1. The Signature Record

The contents of the payload of a signature record consist of three parts: *Version*, digital *Signature* and *Certificate Chain* as shown in Figure 2. The *Version* is a single byte field indicating the version of the specification to which a signature is compliant. Currently the only valid version is 1. The *Signature* field contains either the actual signature or a URI reference to a signature. The signature RTD supports RSA, DSA and ECDSA. The *Certificate Chain* contains the certificate format, the total number of certificates, the list of certificates and an optional URI reference.

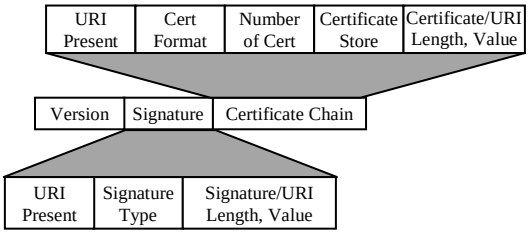

Figure 2. Payload of an NDEF signature record, based on [3], fig. 2

4.2. Use of the Signature Record in an NDEF Message

The signature record applies to all preceding records, starting either from the first record of an NDEF message or from the first record following the preceding signature record as shown in Figure 3. Signature Record 1 signs Records 1 and 2. It also marks the start of the signature of Record 3. Signature Record 2 signs Record 3 only whereas Record 4 has no signature. The signature is applied to the *Type*, *ID* (if present) and *Payload* of these records. The NDEF header and length fields are not signed as shown in Table 2.

In case where only a selection of records in an NDEF message is required to be signed, an *empty signature record* can be inserted into the record sequence to act as a start marker. An *empty signature record* has *TNF*=1 (as it is NFC Forum *well-known*

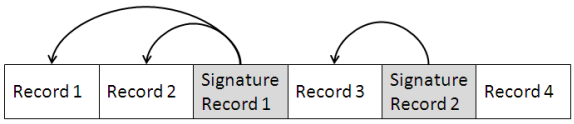

Figure 3. An NDEF message consisting of multiple records [3], fig. 3.

type record), *Type* as 'Sig' and there is no payload associated with this record. Such a record indicates that the preceding records back to the start or the previous signature record are unsigned.

Table 2. Signing an NDEF Record ([2], §3.4)

Field Name	Signed/Unsigned
Message Begin *(MB)*	Not Signed
Message End *(ME)*	Not Signed
Chunk Flag *(CF)*	Not Signed
Short Record *(SR)* Flag	Not Signed
ID-Length *(IL)* Present Flag	Not Signed
Type Name Format *(TNF)*	Not Signed
Type-Length	Not Signed
Payload-Length	Not Signed
ID-Length	Not Signed
Type	Signed
ID	Signed
Payload	Signed

5. Related Work

Haselsteiner [9] discovered that the transmission between the tag and the reader can be modified by an attacker. He pointed out that all the transmitted bits can be modified if Manchester coding with 10% ASK is used whereas, for Miller encoding with 100% ASK, this attack is feasible for certain bits but impossible for others. A strong synchronization is required between the attacker's device and legitimate devices to implement this attack, making it less than practicable. However, the concern here is mostly over reading tags with illegitimate content rather than the subversion of content during transmission.

Madlmayr [5] indicates that the NDEF data is prone to various attacks if proper protection is not used. Roland [10] carried out an analysis regarding signing an NDEF message. He provides the justification for signing a few selected fields of an NDEF message. Roland in [3] exploits some vulnerabilities of the Signature RTD.

Mulliner [4] analyzes the security of NFC-enabled mobile phones. He takes into account not only the NFC-subsystem but also software components that can be controlled through the NFC-interface. The author used the Fuzzing technique to test the NFC software of the Nokia 6131. He found that an NDEF payload length field with values `0xFFFFFFFF` and `0xFFFFFFFE` causes the phone to crash and reset. He also exploits the size of the display screen to launch some attacks on the smart poster.

Verdult and Kooman [11] demonstrate some practical attacks on the Nokia 6212 Classic. It allows users to exchange digital objects easily using the NFC interface. To do so, two phones should be within the proximity coupling distance of about 5 cms. This paper shows that the NFC feature that invokes a Bluetooth connection without user consent can be abused to install malicious software secretly on the phone. This results in a serious vulnerability when smart posters start installing malicious software or spreading viruses.

