Pricing Data Tampering in Automated Fare Collection with NFC-Equipped Smartphones

Fan Dang, Ennan Zhai, Zhenhua Li, *Member, IEEE*, Pengfei Zhou, Aziz Mohaisen, *Senior Member, IEEE*, Kaigui Bian, *Member, IEEE*, Qingfu Wen, and Mo Li, *Member, IEEE*

Abstract—Automated Fare Collection (AFC) systems have been globally deployed for decades, particularly in the public transportation network where the transit fee is calculated based on the length of the trip (*a.k.a.*, distance-based pricing AFC systems). Although most messages of AFC systems are insecurely transferred in plaintext, system operators did not pay much attention to this vulnerability, since the AFC network is basically isolated from the public network (e.g., the Internet)—there is no way of exploiting such a vulnerability from the outside of the AFC network. Nevertheless, in recent years, the advent of Near Field Communication (NFC)-equipped smartphones has opened up a channel to invade into the AFC network from the mobile Internet, i.e., by Host-based Card Emulation (HCE) over NFC-equipped smartphones. In this paper, we identify a novel paradigm of attacks, called *LessPay*, against modern distance-based pricing AFC systems, enabling users to pay much less than what they are supposed to be charged. The identified attack has two important properties: 1) it is invisible to AFC system operators because the attack never causes any inconsistency in the back-end database of the operators; and 2) it can be scalable to affect a large number of users (e.g., 10,000) by only requiring a moderate-sized AFC card pool (e.g., containing 150 cards). To evaluate the efficacy of the attack, we developed an HCE app to launch the LessPay attack; and the real-world experiments demonstrate not only the feasibility of the LessPay attack (with 97.6 percent success rate) but also its low cost in terms of bandwidth and computation. Finally, we propose, implement and evaluate four types of countermeasures, and present security analysis and comparison of these countermeasures on defending against the LessPay attack.

Index Terms—Automated fare collection (AFC), near field communication (NFC), host-based card emulation (HCE), security, vulnerability, attack, countermeasure

1 Introduction

Automated Fare Collection (AFC) systems have been globally deployed for decades to automate manual ticketing and charging systems, particularly in public transportation networks. As transit routes in modern cities are usually quite long, most of today's AFC systems adopt a distance-based pricing strategy, where the transit fee is calculated based on the length of the trip. To date, billions of AFC cards have been issued across the world.

A typical AFC system leverages a symmetric encryption method (e.g., based on 3DES [1] or AES algorithm [2]) to authenticate both the entities and messages involved. When an AFC card is officially issued, an unchangeable unique transaction key, TK, is written into the card, which will be used to generate a dynamic session key, SK; and a message authentication code (or MAC) [3] during the debit phase. Surprisingly, all the other data (e.g., the entrance or exit information used for calculating the trip fare) exchanged between AFC cards and terminals (i.e., fare-gates or fareboxes) are in the plaintext format, which is insecure [4], [5], [6], [7], [8]. The AFC system operators, nevertheless, do not need to worry about such a vulnerability, as the AFC network is well isolated from the public network (e.g., the Internet). Hence, it is quite difficult for any attacker to hack into the infrastructure of AFC systems from the outside of the AFC network in practice.

However, in recent years the advent of Near Field Communication (NFC)-equipped smartphones has bridged the gap between the AFC network and the Internet, thus putting AFC systems in a highly dangerous situation. Nowadays, the NFC module has become one of the default configurations of mainstream smartphones, such as iPhone and many Android phones. The NFC module operates at the same frequency (13.56 MHz) and implements the same standard (ISO/IEC 7816-4 and ISO/IEC 14443) as those in most AFC systems [9]. Moreover, it can work in a special *Host-based Card Emulation (HCE) mode* that allows any Android application to emulate an AFC card and talk directly to an AFC terminal.

