IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 24, 2019, accepted July 8, 2019, date of publication July 15, 2019, date of current version August 1, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2929011

RONFC: A Novel Enabler-Independent NFC Protocol for Mobile Transactions

ISMAIL TURK[1,2], PELIN ANGIN[2], AND AHMET COSAR[3]
[1]Symphony Communication Services, Palo Alto, CA 94304, USA
[2]Department of Computer Engineering, Middle East Technical University, 06800 Ankara, Turkey
[3]Department of Computer Engineering, University of Turkish Aeronautical Association, 06790 Ankara, Turkey

Corresponding author: Ismail Turk (ismail.turk@symphony.com)

ABSTRACT The use of near field communication (NFC) technology for contactless mobile transactions has become popular in the past decade with the availability of this technology in mobile devices. Today, there are millions of the NFC-enabled mobile handsets in the market, with mobile handset manufacturers and mobile network operators enabling m-wallet solutions using the *secure elements* (*SEs*) that they own, thus can remotely control, on the devices. While this approach gives full control to the SE owner to activate any mobile transaction system on a device, having a more flexible approach would increase the benefits that end users could obtain from this technology in a variety of use cases. In this paper, we introduce a novel protocol for the NFC-based mobile transaction procedure, which uses tamper-resistant SEs that are already installed at the transaction terminals, and is mobile handset manufacturer and mobile network operator-independent. We evaluate and show the feasibility of the use of our proposed model with common mobile electronic payment scenarios. The evaluation results demonstrate that the proposed solution is promising for adoption as a secure NFC transaction model, which will have applications in various security-sensitive IoT scenarios, including but not limited to, mobile identification, healthcare, payment, and access control.

INDEX TERMS Contactless transactions, mobile wallet applications, near field communication, secure element.

I. INTRODUCTION

Mobile devices have replaced their desktop counterparts for many daily tasks in the past decade and have become the end users' primary choice for Web-based operations. To illustrate, mobile banking adoption resulted in a number of banks closing some of their branches in the recent years [1]. As a result, many new features were added to smartphones in order to increase their capabilities and to fulfill end user expectations. The Near Field Communication (NFC) technology [2], which facilitates secure data exchange over very short distances on mobile devices, has transformed the mobile transaction industry to include updated specifications, covering payment schemes based on NFC [3]. Consequently, card specifications have also been updated, to include mobile residence of the Secure Element (SE) – a tamper-resistant chip – and its management [4].

The existing infrastructure for NFC-enabled devices permits the SE owner exclusively to enable any feature on the NFC mobile device, which makes the technology dependent on the NFC Enabler (device manufacturer and network operator). To illustrate, if the owner of a payment system wants to accept mobile NFC payments, the integration must be completed with all NFC Enablers, so that payment credentials can be issued to the NFC device SEs, which are under enabler control. The difficulty of reaching this kind of agreement stands as a barrier for more widespread adoption of the NFC technology for mobile transactions.

In this paper, we propose a new NFC protocol – RONFC – which enables mobile transactions to be performed using NFC devices without dependency on NFC Enablers. The proposed solution requires the transaction terminal to have a tamper-resistant chip for calculating the transaction-related cryptogram and advertising the transaction information on the NFC interface using card emulation. In this way, the NFC device reads the information like an NFC card and performs the transaction through the associated mobile application. The main goal of the approach is to remove the dependency on NFC enablers for NFC-based mobile transactions, thereby providing the chance for wider adoption of the

The associate editor coordinating the review of this manuscript and approving it for publication was Shiqiang Wang.

technology in various domains requiring mobile transaction processing.

The main contributions of this paper can be summarized as follows:

- We propose a novel NFC protocol for mobile transactions, which is independent of NFC enablers, thus giving full control to end users to conduct transactions on their mobile device by just using the corresponding application.
- As opposed to legacy NFC-based solutions that offer limited space on the SE for storing the transaction applications, the proposed model enables conducting the transactions through applications of the mobile device itself, providing increased versatility.
- The proposed approach creates the basis for an open protocol for anyone seeking to enable NFC-based transactions in their mobile systems. The independence of the approach from the Card Emulation Mode, which is an optional feature in NFC specifications, makes it compliant with all NFC-enabled devices, enabling widespread adoption.
- The extensible open architecture of the proposed model makes it possible to integrate various authentication methods such as face recognition and fingerprint recognition, providing increased security for sensitive mobile transactions.

The remainder of this article is organized as follows: In Section II we provide preliminaries on NFC and SE, and discuss the NFC enabler dependency problem. In Section III, we provide an overview of related work in mobile transaction models and solutions for the NFC enabler dependency problem. Section IV introduces our proposed approach for NFC-based mobile transactions, illustrating the concepts through the use case of mobile contactless payments. Section V provides a performance and security evaluation of the proposed approach. Section VI provides a discussion of the contributions of the proposed approach and Section VII concludes the paper.

II. PRELIMINARIES

NFC is the technology which lets a contactless device communicate with another contactless device within a range of a few centimeters [5]. A mobile device can use this technology in the following modes of operation:

- Reader Mode: Allows a mobile device to read and write an NFC tag.
- Peer-to-Peer Mode: Allows two NFC devices to exchange information between one another.
- Card Emulation Mode: Allows a mobile device to function as a contactless card.

Although each operation mode has its own advantages, *Card Emulation* has attracted particular attention in the secure identification industry, because this mode allows the user to convert the mobile device into a mobile wallet, which contains credit card information, offline payment cards, loyalty cards, etc., all of which are stored inside the SE [6].

As defined by GlobalPlatform [7], "A SE is a tamper-resistant platform (typically a one-chip secure microcontroller), capable of securely hosting applications and their confidential and cryptographic data (for example cryptographic keys), in accordance with the rules and security requirements set by well-identified and trusted authorities."

An NFC device has two options for including the SE: the SIM-based NFC, which includes the SE in the SIM card itself or the Embedded NFC, which utilizes mobile handset-embedded hardware. Both methods allow the owner of the device to load the necessary credentials to their devices remotely, because the device is already under owner control. Therefore, mobile transaction functions are enabled via remote issuance to the SE inside the mobile device [8].

The security of an SE is ensured by cryptographic *issuer keys*, which are assigned to and only known by the SE owner [9]. SE issuer keys are utilized for remote authentication when installing applications into the chip. Since security-sensitive mobile application installation and personalization requires the transfer of confidential information, secure communication with the remote chip is required, so as to ensure that confidential information is delivered securely [10].

The digital encoding of a physical contactless card differs slightly from the digital encoding of an instance within the mobile device. The physical contactless card is manufactured in secure environments and the target user information is already known when it is being created. However, when the SE of the mobile device is manufactured, the final user is unknown, as well as the applications it will contain. All issuances are performed while the mobile device is actively in use. Therefore, NFC issuance requires remote access to the SEs in order to handle any contactless application activity. As a result, NFC issuance is dependent on the NFC Enabler. This constraining design is thought to have been intentional, as the mobile payment industry players sought to dominate mobile payment platforms and monopolize payment technology during its infancy stage [11].

Dependency on the NFC Enabler is a strong impediment to widespread adoption of the NFC technology in mobile transactions, as extensive integrations with worldwide NFC Enablers are an arduous task. As a result, even after more than a decade since implementation, the NFC technology was only successfully implemented in credit card issuance because of international standards, despite its high potential for facilitating the daily operations of many local operators, small companies and system providers.

III. RELATED WORK

NFC, since its rise as a communication technology, has been utilized as a key component in many fields of mobile computing with recent research focusing on NFC-based solutions for mobile tourism applications [12], wireless power transfer systems [13], airport baggage claim systems [14], and game-based learning applications [15] in addition to well-known mobile payment applications.

The first mobile wallet applications used SIM cards to store private credentials because of proven security of the SIM, and then evolved with secure key exchange protocols in order to ensure end-to-end security [16]. SIM cards are a specific form of smart card that are considered a secure signature creation device; a prerequisite for electronic signatures [17]. For this reason, it became possible to design a mobile transaction system using the SIM as a Public Key Infrastructure (PKI) signature tool [18]. However, there is a key problem with this method: Since SIM cards are under the full control of the Mobile Network Operator (MNO), designing a SIM card-based system results in full dependency on the MNO. Considering the problems that have already occurred due to MNO dependency, some researchers began to study MNO-independent SIM usage scenarios. Specifically, a mobile signature service independent of the MNO was developed using the SIM card as the private key storage medium [19]. Although this approach solves the e-signature flow over SIM cards without MNO dependency, this is not applicable to mobile payments, as it involves a many-to-many relationship between terminal providers, merchants and card providers, and a secure transaction requires cryptographic proof for all. Therefore, having m-signature on a handset for a specific issuer does not solve the NFC enabler independent mobile payment problem.

