Formal Security Analysis and Improvement of a Hash-Based NFC M-Coupon Protocol

Ali Alshehri(✉) and Steve Schneider

Department of Computing, University of Surrey, Guildford GU2 7XH, UK
a.a.alshehri@surrey.ac.uk

Abstract. *Near field communication* (NFC) is a Radio Frequency (RF) technology that allows data to be exchanged between devices that are in close proximity. We formally analyse a hash based NFC mobile coupon protocol using formal methods (*Casper/FDR2*). We discover a few possible attacks which break the requirements of the protocol. We propose solutions to address these attacks based on two different threat models. In addition, we illustrate the modelling from the perspective of the underlying theory perspective, which is beyond the knowledge required for modelling using CasperFDR tool (black-box approach). Therefore, this paper is a facilitating case study for a "black-box" CasperFDR user to become a more powerful analyser.

Keywords: NFC · M-coupon · CasperFDR · Formal verification · Protocol security

1 Introduction

Near Field Communication (NFC) [1] is a radio frequency (RF) communication link, which allows data to be exchanged between devices that are normally less than 10 cm apart [2]. NFC-based mobiles are an emerging technology changing the way we communicate with objects. For instance, payments, tickets and coupons can be exchanged just by waving the NFC-based mobile at the points of sale.

NFC security is an important issue that has been emphasised in the literature [3,4]. Even though NFC has the advantage of a short communication link, security measures must be considered especially with sensitive applications to address security requirements, such as confidentiality, integrity and availability.

The NFC mobile coupon application (M-coupon) is one of the promising and popular applications [5–8]. An M-coupon is a cryptographically secured electronic message with some value. It requires secure issuing and cashing of the M-coupons, otherwise it can cause huge loss and reputation damage for a company [9].

The NFC M-coupon system has a typical scenario, see Fig. 1. All parties have NFC capability, in order to communicate with each other. Firstly, a user scans his NFC mobile against an NFC issuer (e.g., a smart poster or newspaper), and

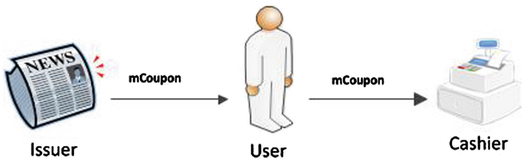


Fig. 1. General NFC mobile coupon

an M-coupon is issued and sent to his mobile. Later, the user goes to the shop to cash the M-coupon with the cashier. The cashier may authenticate the user before the cashier provides the promised bonus. Only the cashier needs to have online access, whereas the issuer and the user can both be offline. Hsiang et al. [10] have proposed a secure hash-based M-coupon protocol which allows secure issuing and cashing of M-coupons. They designed the protocol to address specific M-coupon requirements.

On the other hand, designing a security protocol is a difficult task even with strong encryption methods. Many attacks may be possible on the cryptographic protocols just by intercepting and replaying encrypted messages between entities, without decrypting any messages. Formal security analysis is a powerful approach to check the security of a protocol and whether it address its requirements [11].

In this paper we use the CasperFDR approach [12] based on *Communicating Sequential Processes* (CSP) [13], a formal method (state exploration) approach, to formally analyse the NFC M-coupon protocol proposed by Hsiang et al. [10]. Our analysis found attacks against the protocol. We then provide three solutions to address these vulnerabilities, and formally verify them with CasperFDR.

In addition, we illustrate the modelling from the underline theory angle. Modeling protocols in CasperFDR requires only an abstract description of the protocol and required security requirements to be checked. Then, CasperFDR provides the result detailing whether an attack was found or not. We call this a black-box approach as the underlying models are not shown to the user. In this paper we consider this point that we illustrate the modeling from the underlying theory perspective (CSP aspect). This is important to enable a black-box user to become more powerful in protocol analysis using CasperFDR and model the protocol and its requirements in a precise approach.

2 The Casper Approach

In our analysis we use CSP [13], with its model checker *Failures Divergence Refinement* (FDR2), which is proven to be an effective method in analysing the security of protocols [14]. However, modelling protocols in CSP is not a trivial task. Gavin Lowe developed *CasperFDR* [12], a tool that allows the user to write an abstract description of a security protocol, then the tool produces a model in the CSP language, and directly checks it with FDR2. CasperFDR has been used to analyse a huge number of protocols [15], which proves its capability of finding vulnerabilities.