Francis *et al.* [6] investigate the possibility that a Near Field Communication (NFC) enabled mobile phone, with an embedded secure element (SE), could be used as an attack platform. They showed how an NFC mobile phone can be used for a tag cloning and skimming attack. Their findings indicate that the embedded SE with the existing security controls and the available contactless APIs could be exploited to configure the mobile phone as a contactless attack platform.

6. The Record Composition Attack

The Record Composition Attack is aimed at composing different records in such a way that the digital signature remains valid. There are two scenarios described by M. Roland to accomplish this attack [3].

In the first scenario, two different smart posters are selected in which every record has its own signature. A malicious smart poster record can be created by selecting only a few of the records along with their signatures from the first poster and other records along with their signatures from the second poster. Similarly, unrelated records along with their respective signatures from a number of different posters can be combined together into a single NDEF message. The combined NDEF message will consist of a sequence of records that may be totally meaningless or convey misleading information, but still have valid signatures covering the whole message.

In the second scenario, the Record Composition Attack is accomplished by combining and hiding selected records from different NDEF messages. An adversary takes two or more smart poster records signed by the same or different parties. Each smart poster consists of records of various types like *Text*, *URI* etc. followed by the signature. The attacker takes all records from the posters and combines them to form a new smart poster record. The new poster will have valid signature records corresponding to data from each parent tag. The attacker then effectively removes the unwanted records from the

message by hiding them from the viewer, but keeps the signatures valid.

As all the records are digitally signed, the actual removal of any record invalidates the signature. Instead, the chosen records are retained but hidden from the user by manipulating the unsigned *TNF* field. The *TNF* value is changed from 1 to 5, i.e. from an NFC Forum well-known type to an *unknown* type. The *TNF* value can be changed as it is not signed. The NDEF parser receiving an NDEF record with a *TNF* value of *Unknown* will store the *Payload* of that record without processing it. In this case the *Payload* will not appear to the user. So, rather than removing a record, it has been hidden simply by changing the *TNF* value.

7. The Amended Attack

In fact, Roland's attack [3] described thus far does not necessarily work because there are a few other changes that may have to be carried out in order to keep the signature valid. These necessary modifications were overlooked in [3].

For the new *TNF*=5 of the hidden records, the *Type-Length* field must be zero and there can be no *Type* field (see Table 1). This is not the case for the original record (*TNF* ≠0,6). As the *Type-Length* field is not signed it can indeed be changed to zero, but the *Type* and *ID* fields are digitally signed and omitting or altering these fields to maintain a meaningful payload may invalidate the signature. Specifically, in order for the signature to remain valid, the original signature on the string *Type||ID||Payload* has to be the same as the signature on the new string *ID'||Payload'*. These strings must therefore be identical, with its initial interpretation replaced by one with a different, possibly invalid *ID'* and a new, probably meaningless, message *Payload'*. Quite apart from the semantic issues, the signature verification now fails unless the number of signed bytes is the same, i.e.

(*Type-Length*) + (*ID-Length*) + (*Payload-Length*)
= *(ID-Length') + (Payload-Length')*

Therefore, apart from changing the *TNF* value, some further manipulation of the NDEF header may be required, together with adjustments to the *Type-*, *Payload-* and *ID-* lengths and corresponding removal, addition or repartitioning of bytes in the corresponding three fields.

When the *IL* bit is set, one easy solution is to increment the original value of the *ID-Length* or *Payload-Length* field by *Type-Length*. This corresponds to a reallocation of some of the signed bytes to the *ID* and *Payload* fields. When the *IL* flag is zero this still works providing it is the *Payload-length* field which is incremented by *Type-Length* as mentioned in [10], Section V(L). It works well when the *ID* field is not present, as shown in Figure 4, but in the presence of an *ID* field it results in a new and probably invalid *ID'* field that may be detected by a semantic check. No bytes need removing or adding to the record in these cases.

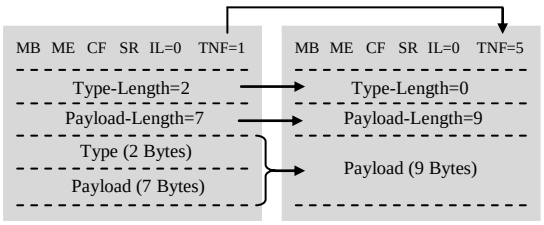

Figure 4. Example changes in the NDEF header when the *ID* field is absent.