The embedded Secure Element (eSE) is an alternative to the SIM card, offering secure storage within an NFC mobile device. However, this method is not without mobile handset manufacturer dependency. Adding a payment instrument into Apple Wallet and Samsung Pay is only possible by processing it through Apple™ and Samsung™, which will require contractual agreement with these corporations.

There have been attempts to solve the problem of NFC enabler dependency by incorporating user-centric models. Specifically, a consumer-centric model was proposed by Akram et al. for smart card management, containing several applications within one card [20]. Over time, even international standards included consumer-centric models for NFC technology [21]. However, solving this problem does not give flexibility to the end-user when it comes to using a mobile application of choice for NFC transactions at different merchants, because SE can only host a few applications, whereas handsets can host many more mobile applications. On the other hand, adding payment function to any mobile application requires executing SE application bootstrapping – installing a corresponding smart card application into the SE – which is another barrier for small businesses. Therefore, user-centric card management approaches can be ideal for remote smart card management, but not for mobile payments.

Another way to remove dependency on the NFC Enabler is by replacing the SE of the NFC device with cloud-based SEs [22]. However, this approach comes with its share of network latency, which is not acceptable in many real-world mobile transaction scenarios. For this reason, we tried to leverage from 4G-based mobile network communication speed to solve this problem in one of our previous studies [23].

On the other hand, it is also possible to create an alternative scheme, which does not contain an SE at all [24]. Some studies analyzed the use of Quick Response (QR) Code technology as a method to pay with a mobile handheld device. Specifically, researchers examined a novel, QR code-based mobile payment system, which does not require the mobile device to have an Internet connection [25]. Although these solutions can address specific use cases, they are not ideal for payment processing, as financial transactions require cryptographically proven security in the first place. Transaction data confidentiality and integrity should be ensured by preparing this data in proven secure hardware components that are approved by the financial industry. This requirement cannot be met with light-weight, software-backed, cryptogram generation solutions or with QR codes that do not allow mutual authentication between components during a transaction, besides being vulnerable against copying visual transaction data.

Over the past decade we have witnessed the evolution of m-commerce [26]. Researchers have come face-to-face with the tremendous potential of m-commerce; however, it has not gained the expected popularity and still lacks a solid solution for more widespread adoption due to the lack of standardized guidelines and lack of players [27].

The approach proposed in this paper differs from previous NFC-based mobile transaction processing approaches in that it removes the dependency on NFC Enablers without requiring usage of external instruments such as cloud-based SEs and enables widespread adoption by only relying on the compulsory Read Mode on NFC-enabled mobile devices.

IV. PROPOSED APPROACH

In this section, we describe our proposed enabler-independent transaction execution flow model called *RONFC* for NFC-based mobile transactions, which works regardless of NFC being SIM-based or embedded SE-based. The proposed method does not even require an SE to be present on the NFC mobile device.

In our proposed solution, we position the SE inside the transaction terminal to create a cryptogram for each transaction. The mobile device reads the information from the terminal through NFC and performs the transaction. In other words, the terminal is required to enter Card Emulation Mode, while requiring the mobile device to *only read* it as it would a plastic card. Thus, we refer to the proposed solution as ''RONFC'', which is short for Read-Only NFC.

Terminals are under control of the Terminal Providers (TP), making it is easy for them to manage the SE inside the terminal. Contacting any Trusted Service Managers (TSM) is not required to reach the terminals or the SE attached to them.

The NFC device is used to communicate with the terminal using the NFC interface, which can be performed by any

TABLE 1. Symbols and abbreviations.

CP	Card Provider
CA	Central Authority
TP	Terminal Provider
SE_{pub}	Public key of the terminal Secure Element
SE_{pr}	Private key of the terminal Secure Element
TP_{pub}	Public key of the Terminal Provider
TP_{pr}	Private key of the Terminal Provider
SE_ID	Unique ID of the SecureElement at the transaction terminal
CP_{pub}	Public key of the Card Provider
CP_{pr}	Private key of the Card Provider
CPCert	Public key certificate of the Card Provider
PID	Unique ID of the Terminal Provider that approves the transaction
TID	Unique Terminal ID that is assigned within the range of Terminal IDs provided by the Central Authority to the Terminal Provider.
CID	Unique ID of the Card Provider used in the transaction
$RndSE_t$	16-byte nonce created by the terminal SE for the transaction identified by t
TranID	Unique transaction identifier created by the terminal
E(M, K)	Encryption of message M with key K
D(C, K)	Decryption of ciphertext C with key K
S(M, K)	Digital signature on message M with private key K of the signer
V(M, S, K)	Verification of digital signature S on message M with public key K of the signer
$CryptoKey_t$	Random AES symmetric-key generated for for the transaction identified by t
$CDATA_t$	Concatenation of transaction data components for the transaction identified by t

mobile application, as this is a mandatory function for an NFC-enabled mobile device. Below, we describe the proposed approach, giving an overview of the system components and their interactions through the use case of a mobile payment transaction, followed by the details of the transaction execution flow. Although the mobile contactless payment scenario is chosen for illustration and ease of explanation here, the approach is applicable with minor modifications for various other NFC-based mobile transactions, such as identity verification (e.g. electronic passports), mobile access control, secure entry, etc. Table 1 provides a list of the symbols and abbreviations used in the protocol description.

A. MOBILE PAYMENT SCENARIO

Contactless mobile payments have been the most popular application of the NFC technology since its invention. In this section, we provide an overview of the functioning of the proposed NFC protocol for a typical mobile payment scenario. Like any legacy secure mobile payment processing system, our solution requires a Central Authority (CA) to manage the operations between the card acquirer and the card issuer. Terminal Providers (TP) here are the existing terminal owners, which are utilized for credit card or any other payment form's acceptance. Card Providers (CP) are the main card issuers. These can be a credit card issuing bank or any other proprietary payment card issuer.

The processing of a mobile payment transaction with RONFC involves the following steps:

1) The payment terminal uses its SE to create a cryptogram for the current transaction data.[1]
2) The payment terminal goes into NFC Card Emulation Mode to advertise the created transaction data through the NFC interface, connects to its TP Web service and waits for a reply once the transaction has started.
3) The mobile device user launches the CP's mobile application to initiate a payment and taps his phone to the payment terminal.
4) The mobile application reads the transaction data and displays the transaction details to the user for verification.
5) Upon verification, the mobile application connects to the CP Web service using a proprietary communication channel, which ensures the security and confidentiality between the remote device and the host system, by employing robust authentication protocols [28].
6) The CP checks the user's credentials and decides if the transaction is approved or denied.
7) Once approved, the CP signs the transaction data as proof of the approval and sends the approval to the CA.
8) The CA identifies the TP from the transaction data and forwards the transaction approval to it.
9) The TP decrypts the message prepared by the SE, and checks the transaction approval given by the CP. Once completed, it prepares approval data for the terminal.
10) The TP replies to the request generated by the payment terminal (created at Step 2).
11) The payment terminal receives the message and verifies it using the SE. The user screen of the terminal displays the verification result.

All communication between the terminal and the Payment Terminal Applet is based on the international standards of Command/Response schemes [29]. The communication between the terminal and the Terminal Provider, and the communication between the mobile application and the Card Provider is proprietary. Thanks to SSL communication, if properly configured, both entities can ensure that the integrity and the confidentiality of the information is guaranteed and we can rely on secure information exchange in mobile networks [30].

Information delivery and acknowledgements are not explained in detail here, as with fair exchange solutions a valid message delivery system can be built [31].

For public-key cryptography, we choose to use Elliptic Curve Cryptography (ECC), as it requires lower key lengths

[1] The SE inside the terminal has a smart card applet to perform these tasks. The SE can be a multi-application chip featuring some other applets on it as well. Alternatively, the terminal applet can reside in a multi-application chip on the terminal that was previously installed for other purposes on the terminal.

for equal security strength when compared to RSA [32]. Please note that the specific encryption and decryption algorithm parameters are to be chosen by the implementer of the proposed scheme depending on the security requirements of the domain as we aim the scheme to be flexible, therefore specific parameter values are not provided here.