CasperFDR is a formal method tool which supports symbolic protocol analysis in the Dolev-Yao model [16] which assumes that no encrypted message can be decrypted without the decryption key, thus the CasperFDR intruder model does not perform any cryptanalysis. However, the intruder does have full control of the network traffic, and tries to break the security protocol from what passes on the network.

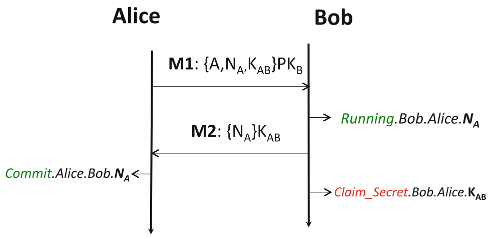
CasperFDR performs a refinement check of the protocol against its requirements. When refinement fails, then it provides a trace which shows how the property fails, that corresponds to an attack. Moreover, CasperFDR manages the Xor operation where attacks against these algebraic properties are considered in CasperFDR.

2.1 Simple Example

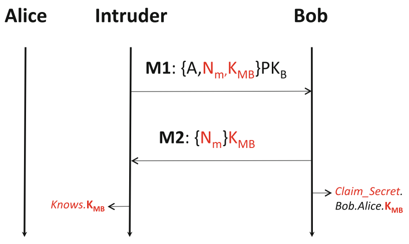
Figure 2a is our demonstrating simple protocol. A two message protocol aims to authenticate Bob to Alice:

1. Alice $\rightarrow$ Bob : $\{A, N_A, K_{AB}\}PK_B$
2. Bob $\rightarrow$ Alice : $\{N_A\}K_{AB}$

Message 1 sent by Alice to Bob contains Alice identity, Nonce (number used once) and a session key K_{AB}, encrypted with Bob's public key. Then, Bob sends message 2 by encrypting the Nonce (N_A) with the session key. Alice authenticates



(a) Simple protocol



(b) Simple protocol attack

Fig. 2. Illustrating example

Bob based on the fact that since message 1 is encrypted by Bob's public key, he is the only one that can extract the session key and the nonce and send message 2. At the end of the protocol both Alice and Bob believe the session key K_{AB} is secret.

Analysing this protocol in CasperFDR is a straightforward procedure. Having modelled the exchanged messages between entities, we check the claimed authentication and secrecy using the following claims:

```
Agreement(Bob,Alice,[NA])
StrongSecret(Bob, KAB,[Alice])
```

The *Agreement* specification means it will check whether Bob is authenticated to Alice and have both of them agreed on the Nonce N_A. The *StrongSecret* specification is checking whether the key K_{AB} is secret between Bob and Alice. CasperFDR will complete the remaining process for us as we explained earlier.

Nevertheless, understanding how these specifications are captured underneath CasperFDR is important if we want precise descriptions of how claimed properties are modelled in a specific application, as we will see later.

Capturing *authentication* between Alice and Bob in the protocol is done by utilising new events injected in the protocol as demonstrated in Fig. 2a. These events are *Running* and *Commit*. Initially, Alice and Bob are modelled as independent CSP processes. After message 1, Bob performs the *Running* event, which means Bob starts running the protocol apparently with Alice. Then, Alice will perform the *Commit* event at the end of her part of the protocol, which means Alice has finished a run of the protocol with Bob. Alice could make sure she was running the protocol with Bob based on the fact that if Alice reaches the *Commit* event then Bob must have reached the *Running* event before. Launching an attack relies on the possibility of the intruder, without taking Bob's role, to engineer a trace of the protocol in which Alice runs the *Commit* event without a corresponding *Running* event from Bob.

For *secrecy*, only the *Claim_Secret* event is used by Alice and Bob. Figure 2a only shows when Bob performs *Claim_Secret* event. An attack is launched if the intruder could break this claim, by finding a trace of the protocol in which the intruder knows a claimed secret, without taking Alice's or Bob's roles.

The *Running*, *Commit* and *Claim_Secret* events can also contain more information specific to the agreement required between the participants. They are constructed by:

$$Agent.Agent.Message$$

For example *Running.Bob.Alice.*N_A, which means Bob starts a run of the protocol, apparently with Alice, using nonce N_A.

The *Casper* analysis finds no attack on authentication, but there is an attack on secrecy. Figure 2b illustrates how an intruder can create a session key that Bob believes is secret with Alice. Anyone can generate message 1 since Bob's public key is publicly known. The intruder impersonates Alice by including Alice's identity. At the end of the protocol run Bob believes the session key K_{MB} is a secret shared with Alice. However, it is known by the intruder.