We propose another solution, which cannot be caught by a semantic check on either the *Type* field or the *ID* field. Since it has no type, the *Payload* cannot fail a semantic check either. First set the *IL* flag to zero if it is not already zero, and remove the bytes containing the *ID-Length*, as in Figure 5. Then increment *Payload-Length* by (*Type-Length)+(ID-Length*), so that the new *Payload* consists of the concatenation of all the bytes formerly in the *Type*, *ID* and *Payload* fields. In this case, *Payload'* consists of all the signed bytes.

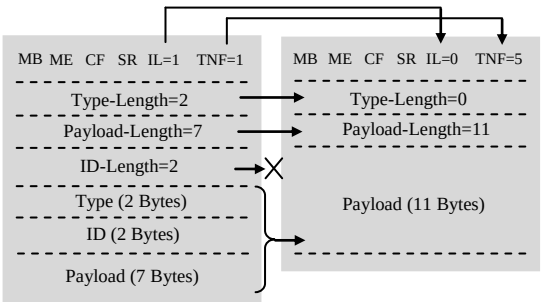

Figure 5. Example changes in the NDEF header when the *ID* field is present.

8. The Record Decomposition Attack

In the second attack described by Roland *et al*. in [3], the payload is split (but *not* chunked) into two parts and spread over two records. The second part is hidden by using a record of *unknown* type, i.e. *TNF*=5. Since *Payload-Length* is an unsigned field, it can be changed in the first record without detection. The signature is computed over the concatenated bytes from the *Type, ID* and *Payload* of all records being signed. So, for the two new records to generate the same signature, it just requires the second record to have no *Type* or *ID* fields. The unknown type does this job as a record with an *unknown* type has no *Type* or *ID* fields. The only thing required to accomplish this attack is the suitable completion of the NDEF header fields of the new unknown-type record.

An example of such an attack is the text of a smart poster stating, *"Do not board the train until you have a valid ticket"*. This text is digitally signed and the signature is stored using the Signature RTD. An attacker may split this message into two separate records as above. The first record stating *"Do not board the train"* will be visible to the user, whereas the second record stating *"until you have a valid ticket"* will not appear to the user as it is sent with the NDEF header fields stating *unknown* type. However, the digital signature will remain valid and so the user will consider it as a valid message. This attack of Roland works in its original form without further modification of length fields such as those described in the previous attack.

9. Countermeasures

Roland proposed that the receiver should only trust the payload bytes from a sequence of records if they are signed and share a common signature record [3]. But this needs very careful interpretation. As shown in the example of the Record Decomposition Attack in §8, the records share a common signature but only part of the message payload is displayed to the user. This partially displayed message with a valid signature cannot be trusted. Hence, the user's *view* of the message cannot be trusted even if all its records share a common signature.

The easiest way to avoid these attacks would be to sign all the header fields so that they may not be altered, but in practice this is out of the question. For example, the *MB* flag is intentionally unsigned so that a group of signed NDEF records may be moved to any position within an NDEF message [10]. This enables a variety of messages to be constructed for different target viewers around an important core content which is signed. However, it is unnecessary to sign the *ME* flag as the end of the section of the message which needs to have its integrity secured is marked by the signature record, so all the records preceding the signature record will have $ME = 0$.

A principle requirement of the signature definition is to be able to partition an NDEF record into multiple record chunks or *vice versa* as shown in Figure 6 without affecting the validity of the signature. This means, in particular, that only the *Payload* can be included in the signature for records after the first chunk, and that the chunk flag *CF* must be omitted from any header data that is included in the signature. The inclusion of any other field from the non-initial chunks, such as *length* fields, *TNF* or *CF* in the signature would also invalidate the signature. The fields from the *initial* chunk which are independent of whether or not the record is chunked are the only ones which could be included in the signature. They are the *MB, IL* and *TNF* fields in the header byte, the *Type-Length* and *Type* fields, and the *ID-Length* and *ID* fields.

However, one could sum the payload lengths from each chunk to obtain the same payload length as in the unchunked record, and include that in the signature because it is unaffected by chunking. This needs to be done with care as it should be possible to compute the signature using a block by block hash function without having to store every chunk payload. The message digest and total payload length must therefore be computed in parallel and, once the last chunk has been read, the length appended as a suffix to the string to being hashed. The resulting MAC is then signed and, if necessary, validated.