B. TRANSACTION EXECUTION FLOW

This section explains in detail the proposed end-to-end mobile transaction execution flow in RONFC. The flow for the mobile payment scenario is as depicted in Fig. 1.

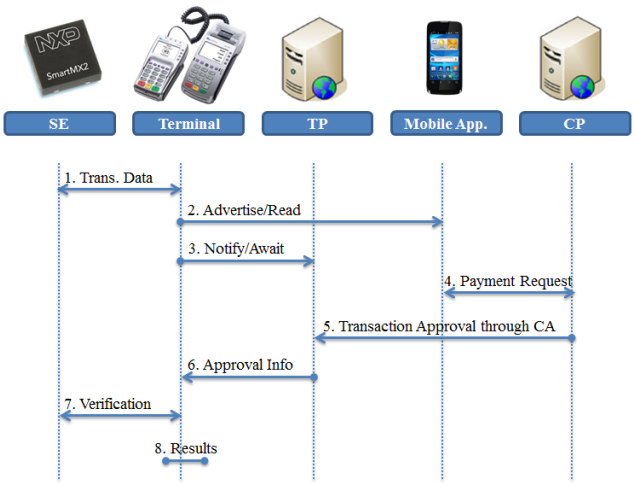

FIGURE 1. RONFC transaction execution flow.

As seen in Fig. 1, the processing of a mobile transaction is separated into eight phases, detailed below.

1) CREATE TRANSACTION DATA

A transaction starts with preparation of the transaction data and its associated cryptogram at the terminal.[2]

Once the terminal SE receives the *Create Transaction Data* command, it generates the secure transaction data using Algorithm I.

With Algorithm I, SE prepares secure transaction data that CP, CA and TP can use to verify the security and the integrity of the transaction request. Algorithm attributes and how they contribute achieving secure transaction data generation are explained below.[3]

- TranCnt is an internal counter in the card applet that is matched and verified at TP in order to avoid double-push and replay attacks.
- TransactionInfo is a concatenation of the unique transaction ID that is provided by the terminal, transaction data, and unique identifiers of the terminal device, terminal provider and SE itself. This information is then encrypted and signed by SE. Thus, each transaction request prepared by SE cryptographically ensures that no transaction attributes can be altered or replaced in transit.
- TransactionSecret is generated to make each transaction cryptogram unique even if all transaction attributes are the same. A random byte array is appended to the transaction secret, which then becomes input for other cryptographic calculations within the algorithm. This ensures transaction data generation security against side channel power analysis attacks.
- A random CryptoKey is generated for each transaction in order to perform transaction data encryption with a different key for each transaction request. This is then encrypted with TP's public key, making decryption possible by TP only.
- Finally, the transaction data is signed with the private key of SE, making terminal generated transaction cryptogram verification possible for CA, which maintains a list of SE public keys and the corresponding certificates.

Algorithm 1 Secure Transaction Data Generation

Input: TranID, $CDATA_{TranID}$

1: TranCnt ← TranCnt + 1 // update the transaction count
2: TransactionInfo ← TranID || $CDATA_{TranID}$|| TID || PID || SE_ID
3: TransactionSecret ← TranID || $RndSE_{TranID}$|| TranCnt
4: SecuredCryptoKey ← E($CryptoKey_{TranID}$, TP_{pub})
5: EncTranData ← E((TransactionInfo || TransactionSecret), $CryptoKey_{TranID}$)
6: TranSignature ← S((TransactionInfo || TransactionSecret), SE_{pr})
7: SecureTransactionData ← (SecuredCryptoKey || EncTranData || TranSignature)
8: return SecureTransactionData

TABLE 2. Sample transaction data components for the mobile payment scenario.

Tag	Length	Value
TranID	4 bytes	4 byte unique transaction ID created by the payment terminal.
Amount	4 bytes	Amount is encoded in 4-byte binary format with an implicit decimal point. For example, 123 (0x7B) is 1.23
Currency	2 bytes	Currency code based on ISO 4217 [33].
Date/Time	6 bytes	3 bytes for date as YYMMDD, 3 bytes for time as HHmmSS
TID	4 bytes	Unique terminal ID that is assigned within the range of terminal IDs provided by the CA to TP.
TerminalName	20 bytes	Any given terminal name

Typical transaction data components for the mobile payment scenario are given in Table 2.

[2]For the case of a mobile payment, the merchant attendant must start the flow from the payment terminal by providing the transaction-related information; namely, the amount, currency, etc., and the payment terminal software must select the corresponding applet inside the SE, then send the transaction data components according to Table 2.

[3]The symbol || denotes concatenation.

2) ADVERTISE TRANSACTION DATA, READ BY NFC

Once the terminal has prepared the secure transaction data, it performs two parallel operations. One is to notify the TP from which this terminal is waiting for a transaction response and the other is to turn on the Card Emulation Mode, in order to allow the NFC-enabled mobile device to read it as a regular NFC tag.[4] The selection of the tag and the command/response formats need to be arranged according to the NFC Forum Specifications for the Type 4 Tag [5].

The terminal will prepare the NFC data by concatenating *TransactionInfo* and *SecureTransactionData* provided by the SE. The data needs to be encapsulated in an NFC Forum Text Format – NFC Data Exchange Format (NDEF) message according to the specifications [5].

The user of the NFC device is already authenticated to its CP by using the mobile application of the CP. The user performs necessary operations to trigger the transaction request on the mobile application and then taps the device to the terminal. The mobile application selects the NFC Forum Type 4 Tag and reads the NDEF message.

The mobile application extracts the *TransactionInfo* and *SecureTransactionData* from the NDEF message. It parses the *TransactionInfo* and displays the relevant transaction information through its user interface. The user confirms the accuracy of the displayed transaction details and permits continuation of the transaction. The mobile application connects to its CP to request execution of the transaction.

3) NOTIFY/AWAIT TRANSACTION

Once the NFC data of the transaction is read by the mobile device, the terminal notifies its provider about the initiated transaction, invoking the relevant method of the TP's Web service with *SecureTransactionData*. This blocking method waits for the CP to send the transaction approval before replying to the call.

4) CARD PROVIDER REQUEST

After Step 2, the mobile application is asked to show its user-related information regarding the transaction and asked to let the user choose the relevant transaction instrument. For example, in a credit card payment scheme, the bank's mobile application will let the user choose a credit card already available in his/her bank account.

The mobile application is responsible for making the call to the CP's Web service in order to initiate the transaction request and delivering the transaction data information read from the terminal.

Once the request is received by the CP, Algorithm II is run for transaction approval/rejection.

Transaction processing is a proprietary flow for the Card Provider, as it involves credibility or balance check of the user. Therefore, details of that process are not part of this protocol. Once a result is obtained from transaction processing, CP prepares a result message and signs that data with its private key. CA maintains public keys and corresponding certificates of all CPs registered in the network. Therefore, each signed transaction result cryptographically proves CP's approval or rejection of the request. Before proceeding to the next step, CP replies to the transaction requesting mobile application, so as to notify the user about the result of the transaction.

[4]For easy detection by the NFC device, it is advised that the terminal prepare the Card Emulation to emulate the card as an NFC Forum Type 4 Tag.

Algorithm 2 Card Provıder Approval

Input: TransactionInfo, SecureTransactionData

1: Result ← ProcessTransaction($CDATA_{TranID}$)
2: TransactionData ← TransactionInfo || SecureTransactionData
3: if (Result = reject)
   ResultMessage ← (TransactionData || CID || RejectMessage || TimeStamp)
4: else
   ResultMessage ← (TransactionData || CID || ApprovalID || TimeStamp)
5: CPTranSignature ← S(ResultMessage, CP_{pr})
6: CPApproval ← ResultMessage || CPTranSignature
7: Send transaction result to mobile application
8: return CPApproval

5) TRANSACTION APPROVAL THROUGH CA

After the transaction approval result data becomes ready, the CP calls the CA's Web service method for the delivery of the transaction result to the TP. This method accepts *ResultMessage*, *CPTranSignature*, CP_{pub} and *CPCert* as input and processes them using Algorithm III for verification before communication of results to the TP.