2.2 Hierarchy of Authentication and Secrecy

CasperFDR provides different flavours of testing authentication and secrecy. The strongest form of authentication specification is *Agreement*. If Alice and Bob meet the *Agreement* specification, then if Bob thinks he has successfully completed a run of the protocol with Alice, then Alice has previously been running the protocol, agreeing on their roles in the protocol, and there is a one-to-one relationship between Alice and Bob i.e. each run of Alice corresponds to a unique run of Bob.

A weaker authentication specification is *NonInjectiveAgreement*. The difference from *Agreement* is that the one-to-one relationship is not required. Each run of one participant matches a run of the other but they can overlap. For example, two "Commit" events may correspond to the same "Running" event.

Secrecy has two forms of specification, *Secret* and *StrongSecret*. *Secret* tests if the intruder could know the secret value at the end of the protocol. *StrongSecret* is stronger than *Secret* in that, including the *Secret* specification, it even checks whether the intruder is able to know the secret value without completing a full run of the protocol.

2.3 Channels

The CasperFDR intruder cannot open encrypted message without the decryption key, but also CasperFDR allows more restriction on the intruder's ability on any messages of the protocol. For example:

```
#Channels
1 NF NRA- NR
2 C NF NRA NR
```

The first line means that on message 1 the intruder neither can fake data *NF* (No Fake), nor honest reascribing *NRA-* (changing the sender ID except to his own ID) nor redirecting *NR* (changing the receiver ID). The second line means that on message 2 the intruder neither can eavesdrop *C*, nor fake data, nor reascribing nor redirecting.

By adjusting some of protocol's channels, we can capture assumptions made in the protocol as we will see later.

We do not restrict the ability of the intruder with respect to the wireless aspect. Even though eavesdropping is still a major threat, the intruder would not have that ability of communicating with the participants at the same time as in the normal wired network e.g. the Internet. In this paper we model the protocols in the Dolev-Yao model. If an attack occurred, we then analyse informally the feasibility in a wireless context. If no attack is found, then it means there should not be any attack in a weaker wireless model.

3 Protocol Security Requirements

The analysed protocol we consider in this paper intends to meet six security requirements, as stated by the protocol designer in [10]:

- **Confidentiality:** A third party should not be able to obtain the M-coupon by eavesdropping.
- **Data Integrity:** An attacker should not be able to modify data during the communication.
- **Forgery Protection:**
  - **No Unauthorized Generation:** An attacker should not be able to issue his own M-coupon.
  - **No Manipulation:** M-coupon should not stay valid after a manipulation.
- **Unauthorized Copying:** An attacker should not be able to produce a valid copy of an M-coupon and cash it in. This requirement can be divided into:
  - **Not Transferable:** Whatever identity is presented at issuing phase should not be changed during the protocol.
  - **User Authentication:** In addition to Not Transferable, the identity of the user is the one who it claims to be. The user who issued the M-coupon must be the one who is cashing it at the cashier. This requires the cashier to authenticate the user through some authentication methods.

  This protocol only addresses the Not Transferable requirement.
- **No Multiple Cash-in:** An attacker should not be able to use the same M-coupon multiple times.

3.1 Formal Definition

Figure 3 illustrates a formal definition in CasperFDR of these requirements and the relationship between them. The formal definitions can apply to a variety of M-coupon protocols [17].

Confidentiality requires that data representing the M-coupon in the protocol must satisfy *StrongSecret* specification between the issuer and the cashier.

NonInjectiveAgreement specification includes three requirements, and layers, of authentication between the cashier and the issuer. We are not concerned here with repeats because that is checked directly by other means, and it may be not required in some systems. For Forgery Protection, after identifying data representing the M-coupon, a *NonInjectiveAgreement* on the M-coupon between the issuer and the cashier is required. This is violated if the cashier accepts an M-coupon not been issued by the issuer. This implies either that the M-coupon has been created by an attacker (i.e. Unauthorised Generation) or else that an M-coupon generated by the issuer has been modified to another (i.e. No Manipulation). Not Transferable is a stronger specification than forgery protection that it also requires an agreement on a user identity attached to the M-coupon. The strongest *NonInjectiveAgreement* specification is Data Integrity: both the cashier and the issuer must agree on all data in the protocol.

No Multiple Cash-in requires an *Agreement* specification between the issuer and the cashier. Every time the cashier accepts an M-coupon, there must be a separate occasion where the issuer must have issued it. Hence the cashier cannot accept an M-coupon more times than it was issued.

User Authentication is an *Agreement* between the user and the cashier on some credential. Even though the M-coupon might be used many times, the user must be authenticated each time.