In this paper, we identify a novel paradigm of attacks, called the *LessPay* attack, against modern distance-based pricing AFC systems. The goal of the attack is to pay less than actually required with a scalable method and does not recover

- *F. Dang, Z. Li, P. Zhou, and Q. Wen are with the School of Software, TNLIST, and KLISS MoE, Tsinghua University, Beijing 100084, China. E-mail: {dangf13, wqf15}@mails.tsinghua.edu.cn, {lizhenhua1983, zhoupf05}@tsinghua.edu.cn.*
- *E. Zhai is with the Department of Computer Science, Yale University, New Haven, CT 06520. E-mail: ennan.zhai@yale.edu.*
- *A. Mohaisen is with the Department of Computer Science, University of Central Florida, Orlando, FL 32816. E-mail: mohaisen@cs.ucf.edu.*
- *K. Bian is with the Department of Computer Science and Technology, Peking University, Beijing 100080, China. E-mail: bkg@pku.edu.cn.*
- *M. Li is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798. E-mail: limo@ntu.edu.sg.*

Manuscript received 18 July 2017; revised 25 May 2018; accepted 26 June 2018. Date of publication 5 July 2018; date of current version 1 Apr. 2019. (Corresponding author: Zhenhua Li.)
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TMC.2018.2853114

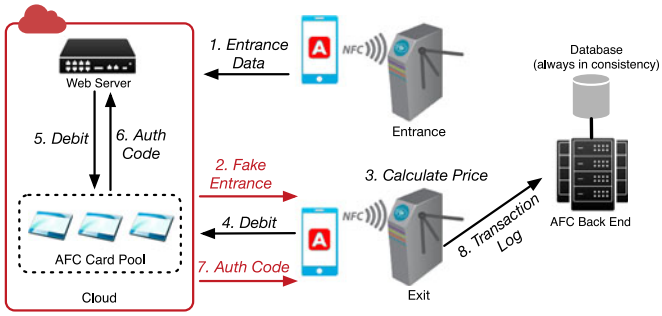


Fig. 1. Architectural overview of our designed attack on an AFC system. Red arrows denote the tampered messages, which however never cause inconsistency in the database of the AFC system.

the secret keys, which usually requires highly professional devices.[1] In order to launch such attacks, the attackers only need to have NFC-equipped smartphones and have installed *LessPay*—our developed HCE app for LessPay—on their smartphones. Fig. 1 presents a step-by-step workflow of the LessPay attack, which consists of two important phases: *tampering entrance data* (Step 1-2) and *relay attack on AFC card* (Step 4-7), which we briefly outline in the following.

- *Phase 1: Tampering entrance data.* As shown in Fig. 1, when a LessPay user wants to have a trip by metro, she first taps her smartphone on an entrance terminal. Then, the entrance terminal writes the entrance data into the AFC card emulated by LessPay, indicating the user's entrance station and timestamp. Subsequently, the entrance data is reported to the cloud of LessPay via a cellular connection (Step 1). After receiving the entrance data, the cloud periodically sends fake entrance data to the user (Step 2), in order to minimize the *expected* fare paid by her (note that the cloud does not know the user's destination). In practice, the period is configured as two minutes and the cellular traffic cost is within tens of KBs.
- *Phase 2: Relay attack on AFC card.* When the user reaches her destination, she taps her smartphone on an exit terminal, and the exit terminal calculates how much the user should pay for the trip according to the fake entrance data (Step 3). Afterward, the exit terminal sends a debit message to the emulated AFC card, which is instantly forwarded to the cloud of LessPay (Step 4). On the cloud side, this debit message is first relayed to the physical AFC card corresponding to the emulated AFC card (Step 5), and then the message authentication code (MAC) is relayed to the web server (Step 6). Finally, the web server returns the debit message together with MAC to LessPay to the smartphone (Step 7), and a transaction log is reported to the AFC back end by the exit terminal (Step 8). According to our measurement results, the round-trip time from Step 4 to Step 7 is generally within 100 ms, which is totally acceptable to user's real-world driving experience.