Algorithm 3 CA Verification

Input: ResultMessage, CPTranSignature, CP_{pub}, CPCert

1: CertValid ← CheckCertificateValidity(CPCert)
2: if (!CertValid)
   abort transaction
3: SignatureValid ← V(ResultMessage, CPTranSignature, CP_{pub})
4: if (SignatureValid)
   send (ResultMessage || CPTranSignature || CP_{pub}|| CPCert) to TP
5: return SignatureValid

Upon receipt of the transaction approval message from the CA, the TP checks the received message, verifies the CP certificate, and verifies *CPTranSignature* using CP_{pub}. Upon completion of this verification process, the TP ensures the validity of the delivered approval message and performs the transaction.

Algorithm 4 Approval Data Generation

Input: SecuredCryptoKey || EncTranData || TranSignature
1: CryptoKey ← D(SecuredCryptoKey, TP_{pr})
2: (TransactionInfo || TransactionSecret) ← D(EncTranData, CryptoKey)
3: SignatureValid ← V(EncTranData, TranSignature, SE_{pub})
4: if (SignatureValid) ApprovalData ← TranID || $RndSE_{TranID}$|| TID || PID || SE_ID SecuredApprovalData ← E(ApprovalData, SE_{pub}) ApprovalSignature ← S(SecuredApprovalData, TP_{pr}) return (SecuredApprovalData || ApprovalSignature)

6) APPROVAL INFO

Upon completion of the validation process, the TP processes the transaction using Algorithm IV, with the input parameters received in Step 3.

With Algorithm IV, TP initially accesses transaction data that was encrypted by the SE and then verifies the content by verifying the signature. The symmetric transaction encryption key is accessed by decrypting SecureCryptoKey using the private key of the TP. Then this key is used to decrypt the content prepared by the SE for the transaction.

After signature verification, TP generates ApprovalData, which contains the original TransactionSecret prepared by the SE (TranID, Random array and unique identifiers of terminal, terminal provider and SE). This information is appended to the prepared approval cryptogram, so that the terminal can match the initial request and approval data when it is delivered to the terminal.

TP encrypts ApprovalData with the public key of SE, making decryption possible by the corresponding SE only.

Finally, TP signs secured approval data using the private key of TP. As each SE has the public key of its terminal provider, SE can verify that approval data has been prepared by its own provider when approval data is delivered to the SE.

TABLE 3. Transaction approval data.

Tag	Length	Value
TranID	4 bytes	4 byte Unique Transaction ID created by the terminal.
$RndSE_{TranID}$	16 bytes	16 byte nonce created by the SE for this transaction
TID	4 bytes	ID of the terminal that requested the transaction
PID	4 bytes	ID of the Terminal Provider, which approves the transaction
SE_ID	4 bytes	ID of the SE, which requested the transaction

Since all tags in the ApprovalData are fixed-length values, they can be concatenated without putting a tag identifier heading the values.

The details of *ApprovalData* parameters are given in Table 3.

Algorithm 5 Approval Verification

Input: SecuredApprovalData || ApprovalSignature
1: SignatureValid ←V(SecuredApprovalData, ApprovalSignature, TP_{pub})
2: ApprovalData ← D (SecuredApprovalData, SE_{pr})
3: TransactionID' ← Parse from ApprovalData
4: $RndSE_{tranID}$' ← Parse from ApprovalData
5: if (TranID' = TranID $RndSE_{TranID}$' = $RndSE_{TranID}$) return success
6: else return failure

7) APPROVAL VERIFICATION

In the approval verification stage, the terminal forwards the received approval message to the SE, which then performs the verification using Algorithm V.

As a first step, SE verifies if approval data is generated by its terminal provider. This is performed by verifying the approval signature using TP's public key. Then, SE decrypts SecureApprovalData using its private key to access TransactionSecret sent by the TP. Finally, SE checks if TransactionSecret matches the original transaction attributes generated by this SE and returns success or failure based on the match result.

8) RESULTS SHOWN

The transaction is successfully completed. The terminal delivers relevant success messages to the user and prints a receipt for reference if required by the transaction design.

V. EVALUATION

In this section, we provide a performance and security evaluation of the proposed enabler-independent NFC protocol for mobile transactions.

A. EXPERIMENTAL SETUP

In order to evaluate the performance of the proposed solution for the discussed mobile payment scenario, we implemented a prototype including the Card Provider, Terminal Provider and Central Authority entities, which are all Java applications running on the same server. Additionally, the following entities were implemented as part of the experiment environment:

a) PC application simulating a Payment Terminal such as Verifone™ and Ingenico™.

b) Smart Card (SC) simulating the SE of the Payment Terminal: In order to simulate the SE, we used a Java Card Open Platform (JCOP) smart card with the JCOP v.2.4.2 R3 version. We used a dual interface (contact and contactless) JCOP card so that we could use the contactless interface for NFC communication. We developed an applet on the JCOP, which performs the transaction cryptogram generation and verification as described earlier.

c) Smart card simulating Card Emulation: As the commercial NFC components having Card Emulation are secured

with confidential Issuer Security Domain (ISD) keys, we were not able to use a commercial product for our experiments. Therefore, we used the same JCOP card that we use as the SE. In order to perform Card Emulation, we developed another applet that accepts the NFC data from the terminal and delivers output to the mobile device when requested.

d) PC/SC reader: As our SE is simulated by a real smart card connected to the PC application, we needed to communicate with the smart card through the PC/SC interface. For this purpose, we used the HID OmniKey 5321[5] smart card reader.

e) NFC Phone: We used Samsung Galaxy S4[6] as an NFC phone, as it is already NFC enabled and can read NFC tags.

f) Mobile Application: We developed an Android application that performs the mobile transaction processing application task described in this paper.

g) Key Handling: Since this setup is just for demo purposes, for the PKI-based communication between the Card Provider, Central Authority and Terminal Provider, we used soft-keys generated in Java [34]. The SE key pair was generated using the Bouncy Castle API and hardcoded into the Smart Card application.

For the experiments, a PC/SC reader was connected to the PC that runs the Payment Terminal application and the contactless JCOP card was placed on it. The transaction starts by providing the amount information to the terminal application, which prepares the transaction information and connects to the JCOP card using the PC/SC interface. First, it selects the applet on the chip and sends the cryptogram generation command. Once the terminal receives the response, it selects the other applet that we developed for card emulation simulation. The terminal sends the NFC data to the Card Emulation applet. Finally, the terminal application closes the PC/SC channel so that the card becomes selectable by other media; the NFC phone, in our case. At this stage, we tap the NFC phone to the card. Our Android application reads the NFC data and performs the transaction by connecting to the Card Provider application on the server.

The terminal application makes a request to the Terminal Provider application at the server after closing the PC/SC channel. This call is a blocking request, meaning the terminal application waits until the Terminal Provider application returns a reply or an error. The Terminal Provider application waits for the Card Provider to reach it through the Central Authority application for a specific period and if no connection is made, it sends a timeout error.

When the terminal application receives a reply from the Terminal Provider application, it reconnects to the JCOP card using the PC/SC interface. Finally, it performs the response verification process by invoking the relevant command and shows the result on the screen.

[5]https://www.hidglobal.com/products/readers/omnikey/5321-cl-sam

[6]https://www.samsung.com/uk/smartphones/galaxy-s4-i9505/GT-I9505ZWABTU/

With this setup, we were able to run our proposed mobile transaction protocol end-to-end, hence proving the feasibility of real-world implementation.

B. PERFORMANCE ANALYSIS

Transaction execution time performance is an important factor in mobile transactions. A recent study involving hotel customers aimed to identify consumer acceptance regarding the use of NFC for hotel payments, revealed that the most important expectation of consumers was high performance [35]. The size of the data communicated during the protocol runtime affects the end-to-end execution time, thus should be kept as low as possible.

The total length of the *TransactionSecret* variable in Algorithm I is 24 bytes. TP_{pub} in Step 5 is the public key of the Terminal Provider key pair, injected into all Terminal Provider SEs during the SE preparation. If ECC Prime 256 is used for generating the key pair, this encryption will result in 80 bytes [32]. For the mobile payment scenario, the total length of *EncTranData* is 72 bytes and *TranSignature* length is 80 bytes. As a result, the return value length will be 252 bytes, which is short enough to send in a single response APDU. Therefore, no chaining is required.