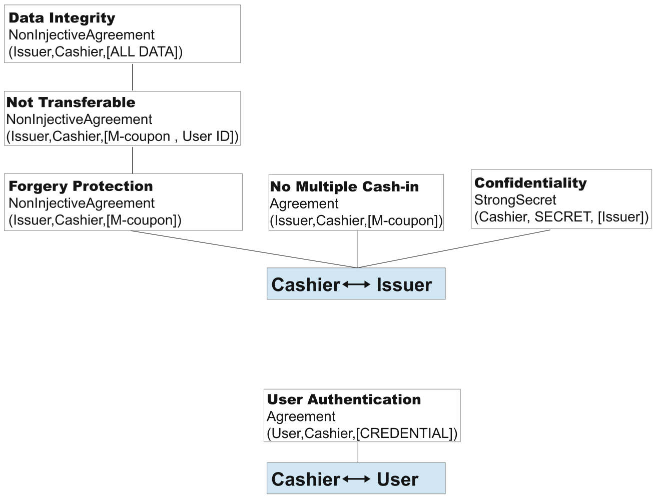


Fig. 3. Hierarchy of authentication/secrecy properties

4 Protocol Description

The M-coupon protocol of [10] uses simple hash functions, which is a computationally light cryptographic method, a suitable choice with standard RFID/NFC tags. Figure 4 shows the messages in protocol notation.

There are four messages in this protocol. Initially, the cashier C and the issuer I share a secret value, X and an offer. The cashier stores a table consisting of hashes of all issuers identities, $h[ID(i)]$.

1. $U \rightarrow I \ : ID(u)$
2. $I \rightarrow U \ : M = ID(u)\ ,\ V\ ,\ C$
3. $U \rightarrow C : M = ID(u)\ ,\ V\ ,\ C$
4. $C \rightarrow U : BONUS$

$ID(i)$	*Issuer ID*
$ID(u)$	*User ID*
Offer	*Data about the Offer*
X	*A secret key shared between the issuer and the cashier*
$\oplus$	*Exclusive or (XOR)*
$h[..]$	*Hash function*
V	$ID(u) \oplus h[ID(i)]$
C	$h[h[ID(i)] \oplus X \oplus Offer]$

Fig. 4. The hash-based M-coupon protocol

At the issuing phase, the user's mobile U sends its identity to the issuer (message 1). Then, the issuer produces a coupon, M (message 2), consisting of three parts: $ID(u)$, $V(= ID(u) \oplus h[ID(i)])$ and $C(= h[h[ID(i)] \oplus X \oplus Offer])$.

At the cashing phase, the user's mobile sends the M-coupon to the cashier (message 3). Then, the cashier obtains $h[ID(i)]$ by computing $ID(u) \oplus V$. The cashier can look up ID(i) from the hash, and then find X and *Offer*. When the cashier has $ID(u)$ and *Offer*, it can decide if the M-coupon has been used before, and reject it accordingly. Then, the cashier will check the validity of the M-coupon by computing $h[h[ID(i)] \oplus X \oplus Offer]$ and confirm it matches C. The cashier stores $ID(u)$ to prevent re-use of the coupon, and sends the Bonus to the user.

The intention is that the Confidentiality and Data Integrity requirements are ensured by use of the secret value X. By including the identity of the user's mobile and offer, Multiple Cash-in and Not Transferable can be managed. Forgery Protection is addressed by the secret value X which is known only by the cashier and the issuer.

5 Modelling

In order to capture the NFC M-coupon requirements in the protocol, we develop a model as shown in Fig. 5. The model captures the following requirements: Confidentiality, Forgery Protection, Not Transferable, Data Integrity and No Multiple Cash-in. Moreover, data representing the M-coupon in the protocol are X and *Offer*.

5.1 The Protocol's Requirements

We show in the following sections what we have to write in CasperFDR to model the protocol's requirements of Sect. 3, then how it is captured from the underneath CSP theory aspect, in terms of *Running*, *Commit* and *Claim_Secret* events. This enabling us to have formal and precise descriptions of capturing NFC mobile coupon requirements.

Confidentiality. We model confidentiality in Casper as follows:

```
StrongSecret(C, X , [I])
StrongSecret(C, Offer , [I])
```

These secrecy specifications means the cashier C claims that X and *Offer* are confidential between the cashier C and the issuer *I*. *StrongSecret* checks whether the intruder is able to break these claims without completing the protocol.