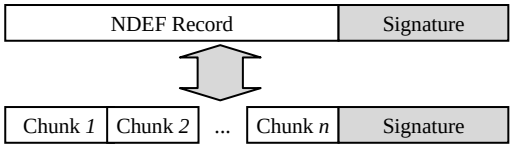

Figure 6. An NDEF record appended with its digital signature is partitioned into multiple chunk records. The signature is valid for both cases.

Another principle which we may wish to respect in proposing any revision of specifications is to insist that the signature is computed on the same components of each record irrespective of chunking. This would slightly simplify validation code, as would omitting the payload length computation.

The last principle worth mentioning is the desire to compute signatures directly from the concatenated record bytes in the order they appear and without alteration. It is easy to observe that the attacks above would not work if, for example, an extra byte *B* of fixed value were inserted between the *Type, ID* and *Payload* strings when necessary to separate them before the signature is computed. Thus, when none of the three components were the empty string, this would mean computing the signature of the string *Type||B||ID||B||Payload*, but it would be computed on just *Payload* when both *Type* and *ID* were of length 0. If this were done, the chunking process would not disturb the calculation of a signature, but the re-partitioning of bytes required in the Record Composition attack would not work. This particular solution becomes unnecessary if the lengths of the various data components are also signed.

9.1. Revision of the Signature Specification

We propose some modifications to the signature specification. The signature is presently computed over *Type*, *ID* and *Payload* fields as recalled in Section 4. Because of the two attacks, the inclusion of additional fields is necessary. However, in order to preserve the validity of signatures when a record is chunked, a different signature process is required for non-initial chunks. The proposed modified signature is compared to the existing specification in Table 3.

Table 3. Comparison of the Existing and the Proposed Signature Record Specifications

Field Name	Existing	Proposed
MB	Not signed	
ME	Not signed	
CF	Not signed	
SR	Not signed	
IL	Not signed	
TNF	Not signed	**Signed** (unless *TNF*=6)
Type-Length	Not signed	**Signed** (unless *TNF*=6)
Payload-Length	Not signed	**Signed** (in modified form)
ID-Length	Not signed	**Signed** (if present)
Type	Signed	
ID	Signed	
Payload	Signed	

The first byte of the NDEF header containing *MB, ME, CF, SR, IL, TNF* cannot be added in full to the signature as noted earlier. However, making the *TNF* value immutable is wise. Its updating was the source of problems in the Record Composition attack. So, part of our proposal is to sign this for all records except the non-initial chunks. For $TNF \neq 6$, create a byte *TNFB* by masking the non-*TNF* bits from the first byte of the record. This byte will be signed. *TNFB* will be the empty string for $TNF = 6$, and so not alter the signature when a record is chunked.

Next, we propose adding the *Type-Length* and *ID-Length* fields to the existing fields for signing except in the case of non-initial chunks, i.e. records with $TNF = 6$, when they are to be omitted. Addition of these two fields to the signature process does not invalidate the signature under the chunking process.

As noted before, *Payload-Length* cannot be signed unless the length is accumulated over all chunks. Let *Total-Payload-Length* denote the sum of *Payload-Length*s over all chunks of a chunked record, and the normal *Payload-Length* for an unchunked record. For convenience, let us define *Total-Payload* similarly: it is the usual *Payload* for an unchunked record and the concatenation of the *Payload*s from all chunks of a chunked record. This means that *Total-Payload-Length* and *Total-Payload* are simply the *Payload-Length* and *Payload* of the corresponding unchunked record.

In our revised signature specification, the contribution to the signature of all chunks from a chunked record is the following string:

TNFB||*Type-Length*||*ID-Length*||*Type*||*ID*||
Total-Payload||*Total-Payload-Length*.

The *TNFB, Type* and *ID* contributions and their lengths are, of course, those given in the *initial* chunk. The contribution of an unchunked record is the same, but can be written more simply as following because it is a single record.

TNFB||*Type-Length*||*ID-Length*||*Type*||*ID*||
Payload||*Payload-Length*.