The key requirement of the LessPay attack is an *AFC card pool* that maintains a number of physical AFC cards for conducting relay attacks (i.e., Step 5 and Step 6 in Fig. 1). The success of relay attacks guarantees two important properties. First, AFC back end cannot detect any data inconsistency during the process of the attack, which means the attack is invisible to AFC system operators. In other words, for an AFC system operator, the debit & MAC provided by LessPay is *indistinguishable* from the ones offered by a legitimate AFC card. Second, as the web server (at the cloud side in Fig. 1) tampers both the station and timestamp information in the entrance data to forge a very short trip, we only need to maintain a relatively small number of cards in the pool to serve for a large number of users, e.g., 150 cards serving 10,000 users. This is because our users' very short fake trips can be easily scheduled by the cloud to totally avoid conflicts.

As a representative case study, we conducted real-world experiments to launch the LessPay attack against the City Traffic Card (CTC) system in City X, one of the major cities in China, with tens of millions population. Specifically, 100 users were recruited and each user randomly used LessPay to take a subway 40 times a month. During three-month experiments (from Jan. 10th to Apr. 10th, 2016) with a total of 12,000 tests, 97.6 percent tests passed (the failed tests are due to the poor quality of cellular connections). After the experiments, all cards in our card pool still work well. This shows the feasibility and scalability of the identified attack.

In order to defend against the LessPay attack, we propose four types of countermeasures corresponding to different protection capabilities and deployment overheads: 1) limiting frame waiting time, or FWT (Section 5.1); 2) protecting the entrance data (Section 5.2); 3) online fare calculation (Section 5.3); and 4) dynamic Quick Response (QR) codes (Section 5.4). We not only design and implement these countermeasures, but also evaluate and analyze the feasibility of these defenses in reality.

In summary, this paper makes the following contributions:

- We identify a real-world attack with NFC-equipped smartphones against the distance-based pricing policy in AFC systems [11], which enables users to pay much less than what they are supposed to be charged.
- We develop an HCE app to launch the LessPay attack (detailed in Section 3).
- We evaluate LessPay with real-world large-scale experiments, which not only demonstrate the feasibility of the attack (with 97.6 percent success rate) but also shows its low cost in terms of bandwidth and computation (detailed in Section 4).
- We propose four types of attack countermeasures, and discuss the feasibility and practicality of deploying these countermeasures (detailed in Section 5).

2 Overview of an AFC Transaction

This section presents an overview of the working principle of current AFC transactions, including stored file structure, entrance protocol, and exit protocol.

File Structure. Among today's AFC systems, the majority of AFC cards follow the ISO/IEC 14443 standard. In this standard, data in a smart card is stored in a very simple file system, organized in a hierarchical tree structure. Each file is identified by its unique file identifier. As an example, Fig. 2 shows the file structure of CTC. The basic card

1. Recovering secret keys is usually achieved via the *side-channel attack* [10] by exploiting extensive physical information like timing information or power consumption during the execution of cryptographic algorithms.

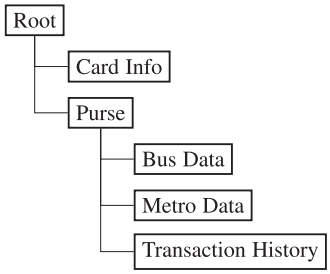


Fig. 2. Example: File structure of CTC.

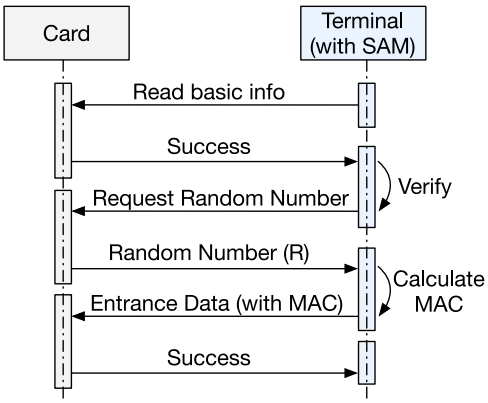


Fig. 3. The entrance protocol.