Figure 5 illustrates the first specification, secrecy of X. When the cashier C performs the *Claim_Secret.C.I.X* event, it can expect X to be a secret with the issuer I who shares the secret key *X*. If this is violated then the intruder can complete a run of the protocol with the cashier without taking the issuer role in the protocol, and learn the secret key *X*. A similar description for the *Offer* specification.

Forgery Protection. We model forgery protection in Casper as follows:

```
NonInjectiveAgreement(I,C,[X,Offer])
```

We identify the M-coupon with *X* and *Offer*. This states that if cashier accepts *X* and *Offer*, then the issuer must have issued them. *NonInjective* means that it is not concerned with repeats. I.e. the cashier can accept many times what was issued once. This is violated if the cashier accepts *X* and *Offer* that have not been issued by the issuer. This implies either that *X* and *Offer* have been created by an attacker (i.e. Unauthorised Generation) or else that an M-coupon generated by the issuer has been modified to another (i.e. No Manipulation). Hence if this property holds then we have Forgery Protection: No Unauthorised Generation and No Manipulation.

This is illustrated in Fig. 5. After the issuer completes its part of the protocol, it performs the *Running.I.C.X.Offer* event, which means the issuer *I* starts a running of the protocol, apparently, with the cashier *C*, agreeing on *X* and *Offer*. Later, the cashier will perform the *Commit.C.I.X.Offer* event at the end of its part of the protocol, which means the cashier *C* has finished the protocol with the issuer *I*, agreeing on the *X* and *Offer*.

Unauthorized Copying (Not Transferable). We model Not Transferable in Casper as follows:

```
NonInjectiveAgreement(I,C,[X,Offer,ID(u)])
```

This specification is similar to forgery protection specification, but also with an agreement on a user identity. The coupon, [*X*, *Offer*], must be attached to one user only *ID(u)*. Both the issuer and the cashier agree on the user to use the coupon as many times as he like, as long as the coupon has been issued by a genuine issuer, and is being used by the intended user. An example for such coupon is a frequent flyer coupon.

This is shown in Fig. 5. After the issuer completes its part of the protocol, it performs the *Running.I.C.X.Offer.ID(u)* event, which means the issuer *I* starts a running of the protocol, apparently with the cashier *C*, agreeing on *X*, *Offer* and *ID(u)*. Later, the cashier will perform the *Commit.C.I.X.Offer.ID(u)* event at the end of its part of the protocol, which means the cashier *C* has finished the protocol with the issuer *I*, agreeing on the *X*, *Offer* and *ID(u)*.

Observe that this property is stronger than Forgery Protection. If it holds then not only the m-coupon must be genuine, as for Forgery Protection, but it must also have the same user.

Data Integrity. We model Data Integrity in Casper as follows:

```
NonInjectiveAgreement(I,C,[X,Offer,ID(u),ID(i)])
```

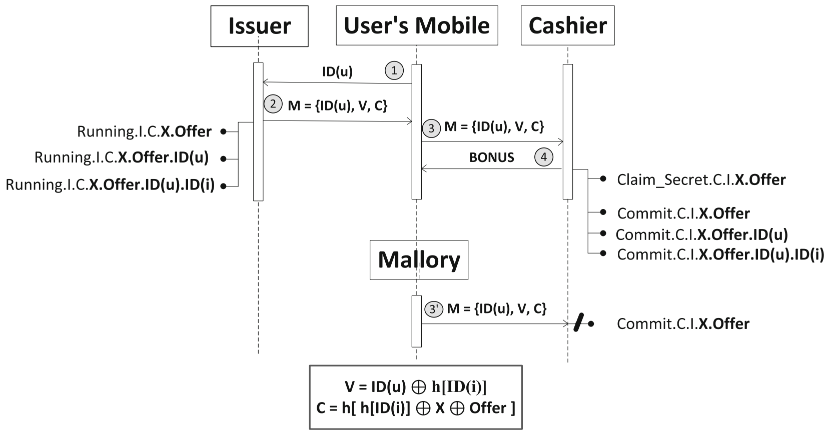


Fig. 5. Capturing hash-based NFC M-coupon requirements

This will check the integrity of the protocol. Both the cashier and the issuer must agree on all the information in the protocol.

No Multiple Cash-in. We model No Multiple Cash-in in Casper as follows:

```
Agreement(I,C,[X,Offer])
```

This specification states that every time the cashier accepts *X* and *Offer*, there must be a separate occasion where the issuer must have issued them. Hence cashier cannot accept *X* and *Offer* more times than issuer sent them.