Because of our definitions of *Total-Payload* and *Total-Payload-Length*, the contributions to the signature are the same for an unchunked record and a chunked version of the same record. This means that the new signature could be simply defined in terms of the concatenated contributions from records in the equivalent unchunked message.

Note, finally, the ambiguity in the string used for *Total-Payload-Length*. This could be up to four bytes, and we do not know if the original unchunked record used one or four bytes if this length is under 2^8. A formal specification would have to determine how it should be given, e.g. the endianness at bit and byte level using four bytes or using the minimum number of bytes.

9.2. Suitability for the Record Chunking Process

The proposed signature scheme can be successfully used for validating messages with many record chunks without the need to store payload data. The first half of the contribution from a chunked record, namely

TNFB||*Type-Length*||*ID-Length*||*Type*||*ID*

is wholly derived from the initial chunk. Thereafter the string for hashing is given by appending the chunk *Payload*s until *Total-Payload* has been appended. At the same time, a record is kept of the sum of the *Payload-Length*s of the chunks. When the last chunk has been received, this sum equals the required *Total-Payload-Length*, and so its value can be appended also.

Therefore, a record may be partitioned into multiple chunks or *vice versa* without affecting the validity of the signature or the ease with which the signature is computed.

10. Security Analysis of the Proposed Specifications

The modifications need to be analyzed for the feasibility of an attack and to justify the inclusion of the various new fields.

10.1. Counter to the Record Composition Attack

The main reason for the success of the Record Composition Attack was the unsigned NDEF header fields that could be manipulated in a specific way to accomplish this attack (see Figures 4 and 5). In our proposed structure, the *TNF*, *Type-Length*, *Payload-Length* and *ID-Length* fields are signed in addition to the already signed *Type*, *ID*, and *Payload* fields. So these fields can no longer be manipulated in the required way. This makes Record Composition Attack impossible.

Specifically, in the terminology of section 7, for the same attack to be successful under the new signature scheme would require at least *Type-Length* = *Type-Length'* = 0 and *ID-Length* = *ID-Length'* as these both fields are signed and cannot be changed. However, *Type-Length* = 0 cannot occur when *TNF* is other than 0, 5 or 6. Although the original attack had an initial *TNF*=1 being changed to *TNF*=5, we should consider the possibility of attacks with these other initial values in order to justify (or not) the inclusion of *TNFB* in the proposed signature.

For *TNF*=0 the record should be empty. In this case the *Payload* is the empty string and making it invisible will not make any difference to what the user reads. For *TNF*=5, an update to *TNF*=5 also makes no difference to the message. Finally, *TNF*=6 indicates that the record is a non-initial chunk. Changing the field to have the value 5 would change it to a non-chunked record and result in the inclusion of additional *Type-Length* and *ID-Length* fields. Although the *ID-Length* field is optional in a record, the *Type-Length* field is not. It contributes 1 byte in the new signature scheme, resulting in a different signature if *TNF* is changed from 6 to 5. We conclude that, whatever the initial value of *TNF*, updating it to 5 invalidates the signature under the proposed scheme just by virtue of including the *Type-Length* and *ID-Length* fields, no matter how the other fields are changed.

Of course, the user needs to be aware of where signed messages start and finish since any signed messages might be combined without change into a larger misleading or wrong message. The signature specification clearly defines the starting and finishing point of the data to be signed. It is up to the user's browser and security policy to make clear where signed messages begin and end. Ideally, it should show a single signed message at a time and indicate that the visible message is signed with nothing hidden.

10.2. Counter to the Record Decomposition Attack

In this attack, a record payload is decomposed into multiple parts which are completed to full (unchunked) records by the addition of relevant header fields. The *TNF* value for some parts is set to 5, making them inactive records. The header fields also contain a *Type-Length* field with value zero. As this one-byte field is digitally signed in the proposed scheme, it will contribute to the string on which the signature is computed and result in an invalid signature. The only way to avoid this byte being part of the signature is to make the second record into a chunk. However, this requires *TNF*=6 and so prevents the value *TNF*=5 which is needed to hide the record's payload in the attack. Thus, the Record Decomposition Attack is successfully countered in the proposed scheme.