In order to measure the transaction execution time performance of the developed prototype, we created five reference points as shown in Fig. 2.

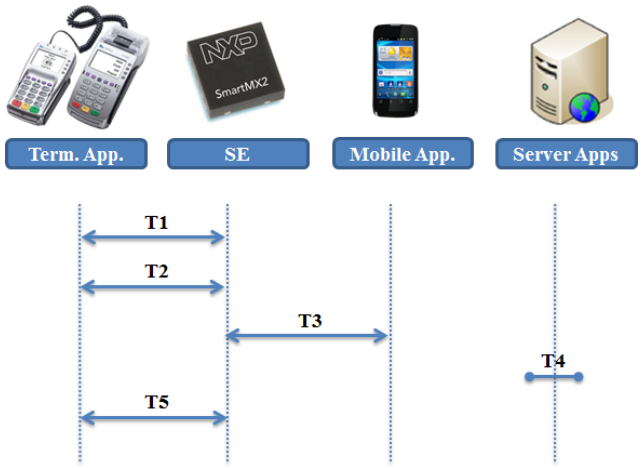

FIGURE 2. Transaction execution time components for the mobile payment scenario.

The T1 reference point measures the total time required by the SE to prepare the transaction data including the cryptographic operations. This includes the time period between selecting the chip applet and receiving the reply for the Create Transaction Data command.

The T2 reference point measures the total time required for loading the SE with NFC data. This actually simulates putting the terminal into Card Emulation Mode. The time measurement starts when the command is invoked and finishes when the SE delivers a success message.

The T3 reference point measures the total time taken by the mobile application to read the NFC data. In order to include the tag discovery time with the measurement, we started the timer when the ''Pay'' button was clicked on the mobile application and stopped it when the mobile application called the Card Provider Server application. For an accurate measurement, we clicked the ''Pay'' button when the phone was physically on the test card.

The T4 reference point measures the total time required to complete the tasks of the Card Provider, Central Authority and Terminal Provider on the server. As all three applications reside on the same server within our test setup, there was almost no network delay. In real life, there will undoubtedly be network delays and this must be taken into account. Therefore, this performance analysis reference point analyzes the total time required by the server for soft-key based cryptographic operations. The Card Provider application creates a log entry when it receives the payment request from the mobile application and the Terminal Provider application creates another log entry in the same file on the server when it replies to the Terminal Application that is waiting for the reply of the transaction request.

The T5 reference point measures the total time that the SE requires to verify the message delivered by the Terminal Provider. The time measurement starts with the selection of the chip applet and finishes with receipt of reply for the Approval Verification command.

We repeated the experiments 20 times and the minimum, maximum and average response times in milliseconds for the different transaction execution time components are reported in Table 4.

TABLE 4. Execution time performance results.

Step	Min (ms)	Max (ms)	Average (ms)
T1	342	588	386
T2	33	70	42
T3	651	2092	874
T4	280	466	321
T5	306	422	345
Total	**1612**	**3638**	**1968**

Note that the total values in the performance analysis results table do not contain any network connection overhead. If we roughly assume that the network overhead were around 2-3 seconds, it would mean that all the payment flow execution would be completed within 5 seconds.

Evaluating the performance results, we conjecture that completing the entire payment within 5 seconds is acceptable, but can be improved by further optimizing the implementation of the involved components. Considering that regular NFC payments also have a 2-3 second network overhead and complete the payment application reading process in about 1 second, the total execution takes around 4 seconds. However, this time excludes the total time spent by user to enter their PIN, bringing the total time to more than 5 seconds. In our design, the user is able to perform verification upfront using the mobile application, therefore the end-to-end payment flow execution offers a swift transaction completion compared to a regular NFC payment. Also, note that the provided numbers are based on a specific implementation of the mobile payment scenario to demonstrate the feasibility of the proposed approach. Real world implementations of the proposed protocol for various domains will have varying performance results.

C. SECURITY ANALYSIS

The end-to-end security of RONFC primarily relies on the security of the components involved in the protocol, as well as the security of their communication. Thus, in this section, we discuss the security of the individual system components and we analyze their role in overall system security. In order to clarify the communication channels between the components, their relationships are depicted in Fig. 3.

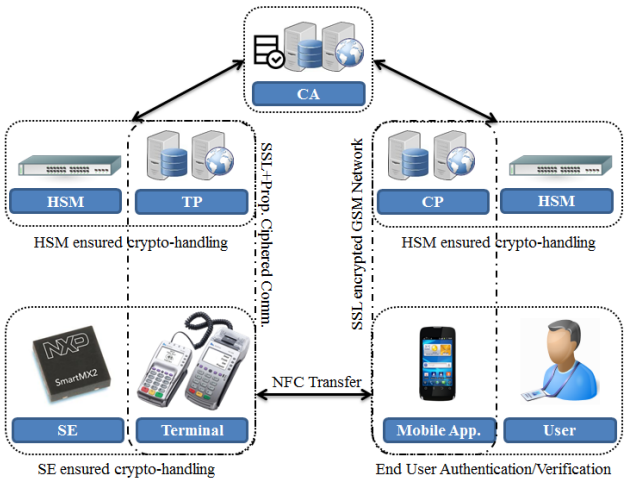

FIGURE 3. Transaction flow components and communication channels.

1) TERMINAL SECURITY

The SE plays an important role in RONFC's overall security, because the transaction starts with the cryptogram created by the SE and ends with the cryptogram verification by the SE. SEs are tamper-resistant and certified at the Evaluation Assurance Level (EAL) 5+ for both the underlying hardware and also the Chip Operating System running on the chip. This ensures that the hardware is protected against attacks and information is securely stored, while also protecting the chip from attacks via the operating system, which also stores the information securely. The terminal application uses the attached SE for key storage, transaction cryptogram calculation and transaction approval verification. Therefore, the terminal application itself is an agnostic entity from a security standpoint, except for the need of ensuring secure communication with its Terminal Provider.

2) TERMINAL PROVIDER AND CARD PROVIDER

The Terminal Provider System and the Card Provider System will be made up of a combination of several Web Services and a cryptographic calculation. We recommend utilizing a Hardware Security Module (HSM) to store the cryptographic keys and perform cryptographic calculations. While HSM devices are similar to smart cards in terms of the ensured security, they are also Common Criteria EAL 5+ certified devices, which ensures the security of the keys inside. For proper handling of the key, providers should follow the recommendations in [36].

3) MOBILE APPLICATION

The mobile application simply reads the NFC data, shows transaction information to the user for verification, transmits to the Card Provider, and thus, plays an agnostic role in terms of cryptogram security. However, the mobile application is in charge of end-user verification prior to triggering the transaction execution. This is similar to signing into the mobile application of a bank either by PIN, fingerprint, or any other user authentication verification mechanism.

4) TERMINAL COMMUNICATION CHANNEL

The communication channel between the terminal and the Terminal Provider carries ciphered data that is encrypted by a secured hardware (HSM) and can only be decrypted by a secured hardware (SE). Thus, it is resistant against man-in-the-middle attacks. However, in order to ensure resistance against replay-attacks and to reduce the number of known transaction samples of a dedicated terminal, it is recommended that the terminal communication channel is secured with an SSL connection and additional proprietary communication-ciphering techniques.

5) CARD PROVIDER CHANNEL

Ensuring user access security on mobile applications was a popular subject of research a decade ago and, since then, advances have been made regarding secure channel creation between a mobile application and its provider [37].

Within our design, the Card Provider channel is very important, as the system does not store card information anywhere; instead, relying on the Card Provider itself which already stores the required information. The Card Provider manages card information as they normally do in their system. Hence it is still the Card Provider's responsibility to ensure security between the mobile application and their host system.

6) NFC DATA TRANSMISSION

The NFC data prepared by the terminal consists of two parts; a public part which is to be read by the mobile application, Card Provider and Central Authority, which then have access to the transaction parameters - such as amount, currency, etc. and a confidential part that is prepared by the SE, which can only be read by the owner Terminal Provider.

The confidential part is prepared based on the PKI infrastructure. Therefore, it remains confidential until it reaches its owner. The public part only contains some unique identifiers and some transaction-related information, which does not include any personal or confidential information.