Figure 5 illustrates a scenario where the cashier is engaging in the protocol twice, with one issuer run. The first time the cashier runs the protocol with the user's mobile, and the second, illegal, time with Mallory who might be an intruder or the user himself. The second *Commit* should not occur if there was not a separate *Running*.

5.2 Intruder Knowledge

The analysis also requires us to define the initial knowledge of the intruder. The intruder knows the following: the identities of himself, the user and the cashier, and the hash function.

5.3 Assumptions

There is an assumptions made by the protocol's designers that the client's ID is bound to the client's mobile device, and therefore the client is authenticated at issuing and cashing phases. Therefore, we analyse this protocol in two different assumptions: as no assumption made (the Dolev-Yao model) and as the user's ID is bound to the mobile.

The main goal for analysing the protocol under this assumption is that if an attack is discovered under the Dolev-Yao model, then we should examine if the attack still applies under the assumption made.

We blind message 1 from the intruder i.e. the intruder can not eavesdrop, fake, re-ascribe or redirect message 1. We model this in Casper as follows:

```
#Channels
1 C NF NRA NR
```

6 Analysis

The outcome of the analysis shows no attack on Confidentiality or Forgery Protection.

However, attacks were found on the properties of Not Transferable, Data Integrity and No Multiple Cash-in. The main vulnerability is a simple logical attack against the hashes of the M-coupon. The identity of the user attached to the M-coupon can be easily extracted and changed to any identity. If we consider the M-coupon, the identity $ID(u)$ is not attached correctly to the M-coupon. Anyone is able to compute the first two parts, $ID(u)$ and V to get $h[ID(i)]$:

$$h[ID(i)] = ID(u) \oplus V$$

By obtaining $h[ID(i)]$, the intruder is able to attach any identity, such as $ID(intruder)$ without changing the third part C, and thus produce a new coupon M':

$$V' = ID(intruder) \oplus h[ID(i)]$$
$$M' = ID(intruder)\ ,\ V'\ ,\ C$$

Even though this analysis was under the Dolev-Yao threat model, the properties are still broken under the assumption of a bounded user ID and in a wireless context. The attacker still could change the user identity in an eavesdropped M-coupon to his own identity, and cash it in with the cashier. The intruder could even know the user ID by pretending to be an issuer.

As far as the analysis is concerned, the *Unauthorized Copying* property can be divided into two properties: *Not Transferable* and *User Authentication. Not Transferable* is an agreement between the issuer and the cashier that whatever user identity presented at issuing phase, it should not be changed during the protocol. On the other hand, *User authentication* is stronger in that the identity of the user must also be the one who it is claimed to be. I.e. it is an agreement between the user and the cashier. This protocol only tries to address the requirement of Not Transferable, which may be sufficient in the case of their assumption or in a secure and trusted issuing phase.

Table 1. Hash based protocol and provided solutions against intended/addressed/failed requirements

	Hash-based	Enhanced Hash-based	Footfall	Premium
Confidentiality	√	√	√	√
Forgery protection	√	√	√	√
Data integrity	x	√	√	√
No multiple cash in	x	√	√	√
Not transferable	x	√		√
User authentication				√

7 Suggested Solution

We suggest three solutions to address the found vulnerability: An *enhanced hash-based protocol* which is a solution based on the assumption of the bounded ID assumption. In addition, We provide two kinds of marketing-oriented M-coupon protocols, *the footfall M-coupon protocol* and *the premium M-coupon protocol*, both of which are analysed within the Dolev-Yao model. Table 1 summarises the solutions provided against the properties they address.

7.1 Enhanced Hash-Based Protocol

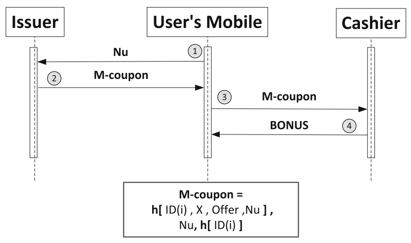
In order to address the broken properties (Not Transferable, Data Integrity and No Multiple Cash-in) in the original hash-based protocol, the identity of the user must be attached correctly to the coupon. This solution must be only considered in a secure and trusted issuing phase. The change needed is to replace C in Fig. 4 to become:

$$C = h[h[ID(i)], X, \mathit{Offer}, ID(u)]$$

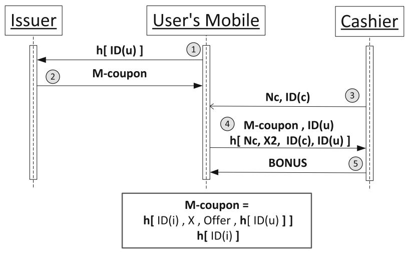
As far as the Not Transferable property is concerned, it is only useful within a trusted issuing phase, which is not always the case. The fact that user ID can be faked by anyone makes combining the User Authentication property with Not Transferable property more useful and meaningful. So, the best choice would be to use them all: the premium protocol, or drop them all: the footfall protocol.