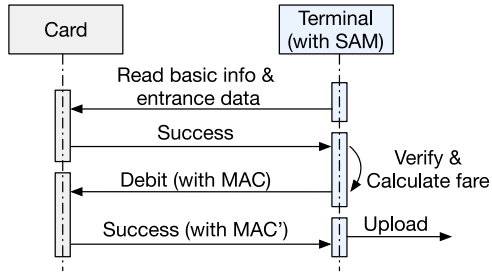


Fig. 4. The exit protocol.

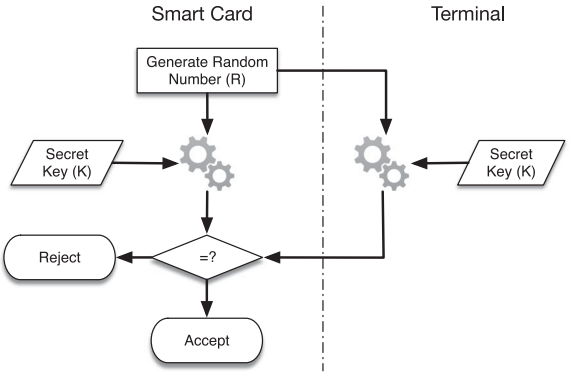


Fig. 5. External authentication, used by the card to validate the terminal.

information including card number, card type, and expiration is stored under the root directory. The data involved in the transactions of bus and metro is stored in the purse directory.

Entrance Protocol. When a passenger (with an AFC card) wants to enter a station, the AFC system needs to execute the entrance protocol, as shown in Fig. 3, based on the following three steps.

- First, the station's terminal requests and reads the basic information of this passenger's AFC card, including the card number, the expiration, and the balance. The terminal verifies this information, including checking the expiration and whether the balance is sufficient.
- Second, if the above verification succeeds, the terminal would try to write the entrance data to the `Metro Data` file (just using the metro as an example). However, before writing the entrance data, the AFC card needs to perform a one-way authentication to the terminal. As shown in Fig. 3, the terminal gets a random number R from the AFC card, and then calculates a MAC'[2] using R with a pre-installed key[3] shared with this AFC card (right-hand operations in Fig. 5).
- Finally, after generating MAC, the terminal sends the entrance data with the calculated MAC to the AFC card. The card performs an external authentication (shown in Fig. 5): if passed, the entrance data would be written on the card. On the other hand, the external authentication works as follows. As shown in Fig. 5 (left-hand), the AFC card first encrypts the random number R with the key shared with the terminal. Because the AFC card has received the terminal's MAC, which has been computed by encrypting the same random number R with the same key (the right-hand operation in Fig. 5), the AFC card can check whether the terminal's authentication passes through comparing the two ciphertexts. If the terminal is fake, the authentication fails.

After the whole procedure completes, the passenger will be allowed to enter the station, and her AFC card has been written her entrance information.

Exit Protocol. When the trip is finished, the passenger taps her card on the exit terminal. The terminal performs the exit protocol, which is shown in Fig. 4, based on the following two steps.

- First, the terminal reads the same basic information as the entrance stage, including the card number and the expiration, as well as the entrance data from the card. Then, the terminal verifies the above information. If the verification succeeds, the terminal calculates the fare that the passenger needs to pay. The verification process is the same as the first step in the entrance protocol.
- Second, in order to upload the transaction log information to the AFC back end, the card and terminal need to perform a mutual authentication with each other. In other words, besides the authentication to the terminal, in this step (called *debit checking* step), the terminal also needs to check whether the AFC card is emulated or fake. The process that the card authenticates the terminal is almost the same as the

2. A 2-key 3DES-MAC algorithm in CBC mode is used.

3. The key of each card is unique in practice. Instead of storing all keys (which is obviously impossible), the key of each card is generated using a root key and its card number. The root key is stored in a so-called SAM module attached to the terminal. The terminal uses SAM to generate the each-card key.