The specification for the unknown type record with *TNF*=5 has some redundant data. The *Type-Length* field is always zero and therefore redundant. This redundancy, in contrast to the record chunking case, proves to be a mechanism preventing the Record Decomposition Attack. If it were removed, the heading requirement for the hidden parts of the payload in the Record Decomposition Attack would be the first NDEF header byte and the *Payload-Length* field. If none of this information were included in the revised signature specification, data from the header fields would not invalidate the signature. Therefore excluding this redundancy would make the Record Decomposition Attack feasible for the revised signature specification if it excluded *TNFB* and *Total-Payload*.

In conclusion, although the *Type-Length* field is redundant in an unknown type record, it helps prevent the Record Decomposing Attack. Nevertheless, other fields in the proposed signature specification ensure that this redundancy could be safely removed.

10.3. Other Combinations of Records

This section discusses other potentially malicious combinations of records with respect to the proposed signature specification. §§10.1 and 10.2 considered all possibilities for hiding part of a signed payload. One can ask if there are other changes to a sequence records which would not affect the signature. The first such combination is to sign the last of the *middle* and *terminating* chunks (and perhaps more subsequent records) while omitting the *initial* and the first few *middle* chunks. Although the signature specification only covers the complete sequence of chunks, it could be abused, with the first chunk to be signed being treated as the initial chunk, contributing its values for the *ID* and *Type*, among other things. This combination might have its *Type* and *ID* properties changed since they are inherited from the initial chunk which may not have been signed. However, such a change is not allowable according to the NDEF specification [1]. This is because the first record in a sequence of signed records must be

the first record of the message or be preceded by a signature record. As a signature record cannot be a chunk (it has *TNF*=1, not 6), the start of the signed sequence of records must be before the initial chunk. Consequently, the *Type* and *ID* properties and their lengths are always signed for the part of the payload which is signed.

Let us now consider manipulation of the unsigned bits in the first header byte. We can ignore the *MB* and *ME* flags as they do not affect the semantic content of the records.

Any alteration to the *SR* bit changes the location of the other signed bytes, such as the *Payload*. This leads to an invalid signature unless there is a corresponding addition or removal of three *Payload-Length* bytes. If this changes the value of *Payload-Length* then the signature will be incorrect as that value is signed. If that value is unchanged then the *Payload* is unchanged, so that the interpretation of the record is unchanged. Hence if the *SR* bit can be changed without invalidating the signature then the message content is unchanged.

Switching the *IL* bit without invalidating the signature is not possible except for non-initial chunks where the value is irrelevant. Moreover, the *IL* flag is always zero for non-initial chunks as defined in the NDEF specification [1]. Changing *IL* introduces a byte for *ID-Length* into the signature stream or removes it, thereby altering the signature.

Finally, we briefly comment on the need to include the *TNF* value in the signature. For example, without *TNFB*, any of the values 1, 2, 3, 4 might be inter-changed without change to the sequence of bytes being signed. This would lead to a different interpretation of the *Type* value and hence a different interpretation of the *Payload*. We would then have to rely on the parser flagging an inconsistency. It is quite possible, although unlikely, for the differently interpreted payload to convey incorrect information to the user. Thus it is still wise to include *TNFB* in the signature scheme.

11. Implementation Issues

11.1. Modification to the Signature Specification

The proposed signature scheme is different from the existing one because of the addition of, *inter alia*, *Type-Length* and *ID-Length* fields. As such, it must be assigned a different version number in order to maintain backward compatibility. Fortunately, in this specification there is a version number which can be incremented.

The payload of the signature RTD consists of three fields as noted in §4, Fig 2: *Version*, *Signature* and *Certificate Chain*. These three fields are transmitted in the same order, with *Version* in first place. An NDEF parser can determine the signature specification by first analyzing the version number. This is a single byte field so it can handle up to 256 versions of signature specification. As the existing signature specification is the only version presently available, the only currently valid version number is 1. So our proposed signature specification should be given version number 2.

The proposed specification is not compatible with version 1 because of the extra signed fields. Hence signature validators will have to be upgraded to enable version 2 signatures to be checked.

11.2. Modifying the NDEF Specification

Implementing changes in the NDEF specification is tricky as there is no *Version* field available in the NDEF format. This could be implemented by adding a single byte version number prefix with most significant bit 0 to distinguish it from the initial byte of the first record which has *MB* flag set to 1. Alternatively a complete version record could be constructed with additional details for parsing the information efficiently. However, it seems sensible to avoid changing both the NDEF and Signature RTD specifications when it is only necessary to change one of them.