7.2 Marketing-Oriented Protocols Solutions

The footfall M-coupon protocol is used when the main purpose of the M-coupon is to increase the number of people visiting the shop, regardless of whom is using it. Conversely, the premium M-coupon protocol is used when the client has paid for it, and only the intended user is allowed to cash it. The premium M-coupon protocol addresses all requirements discussed in Sect. 3. The footfall M-coupon protocol addresses the same requirements, except for Not Transferable and User Authentication.



(a) the footfall M-coupon protocol


Issuer
User's Mobile
Cashier
1 h[ID(u)]
2 M-coupon
3 Nc, ID(c)
4 M-coupon , ID(u) h[Nc, X2, ID(c), ID(u)]
5 BONUS
M-coupon = h[ID(i) , X , Offer , h[ID(u)]] h[ID(i)]

(b) the premium M-coupon protocol

Fig. 6. Suggested solution

The Footfall M-coupon Protocol. Fig. 6a shows the footfall M-coupon protocol.

Footfall/Premium protocol notations:

ID(i) = Issuer ID.
ID(u) = User ID.
ID(c) = Cashier ID.
Offer = Data about the Offer.
X = A secret key between the issuer and the cashier.
X2 = A secret key between the user and the cashier.
Nu = User's nonce (random number).
Nc = Cashier's nonce.
h[] = Hash function.

There are four messages in this protocol:
1. U → I : Nu
2. I → U : M-coupon = h[ID(i), X , Offer ,Nu] , Nu, h[ID(i)]
3. U → C: M-coupon = h[ID(i) , X , Offer ,Nu] , Nu, h[ID(i)]
4. C → U : BONUS

After the user brings his mobile close to the issuer, his mobile sends a random number Nu (message 1). Then, the Issuer sends the M-coupon to the user (message 2). The M-coupon contains a hash of the issuer identity, the secret key X, the promised offer and user random number. In addition, a hash of the issuer identity is sent, and the user's random number. Then, the user brings his mobile near the cashier and sends the M-coupon (message 3). From the table of hashed issuer identities, the cashier uses h[ID(i)] to find the corresponding ID(i), secret X and the offer. The cashier can check the validity of the M-coupon. Through the nonce Nu the cashier can manage the M-coupon that every issued M-coupon has a unique random number. The cashier can, for example, stop using the M-coupon after five uses. Finally, if all these conditions are satisfied, then the bonus is given to the user (message 4).

Confidentiality, Data Integrity and Forgery Protection requirements are ensured by use of the secret value X and offer. Multiple Cash-in can be managed by including the nonce. However, a stronger Multiple Cash-in is provided in the premium M-coupon protocol.

```
Confidentiality-Footfall: StrongSecret(C,x,offer,[I])
Forgery Protection-Footfall: NonIAgreement(I,C,[x,offer,nu]
Data Integrity-Footfall: NonIAgreement(I,C,[x,offer,nu,I])
No Multiple Cash-in -Footfall: Agreement(I,C,[x,offer,nu])
```

The Premium M-coupon Protocol. The main enhancement in this protocol is attaching an authentic user identity to the coupon. I.e. addressing Not Transferable and User Authentication. Figure 6b illustrates the premium M-coupon protocol.

1. U → I : h[ID(u)]
2. I → U : ***M-coupon***
3. C → U : Nc , ID(c)
4. U → C : ***M-coupon*** *, ID(u) , h[Nc , X2 , ID(c) , ID(u)]*
5. C → U : BONUS
M-coupon *= h[ID(i) , X , Offer , h[ID(u)]] , h[ID(i)]*

The user's mobile sends a hash of his identity ID(u) to the issuer (message 1). Then, the issuer sends the M-coupon to the user (message 2). The M-coupon contains a hash of: the issuer identity, the secret X, the offer and the hashed user's identity. In addition, it contains a hash of the issuer identity. At the cashing phase, the cashier sends his identity and a nonce Nc (message 3). At message 4, the user sends the M-coupon and the user identity, with a new hash containing Nc, the secret value X2, the cashier identity and the user identity.