TABLE 1
Metro Entrance Data

#	Entrance Data	Enter Time	Metro Line	Station	Balance When Entering
1	1512051417043D014C1D	2015-12-05 14:17	4	Station A	75.00
2	1511301135020801B009	2015-11-30 11:35	2	Station B	24.80
3	15112215225E1D01AC0D	2015-11-22 15:22	X	Station C	35.00
4	15112009560A11016612	2015-11-20 09:56	10	Station D	47.10
5	15111220090401015203	2015-11-12 20:09	1	Station E	8.50

authentication step in the entrance protocol. On the contrary, i.e., the terminal authenticating the card, the AFC card needs to use its private transaction key TK to generate a session key SK and a MAC' (generated using the SK), and then sends them to the terminal for the authentication. The most important property in this step is: *a fake or emulated AFC card cannot have a transaction key to pass the authentication*.

After the mutual authentication, the terminal uploads the transaction information to its back-end database.

When AFC network was isolated from the Internet, the operators did not pay much attention to the communication between the card and the terminal. They only cared about the security of the card. Therefore, the debit procedure is protected by the secret key and the update of entrance data is protected by MAC, but the entrance protocol is still vulnerable due to the protection is one-way so that the data in the card is hard to tamper but easy to falsify seen from the terminal.

3 Attack Model and Implementation of LessPay

As shown in Fig. 1, there are six steps for launching the LessPay attack (i.e., Step 1-2 and Step 4-7). Step 3 and 8 do not belong to the attack since they occur on the terminal side and are not controlled by the attacker. Step 1-2 and Step 4-7 formulate two important phases in our attack: tampering entrance data and relay attack on AFC card. We next detail each of the phases.

3.1 Tampering Entrance Data

In order to tamper the entrance data, we need to know two important pieces of information: 1) the data structure of entrance data, and 2) the station data, e.g., GPS latitude and longitude coordinates. In this section, we describe a collection of approaches to infer the above information.

Collecting Entrance Data. Instead of collecting entrance data by physically accessing metro stations, we developed a lightweight app (different from LessPay app) to specifically collect data listed in Fig. 2. To allure users to download the app, the app itself provides useful features including parsing the balance and transaction histories (which metro line and when the user rode, as well as the fare) when the user taps the card on her NFC smartphone. We distributed this app in *Google Play*. With the agreement of our users, we collected the anonymized data (the card is innominate) from 97 different cards.

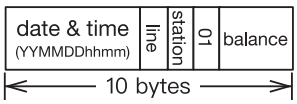


Fig. 6. Data structure of entrance data.

Obtaining Data Structure of Entrance Data. By collecting the entrance data, we analyze it and try to learn its structure. For example, Table 1 lists five items of our collected data. By observing and cross-checking the data, we find that the metro entrance data contains the following elements:

- The entrance time (`yyMMddhhmm` format, 5 bytes[4])
- The entrance metro line number (1 byte)
- The entrance station identifier (1 byte)
- The balance when entering the station (little endian in 2 bytes, e.g., `4C1D` represents `0x1D4C` (`7500`) cents)

Thus, we obtain the structure of entrance data shown in Fig. 6.

Obtaining Station Information. Rather than collecting station data by visiting each station (seems impossible), we found a third-party application called E-Card Tapper [12], which is able to parse the transaction histories as well as the trip records and details. Driven by this finding, we reversed this application using Apktool [13] and dumped the station data from the inner SQLite database of E-Card Tapper in order to extract its stored station information, such as the station identifier.

Besides this basic information on stations, we also need to infer the GPS latitude and longitude coordinates of each station. Thus, we get the location coordinates of stations using Google Maps.

Tampering the Entrance Data. We now already have enough information (i.e., entrance data structure and station information) to tamper the entrance data. In the LessPay implementation, as shown in Fig. 1, the web server in the cloud is responsible for generating the fake entrance data based on the above-collected data. To falsify a piece of valid entrance data, we simply prepare the legitimate entrance time, station information, and the balance. In order to *minimize* the fare, the attacker's cloud needs to generate the proper entrance data according to the destination. Details about the implementation of tampering the data are described in Section 3.3.