7) CENTRAL AUTHORITY ENSURED COMMUNICATION

Within our proposed transaction design, the server-side applications – Card Provider, Central Authority and Terminal Provider – communicate with each other, in order to process the transaction. Although SE-encrypted data is still being transmitted and the confidentiality is ensured, there is an additional layer that ensures the security of message delivery on server-side applications. The Central Authority serves as a go-between for communication between the Card Provider and the Terminal Provider. Based on PKI infrastructure, the Card Provider message features the Central Authority certificate in its signed data. In this way, the Terminal Provider ensures that the delivered message is signed by a real Card Provider. Since the Terminal Provider always receives messages directly from the Central Authority, it ensures that the transaction is seen by the Central Authority and that it is the guarantor of the approved transaction.

D. VULNERABILITY ANALYSIS

In order to ensure a fraud-resistant system, a mobile transaction processing system needs to be secure against logical attacks, besides achieving component security. In this section, we discuss possible logical attacks and discuss RONFC's resilience against those attacks.

1) DISHONEST CARD PROVIDER

There are two main scenarios to consider for a dishonest card provider:

Scenario 1: A Card Provider may try manipulating transaction parameters, i.e. the value of $CDATA_{TranID}$, to gain monetary advantage over the approved payment amount.

Let the modified transaction parameters in Algorithm II be TransactionData' $\neq$ TransactionData, which will create the transaction ResultMessage passed onto TP as follows (note that TransactionData = TransactionInfo || SecureTransactionData):

ResultMessage $\leftarrow$ TransactionData' || CID || ApprovalID || TimeStamp

Based on Algorithm I, TP will have separately received SecureTransactionData as follows:

EncTranData $\leftarrow$ E(TransactionInfo || TransactionSecret), $CryptoKey_{TranID}$)

TranSignature $\leftarrow$ S((TransactionInfo || TransactionSecret), SE_{pr})

SecureTransactionData $\leftarrow$ (SecuredCryptoKey || EncTranData || TranSignature)

TP decrypts EncTranData in Step 6:

(TransactionInfo || TransactionSecret) $\leftarrow$ D(EncTranData, CryptoKey)

Upon comparison of the TransactionInfo parts of ResultMessage and the above decryption, transaction verification by TP will fail as TransactionInfo' $\neq$ TransactionInfo.

Scenario 2: A Card Provider may try replaying a previous successful transaction message of the same user to gain monetary advantage, such as replacing transactionData of a \$100 worth transaction request with the transactionData of a previous \$1 worth transaction, thereby collecting \$100 from its user while aiming to pay \$1 to the TP. The replayed transaction message will pass signature verification, but will be rejected by the Terminal Provider and also by SE because of the transaction counter mismatch.

2) DISHONEST TERMINAL PROVIDER

A Terminal Provider may try creating fake copies of approved transactions to gain advantage. This is simply not possible, as the Central Authority is the intermediary authority of all transactions approved by the Card Provider and transmitted to the Terminal Provider. Fake copies will be detected by the Central Authority during the settlement process.

A Terminal Provider may intentionally report failure to the terminal waiting for the transaction result, although it had received approval from the Card Provider. Thus, it can force the terminal application to start the transaction all over again and try claiming both transaction allowances from the Card Provider. However, once the Card Provider approves a transaction and successfully transmits this approval to the Terminal Provider over the Central Authority, it replies at the same time to the mobile application that requested the transaction and notifies the user about this approval. As a result of the fake failure report, the end user will see the approved transaction in his/her mobile application, whereas the terminal attendant will claim that the transaction is rejected. Therefore, this attack is trivially mitigated.

3) FAKE MOBILE APPLICATION

An unknown attacker may try creating a fake mobile application to get into the transaction flow. At the very first step, such an application will fail to register with the CP system, and will not be able to place any transaction requests. Thus, such an application can only target collecting valid transaction data from valid terminals to create fake transactions.

Let TD_A = TransactionInfoA || SecureTransactionDataA be the transaction data collected and saved by the attacker from transaction terminal A and assume that the attacker tries to send this data to the CP for a transaction at terminal B. Further, assume that TransactionInfoA = TransactionInfoB.

From Algorithm I, for the captured transaction:

TransactionSecretA $\leftarrow$ TranID || $RndSE_{TranID-A}$|| TranCnt

EncTranDataA $\leftarrow$ E((TransactionInfoA || TransactionSecretA), $CryptoKey_{TranID}$)

Whereas for the transaction at terminal B:

TransactionSecretB $\leftarrow$ TranID || $RndSE_{TranID-B}$|| TranCnt

EncTranDataB $\leftarrow$ E((TransactionInfoB || TransactionSecretB), $CryptoKey_{TranID}$)

In Step 6, TP for terminal B will decrypt EncTranDataB as follows:

(TransactionInfoB || TransactionSecretB) $\leftarrow$ D(EncTranDataB, CryptoKey)

TransactionSecretB will be compared with TransactionSecretA passed from the CP.

As $RndSE_{TranID-A} \neq RndSE_{TranID-B}$, due to uniqueness of nonce values generated by the SE, a mismatch will be detected by the TP. Consequently, the attack attempt will fail.

4) FAKE TERMINAL APPLICATION

A fake terminal application may try manipulating transaction parameters for its own advantage. Such a transaction may pass checks performed by the mobile application, Card Provider and Central Authority, as they can only see the public part of the transaction.

Let the modified transaction parameters in Algorithm II as provided by the terminal application be TransactionInfo' $\neq$ TransactionInfo, hence, TransactionData' $\neq$ TransactionData, which will create the Card Provider Approval ResultMessage as follows:

ResultMessage $\leftarrow$ TransactionData' || CID || ApprovalID || TimeStamp

Based on Algorithm I, TP will have separately received SecureTransactionData as follows:

EncTranData $\leftarrow$ E(TransactionInfo || TransactionSecret), $CryptoKey_{TranID}$)

TranSignature $\leftarrow$ S((TransactionInfo || TransactionSecret), SE_{pr})

SecureTransactionData $\leftarrow$ (SecuredCryptoKey || EncTranData || TranSignature)

TP decrypts EncTranData in Step 6:

(TransactionInfo || TransactionSecret) $\leftarrow$ D(EncTranData, CryptoKey)

Upon comparison of the TransactionInfo parts of ResultMessage and the above decryption, transaction verification by TP will fail as TransactionInfo' $\neq$ TransactionInfo. Consequently, TP will detect the mismatch between the public parameters and the decrypted message and reject the transaction.

5) TRANSACTION SNIFFING

As encrypted communication is mandatory in the proposed model, one cannot reveal any secret information with transaction sniffing. Public transaction information can be accessed by sniffing the communication between the terminal and the mobile device, but this will not result in security vulnerabilities or will not create a basis for fraud-attempts. Therefore, gaining advantage through transaction sniffing is simply not possible and at the same time this makes the blind replay attacks impossible.

6) DOUBLE-SPEND AND DOUBLE-PUSH

Each transaction in the proposed protocol has a unique transaction identifier that is part of the encrypted transaction data and transaction flow is encapsulated with strong

cryptographic proofs of the requester. Thus, the Card Provider and Terminal Provider make secure settlement through the Central Authority based on approved transactions and their cryptographic proofs. Double-spend and double-push attacks are not possible, as they will be detected during the settlement process between the providers.

VI. DISCUSSION

The proposed RONFC protocol offers numerous advantages over existing approaches for processing mobile transactions using the NFC technology. Some of these advantages can be summarized as follows:

- NFC-Enabler independence: RONFC gives full control to the users for mobile transactions by just requiring the use of the corresponding mobile application on NFC-enabled devices.
- No application limitation: In regular NFC, the SE on the NFC mobile device – whether SIM-based or embedded – has a limited space for the chip applets, which can typically hold only a few. For this reason, an NFC mobile device owner can only use the device for a limited number of applications. However, with the proposed design, the NFC mobile device can enable transactions through mobile applications and as a result, offer many more NFC-based applications. The use of mobile applications in RONFC also enables users to choose trusted application providers [38] for sensitive transactions.
- No card emulation issues: The Card Emulation Mode of NFC is an optional feature in NFC specifications; therefore, some of the NFC mobile devices do not feature card emulation at all. Consequently, creating an NFC-based transaction scheme based on a regular NFC SE results in system malfunctioning for some devices, even if they have NFC capability. However, the RONFC approach works with all the NFC-enabled mobile devices, as the Reader Mode is mandatory within NFC specifications.
- No TSM required: In the RONFC approach, the Card Providers can enable the NFC transactions themselves, simply by updating their mobile application. Also, the Terminal Providers have the ability to accept NFC transactions independently, simply by updating their terminals. As a result of this, to enable NFC transactions, NFC Enabler and their TSMs are not needed.
- Interoperability: Regular NFC transactions require proprietary integration schemes for each NFC Enabler or for their TSMs [39]. However, within our open protocol design, service (e.g. payment) providers only need to integrate with the Central Authority, which ensures the connection to the worldwide NFC ecosystem.
- On-device authentication: Entering PINs or other authentication means on an unknown device is a security concern for most people, whereas our design offers the possibility to authenticate users on their mobile device. Moreover, multiple authentication methods that are available on mobile devices (such as face recognition, fingerprint scanning etc.) can be supported.