The cashier can send the bonus based on verifying the two hashes in message 4. From the table of hashed issuer identities, the cashier uses h[ID(i)] to find the corresponding ID(i), secret X and the offer, with user identity known from message 4, the cashier can check the validity of the M-coupon. The second hash authenticates the user, the cashier uses ID(u) to find the corresponding secret X2, and combines it with already known data (Nc, ID(c), ID(u)) to check the validity of the second hash. The cashier can link the M-coupon hash with the second one by checking that both of them include the same identity ID(u).

```
Confidentiality-Premium: StrongSecret(C,x,offer,x2,[I])
Forgery Protection-Premium: NonIAgreement(I,C,[x,offer]
Data Integrity-Premium: NonIAgreement(I,C,[x,offer,I,U])
No Multiple Cash-in-Premium: Agreement(I,C,[x,offer,U])
Not transfarable-Premium: NonIAgreement(I,C,[U])
User Authentecation-Premium: Agreement(U,C,[nc,x2,U])
```

We formally verify the security of these solutions: the Casper/FDR2 analysis found no attacks.

8 Conclusion

We used the formal model-checker Casper/FDR2 to examine a hash based M-coupon protocol and check whether it meets its requirements. The outcome of the analysis shows a simple logical attack in the hash combination of the M-coupon, which damages many of protocol's requirements. Solutions were provided based on two assumptions: when the issuing phase is trusted where the intruder is more restricted; and where the intruder has the power to claim any identity. This paper can be considered as a case study of how a black-box analysis can provide powerful results about NFC protocols.

Acknowledgement. This research was supported by Ministry of Higher Education in Saudi Arabia. We thank the anonymous reviewers for their constructive comments.

References

1. ISO/IEC: Information technology - telecommunications and information exchange between systems - near field communication - interface and protocol (NFCIP-1) (2004)
2. Finkenzeller, K.: RFID Handbuch: Fundamentals and Applications in Contact-less Smart Cards, Radio Frequency Identification and Near-Field Communication, 3rd edn. John Wiley and Sons, Ltd., New York (2010)
3. Haselsteiner, E., Breitfuß, K.: Security in near field communication (NFC). In: Proceedings of Workshop on RFID and Lightweight Crypto (RFIDSec06) (2006)
4. Mulliner, C.: Vulnerability analysis and attacks on NFC-enabled mobile phones. In: ARES, pp. 695–700 (2009)
5. Juniper Research: Mobile coupons – ecosystem analysis and marketing channel strategy 2011–2016. Technical report, Juniper Research (2011)
6. Clark, S.: Survey: discounts and coupons will drive adoption of mobile payments (2011). http://www.nfcworld.com/2011/06/23/38289/survey-discounts-and-coupons-will-drive-adoption-of-mobile-payments
7. Smart Card Alliance: Proximity mobile payments business scenarios: Research report on stakeholder perspective. Technical report, Smart Card Alliance (2008)
8. Brown, C.: The future is NFC says coupons.com exec (2011). http://www.nfcworld.com/2011/03/10/36399/the-future-is-nfc-says-coupons-com-exec/
9. Wolverton, T.: Disney battles coupon goof (2002). http://news.cnet.com/2100-1017-964831.html
10. Hsiang, H.C., Shih, W.K.: Secure mcoupons scheme using nfc. In: International Conference on Business and Information (2008)
11. Lowe, G.: An attack on the needham-schroeder public-key authentication protocol. Inf. Process. Lett. **56**(3), 131–133 (1995)
12. Lowe, G.: Casper: a compiler for the analysis of security protocols. J. Comput. Secur. **6**(1–2), 53–84 (1998)
13. Hoare, C.A.R.: Communicating Sequential Processes. Prentice-Hall, Upper Saddle River (1985)
14. Ryan, P.Y.A., Schneider, S.A., Goldsmith, M., Lowe, G., Roscoe, A.W.: Modelling and Analysis of Security Protocols. Addison-Wesley-Longman, New York (2001)

15. Donovan, B., Norris, P., Lowe, G.: Analyzing a library of security protocols using Casper and FDR. In: Proceedings of the Workshop on Formal Methods and Security Protocols (1999)
16. Dolev, D., Yao, A.: On the security of public-key protocols. IEEE Trans. Inf. Theory **2**(29), 198–208 (1983)
17. Alshehri, A., Schneider, S.: Formally defining NFC M-coupon requirements, with a case study. In: International Conference for Internet Technology and Secured Transactions, ICITST 2013 (2013). doi:10.1109/ICITST.2013.6750161, http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6750161&tag=1