11.3. The User Interface

A digitally signed NDEF message should display some information for the user at the application level. It may include the name of the signing authority (from an x.509 certificate) with some additional details for the assurance of the user.

Our consideration of security issues showed that it is still important for users to be informed whether or not messages have been signed and for their browsers to make clear where individual signed messages begin and end. It should not be possible for signed messages to be concatenated without the user being aware where one message has finished and the next has started.

A signature can potentially be removed from a tag without any indication to the user (such as in a duplicate tag). It is up to the user whether he trusts the contents of a message without a signature or not. However, it is clear from the example attacks that the browser should be pro-active in warning the user of potential dangers, including the lack of any signature.

12. Conclusion

The Record Composition and Decomposition Attacks exploit unsigned fields in the NDEF header. Previously proposed attacks were not fully implementable without further modifications to these header fields. We refined those attacks and explained precisely what additional changes need to be made to

exploit the unsigned fields. Such attacks can be countered if the length fields of the NDEF header are also signed. We proposed a solution that requires modification to the Signature RTD in which, amongst others, the *TNF*, *Type-Length*, *Payload-Length* and *ID-Length* fields are included. We presented a security analysis of the proposed scheme, and verified that it was no longer possible to exploit the NDEF header in attacks of the type discussed, thus successfully countering Record Composition and Decomposition Attacks in particular. Inclusion of the *TNF* field accounted for some remaining semantic issues. Because of their impracticality, we discarded alternative solutions involving updating the NDEF specification.

13. References

[1] NFC Forum, "NFC Data Exchange Format (NDEF): Technical Specification", *http://www.nfc-forum.org/specs/spec_list/*, July 2006.

[2] NFC Forum, "Signature Record Type Definition: Technical Specification", *http://www.nfc-forum.org/specs/spec_list/*, November 2010.

[3] Michael Roland and Josef Langer and Josef Scharinger, "Security Vulnerabilities of the NDEF Signature Record Type", *Third International Workshop on Near Field Communication*, Hagenberg, Austria, February 22-23, 2011. IEEE Computer Society, 2011, pp. 65–70.

[4] C. Mulliner, "Vulnerability Analysis and Attacks on NFC-Enabled Mobile Phones", *The Forth International Conference on Availability, Reliability and Security*, Fukuoka, Japan, March 16-19, 2009. IEEE Computer Society, 2009, pp. 695–700.

[5] G. Madlmayr, J. Langer, C. Kantner, and J. Scharinger, "NFC Devices: Security and Privacy", *Third International Conference on Availability, Reliability and Security*, Technical University of Catalonia, Barcelona, Spain, March 4-7, 2008. IEEE Computer Society, 2008, pp. 642–647.

[6] L. Francis, G. Hancke, K. Mayes, and K. Markantonakis, "On the security issues of NFC enabled mobile phones", *International Journal of Internet Technology and Secured Transactions*, vol. 2, *Number 3-4, 2010*. Inderscience Enterprises Ltd, 2010, pp. 336–356.

[7] NFC Forum, "NFC Forum Home Page", *http://www.nfcforum.org/home/*, 2004.

[8] NFC Forum, "Smart Poster Record Type Definition: Technical Specification", *http://www.nfc-forum.org/specs/spec_list/*, July 2006.

[9] E. Haselsteiner and K. Breitfuß, "Security in Near Field communication (NFC): Strengths and Weaknesses", *Workshop on RFID Security (RFIDSec)*, Graz, Austria, July 12-14, 2006.

[10] M. Roland and J. Langer, "Digital Signature Records for the NFC Data Exchange Format", *Second International Workshop on Near Field Communication*, Monaco, April 20-22, 2010. IEEE Computer Society, 2010, pp. 71–76.

[11] R. Verdult and F. Kooman, "Practical attacks on NFC enabled cell phones", *Third International Workshop on Near Field Communication*, Hagenberg, Austria, February 22-23, 2011. IEEE Computer Society, 2011, pp. 77–82.

[12] M. Q. Saeed and C. D. Walter, " A Record Composition/Decomposition Attack on the NDEF Signature Record Type Definition, *6th International Conference on Internet Technology and Secured Transactions*, Abu Dhabi, UAE, December 11-14, 2011. IEEE Computer Society, 2011, pp 284-287.