A. APPLICATION TO CLOSED-CIRCUIT PAYMENT SYSTEMS

As mentioned above, mobile payment is the current most popular application of NFC for mobile transactions. If all the entities of a payment system are provided by the same system owner and the transaction goes through within the same provider's communication channels, then the payment system is referred to as a closed-circuit payment system.

Regular credit card payment systems are considered an open-circuit payment system, as a credit card issued by a bank may be used at any payment terminal around the world. The transaction is performed according to international standards and approval is given by the issuing bank at the end. Finally, the amount approved by the issuing bank is transferred to the bank account of the merchant who processed the payment. A typical example of closed-circuit payment system is a public transportation payment system. The transportation payment card is issued by the system operator and it can be used only within that city/system.

RONFC can easily be applied to close-circuit payment systems, with the Card Provider and Terminal Provider roles combined into one, obviating the need to have the Central Authority in between payment processing. This will also enable easy conversion into an open-circuit system if needed.

B. COMPARISON WITH LEGACY SOLUTIONS

To demonstrate the advantage of RONFC over legacy mobile transaction processing solutions, we discuss bootstrapping an end user's mobile device for a typical mobile payment scenario as follows: *XStores* has a mobile application that keeps user account, credit cards of the user that are issued by any bank, and *XPoints* that users collect from their purchases. XStores management decides to accept mobile payment at their stores, allowing users to be able to pay with the XStores mobile application either through one of the loaded credit cards or using XPoints.

The legacy approach will rely on accepting mobile payments through major mobile wallet providers (Apple™, Google™, Samsung™), given that XStores has payment terminals capable of processing payment through these mobile wallets. The very first problem XStores management will face is that major mobile wallet providers only allow adding cards to their wallets. Adding an application into the wallet to make a selection between cards or XPoints during transactions is not possible. Secondly, XStores will not be able to add credit cards of their users to their mobile wallets, as XStores is not the issuer of those cards. This leaves XStores management with only the option of adding XPoints account as a card in mobile wallets. In order to do so, XStores needs to create a card program that can be added in mobile wallets, implement remote issuance protocols of all mobile wallet providers, request XStores users to follow steps of adding their XPoints account as a card to their mobile devices and finally, XStores will start paying transaction fees to mobile wallet providers for each payment their users are performing using mobile devices.

With RONFC, XStores with payment terminals capable of processing RONFC transactions will only need to implement TP and CP roles, and they can start accepting mobile payments at their stores through the XStores mobile application, from all users and without an additional activation requirement. Users will be able to make a selection between credit cards or XPoints, or even split the payment between different forms. XStores does not need to exercise any financial or operational contract with mobile wallet providers and any mobile device having the XStores application becomes a mobile payment instrument.

VII. CONCLUSION

The developments in the NFC technology in the past decade have made it a popular choice for various types of mobile transactions, especially in the mobile payment industry. Today, there are hundreds of mobile device types that are NFC-enabled, and millions of people using these devices. A big portion of those devices can perform regular NFC transactions, if they contain a card issued by an NFC Enabler.

The legacy design of the NFC-based mobile transactions, where mobile network operators had a leading role, has resulted in high dependency on the NFC Enablers, creating a barrier for the widespread adoption of the technology, due to the wide range of integrations service providers need to develop to become NFC-ready.

In order to solve the enabler dependency problem, in this paper we proposed *RONFC*, an enabler-independent NFC transaction protocol, which offers numerous advantages compared to regular NFC transaction methods, by requiring the mobile device to operate only in the *Read* mode of NFC. The proposed protocol does not require any additional infrastructure/hardware elements on the transaction processing terminals, enabling easy adoption.

We developed a prototype of RONFC for a typical mobile contactless payment scenario and showed through performance experiments that the proposed approach is promising to meet the needs of real-world NFC-based mobile transactions, while satisfying their security requirements.

The proposed protocol is applicable for various fields that rely on NFC-based transactions, including secure access to physical facilities, mobile healthcare and insurance applications, mobile identity/access control solutions, among others. Future work will involve implementation of RONFC for other types of mobile transactions.

REFERENCES

[1] K. Nam, Z. Lee, and B. G. Lee, "How Internet has reshaped the user experience of banking service?" *KSII Trans. Internet Inf. Syst.*, vol. 10, no. 2, pp. 684–702, Feb. 2016. doi: 10.3837/tiis.2016.02.014.

[2] I. Turk and A. Cosar, "Internet connection sharing through NFC for connection loss problem in Internet-of-Things devices," in *Proc. Int. Conf. Next Gener. Wired/Wireless Netw.*, Aug. 2015, pp. 329–342. doi: 10.1007/978-3-319-23126-6_30.

[3] V. K. Raina, "NFC payment architecture," in *NFC Payment Systems and the New Era of Transaction Processing*. Hersley, PA, USA: IGI Global, 2017, ch. 2, pp. 43–73. doi: 10.4018/978-1-5225-2306-2.ch002.

[4] P. Andersson, J. Markendahl, and L. G. Mattsson, "Global policy networks' involvement in service innovation. Turning the mobile phone into a wallet by applying NFC techno-logy," *IMP J.*, vol. 5, no. 3, pp. 193–211, 2011.

[5] *NFC Forum Protocol Technical Specification*. Accessed: Feb. 9, 2019. [Online]. Available: http://www.nfc-forum.org

[6] B. Ozdenizci, K. Ok, and V. Coskun, "NFC loyal for enhancing loyalty services through near field communication," *Wireless Pers. Commun.*, vol. 68, no. 4, pp. 1923–1942, Feb. 2013. doi: 10.1007/s11277-012-0556-z.

[7] GlobalPlatform. *Introduction to Secure Elements*. Accessed: Feb. 9, 2019. [Online]. Available: https://globalplatform.org/wp-content/uploads/2018/05/Introduction-to-Secure-Element-15May2018.pdf

[8] V. Coskun, K. Ok, and B. Ozdenizci, *Near Field Communication (NFC): From Theory to Practice*, London, U.K.: Wiley, 2012.

[9] K. Mayes, and K. Markantonakis, *Smart Cards, Tokens, Security and Applications*. London, U.K.: Springer, 2008.

[10] L. Francis, G. Hancke, K. Mayes, and K. Markantonakis, "On the security issues of NFC enabled mobile phones," *Int. J. Internet Technol. Secured Trans.*, vol. 2, nos. 3–4, pp. 336–356, Dec. 2010. doi: 10.1504/IJITST.2010.037408.

[11] S. K. Staykova and J. Damsgaard, "The race to dominate the mobile payments platform: Entry and expansion strategies," *Electron. Commerce Res. Appl.*, vol. 14, no. 5, pp. 319–330, Sep./Oct. 2015. doi: 10.1016/j.elerap.2015.03.004.

[12] J. Pesonen and E. Hoster, "Near field communication technology in tourism," *Tourism Manage. Perspect.*, vol. 4, pp. 11–18, Oct. 2012.

[13] D. K. Biswas, M. Sinclair, J. Hyde, and I. Mahbub, "An NFC (near-field communication) based wireless power transfer system design with miniaturized receiver coil for optogenetic implants," in *Proc. Texas Symp. Wireless Microw. Circuits Syst. (WMCS)*, Waco, TX, USA, Apr. 2018, pp. 1–5.

[14] M. B. Renardi, K. Kuspriyanto, N. C. Basjaruddin, and A. Prafanto, "Baggage claim in airports using near field communication," *Indonesian J. Electr. Eng. Comput. Sci.*, vol. 7, no. 2, pp. 442–448, Aug. 2017.

[15] L. Dzafic and M. Ebner, "Game based learning through near field communication," in *Game-Based Learning Theory, Strategies and Performance Outcomes*, B. Youngkyun, Ed. Commack, NY, USA: Nova, 2017, pp. 295–322.

[16] K. Ok, V. Coskun, S. B. Yarman, C. Cevikbas, and B. Ozdenizci, "SIM-Sec: A key exchange protocol between SIM card and service provider," *Wireless Pers. Commun.*, vol. 89, no. 4, pp. 1371–1390, Aug. 2016. doi: 10.1007/s11277-016-3326-5.

[17] J. Domingo-Ferrer, J. Posegga, F. Sebé, and V. Torra, "Advances in smart cards," *Comput. Netw.*, vol. 51, no. 9, pp. 2219–2222, Jun. 2007. doi: 10.1016/j.comnet.2007.01.007.

[18] M. Hassinen, K. Hypponen, and K. Haataja, "An open, PKI-based mobile payment system," in *Proc. Emerg. Trends Inf. Commun. Secur.*, 2006, pp. 86–100. doi: 10.1007/11766155_7.

[19] A. Ruiz-Martínez, J. Sánchez-Montesinos, and D. Sánchez-Martínez, "A mobile network operator-independent mobile signature service," *J. Netw. Comput. Appl.*, vol. 34, no. 1, pp. 294–311, Jan. 2011. doi: 10.1016/j.jnca.2010.07.003.

[20] R. N. Akram, K. Markantonakis, and D. Sauveron, "A novel consumer-centric card management architecture and potential security issues," *Inf. Sci.*, vol. 321, pp. 150–161, Nov. 2015. doi: 10.1016/j.ins.2014.12.049.

[21] *GlobalPlatform A New Model: The Consumer-Centric Model and How it Applies to the Mobile Ecosystem*, Global Platform, San Mateo, CA, USA, 2012.

[22] P. Pourghomi, M. Q. Saeed, and G. Ghinea, "A secure cloud-based NFC mobile payment protocol," *Int. J. Adv. Comput. Sci. Appl.*, vol. 5, no. 10, pp. 24–31, 2014. doi: 10.14569/IJACSA.2014.051004.

[23] I. Turk and A. Cosar, "Having 4G, enabling cloud based execution for NFC based financial transactions," in *Proc. 11th Int. Conf. Innov. Inf. Technol. (IIT)*, Dubai, United Arab Emirates, Nov. 2015, pp. 63–67. doi: 10.1109/INNOVATIONS.2015.7381516.

[24] T.-M. Grønli, P. Pourghomi, and G. Ghinea, "Towards NFC payments using a lightweight architecture for the Web of Things," *Computing*, vol. 97, no. 10, pp. 985–999, Oct. 2015. doi: 10.1007/s00607-014-0397-6.

[25] H. Suryotrisongko, Sugiharsono and B. Setiawan, "A novel mobile payment scheme based on secure quick response payment with minimal infrastructure for cooperative enterprise in developing countries," *Procedia—Social Behav. Sci.*, vol. 65, pp. 906–912, Dec. 2012. doi: 10.1016/j.sbspro.2012.11.218.

[26] J. Liu, R. J. Kauffman, and D. Ma, "Competition, cooperation, and regulation: Understanding the evolution of the mobile payments technology ecosystem," *Electron. Commerce Res. Appl.*, vol. 14, no. 5, pp. 372–391, Sep./Oct. 2015. doi: 10.1016/j.elerap.2015.03.003.
[27] D. Pal, C. Vanijja, and B. Papasratorn, "An empirical analysis towards the adoption of NFC mobile payment system by the end user," *Procedia Comput. Sci.*, vol. 69, pp. 13–25, Jan. 2015. doi: 10.1016/j.procs.2015.10.002.
[28] S. Kumari, X. Li, F. Wu, A. K. Das, V. Odelu, and M. K. Khan, "A User Anonymous Mutual Authentication Protocol," *KSII Trans. Internet Inf. Syst.*, vol. 10, no. 9, pp. 4508–4528, Sep. 2016. doi: 10.3837/tiis.2016.09.026.
[29] *Identification Cards—Integrated Circuit Cards—Part 4*, document ISO 7816-4, 2013.
[30] Y. Lu, L. Li, H. Peng, and Y. Yang, "Robust ID based mutual authentication and key agreement scheme preserving user anonymity in mobile networks," *KSII Trans. Internet Inf. Syst.*, vol. 10, no. 3, pp. 1273–1288, Mar. 2016. doi: 10.3837/tiis.2016.03.018.
[31] J. D. Guttman, "State and progress in strand spaces: Proving fair exchange," *J. Automated Reasoning*, vol. 48, no. 2, pp. 159–195, Feb. 2012. doi: 10.1007/s10817-010-9202-1.
[32] *Elliptic Curve Digital Signature Algorithm*, document ANSI X9.62 and FIPS 186-2, 1998.
[33] *Currency Codes*, document ISO 4217, 2015.
[34] *Bouncy Castle Java API*. Accessed: Feb. 15, 2019. [Online]. Available: http://www.bouncycastle.org
[35] C. Morosan and A. DeFranco, "It's about time: Revisiting UTAUT2 to examine consumers' intentions to use NFC mobile payments in hotels," *Int. J. Hospitality Manage.*, vol. 53, pp. 17–29, Feb. 2016. doi: 10.1016/j.ijhm.2015.11.003.
[36] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "NIST special publication 800-57 recommendation for key management-part 1: General(revised)," NIST, Gaithersburg, MD, USA, Tech. Rep., 2007. doi: 10.6028/NIST.SP.800-57p1r2007.
[37] A. N. Khan, M. L. M. Kiah, S. U. Khan, and S. A. Madani, "Towards secure mobile cloud computing: A survey," *Future Gener. Comput. Syst.*, vol. 29, no. 5, pp. 1278–1299, Jul. 2013. doi: 10.1016/j.future.2012.08.003.
[38] A. Köster, C. Matt, and T. Hess, "Carefully choose your (payment) partner: How payment provider reputation influences m-commerce transactions," *Electron. Commerce Res. Appl.*, vol. 15, pp. 26–37, Jan./Feb. 2016. doi: 10.1016/j.elerap.2015.11.002.
[39] M. de Reuver, E. Verschuur, F. Nikayin, N. Cerpa, and H. Bouwman, "Collective action for mobile payment platforms: A case study on collaboration issues between banks and telecom operators," *Electron. Commerce Res. Appl.*, vol. 14, no. 5, pp. 331–344, Sep./Oct. 2015. doi: 10.1016/j.elerap.2014.08.004.

ISMAIL TURK received the B.S. degree from Bilkent University, in 2006, and the M.S. degree from Middle East Technical University, Ankara, Turkey, in 2009, both in computer technology and computer engineering. He is currently pursuing the Ph.D. degree in computer engineering with Middle East Technical University. He has also worked in payment industry companies for more than ten years, mainly responsible for NFC Payment and Identification Systems. He is currently a Principal Security Engineer with Symphony Communication Services, Los Angeles, CA, USA.

PELIN ANGIN received the B.S. degree in computer engineering from Bilkent University, Turkey, and the Ph.D. degree in computer science from Purdue University, West Lafayette, IN, USA, in 2013. She was a Visiting Assistant Professor (2014–2015) with Purdue University, and then as a Postdoctoral Research Associate, until 2016. She pursues her academic career with Middle East Technical University, Ankara, Turkey, since 2017, as an Assistant Professor of computer engineering. She is also affiliated with the S2RL and WINS research laboratories. Her research interests include the fields of distributed systems, cloud computing, and the IoT security.

AHMET COSAR received the B.S., M.S., and Ph.D. degrees from Middle East Technical University (METU), Bilkent University, and the University of Minnesota, respectively, all in computer engineering. He was a Faculty Member with the METU Computer Engineering Department, from 1996 to 2018. He is currently the Head of the Computer Engineering Department, University of Turkish Aeronautical Association. His research interests include distributed databases, data mining, e-commerce, and web-based software architectures. He has also worked as a Visiting Faculty Member with the University of Sharjah, UAE, and Manas University, Kyrgyzstan, and has also lectured at the American University of Central Asia.