3.2 Relay Attack on AFC Card

This phase covers Step 4-7 shown in Fig. 1. During this phase, our purpose is to try to pass the mutual authentication in the exit protocol (mentioned in Section 2). This is because our emulated card receives a debit from the terminal, and the debit is protected by transaction key TK via the generated session key SK and MAC' (mentioned in Section 2). In practice, because a contactless smart card is a combination of MCU

4. Noted using patterns for formatting and parsing in JDK 1.8.

(microcontroller unit, like the most popular Intel 8051) and an radio frequency (RF) module, under the protection of the firmware in the MCU, the TK is not readable. Therefore, it is impossible to emulate an AFC card with debit support. In other words, the challenge in this phase is how we can get a transaction key TK for our emulated card to make it pass the mutual authentication.

We use the physical card equipped with TK to bypass this security check. This physical card is put in the cloud's AFC card pool (see Fig. 1), and it corresponds to the emulated card that receives the debit from the terminal. In other words, in LessPay, the emulated card should have a corresponding physical AFC card in the cloud-side card pool. Our intuition here is to make our emulated card act as a "proxy"-card and make the cloud server together with the physical card act as a "proxy"-reader. Such a design enables the emulated card to forward the debit command to the real card (i.e., the physical card) to generate MAC', because only that physical card has the needed transaction key TK.

During Step 4-7 in Fig. 1, the debit message transmitted by the terminal is first received by the "proxy"-card (i.e., the emulated card) and relayed to the cloud server. The cloud server will transmit the debit to a physical AFC card. Since the message is authenticated by MAC, the physical card will assume that it is communicating with a legitimate terminal and respond normally. Then, the response is forwarded to LessPay, which will respond to the terminal with the debit response. Still, the intact message is authenticated by MAC', which is identical to a real card, so the terminal cannot distinguish between the physical card and our emulated card.

Using such a relay attack, the attacker is able to overcome the fact that our emulated card lacks TK. Moreover, the valid MAC' will not cause any inconsistency.

3.3 Implementation

Based on the above two important design phases, the implementation of the LessPay attack consists of a front-end mobile app LessPay and a cloud-side service (i.e., the cloud in Fig. 1). The LessPay app requires an NFC-equipped smartphone with Android 4.4 or above. Regarding the cloud server side, any regular server or workstation is enough to meet the system requirement.

3.3.1 LessPay Client Implementation

Before HCE techniques are proposed, a secure element is required to perform the communication with the NFC terminal, and no Android application is involved in the transaction at all. Nevertheless, since Android 4.4 is released, it is possible to emulate a card using the HCE technology to emulate an ISO/IEC 14443 smart card without a secure element. Emulating an AFC card requires the following three features:

An Application ID (AID). When tapping the phone on a terminal, the HCE service is triggered by a `SELECT` command. This is identified by an AID. The AID of CTC is `1PAY.SYS.DDF01`, which we use to register our app.

An Emulated Card. An emulated ISO/IEC 14443 card needs to be implemented for communicating with the terminal. As we mentioned in Section 2, the data in a card is organized in files. The file structure of this emulated card is the same as the structure shown in Fig. 2. The messages transmitted and received between the card and the terminal are called application protocol data unit (APDU) [4]. The application-level protocol is half-duplex, by implementing a `processCommandApdu` method: the input is the command APDU that the reader sends and the output is the response APDU. The following commands in the standard are implemented in LessPay:

- `SELECT`: Select a different directory.
- `READ BINARY`: Read data from a specific file.
- `UPDATE BINARY`: Update data in a specific file. As we mentioned in Section 2, updating a file requires authentication. According to the standard, it is a one-way authentication that the card validates the terminal. In our attack model, we have to trust the terminal and ignore the MAC unconditionally. As a result, when the terminal gets a random number, we simply return a fixed one (see next item) and accept the MAC without any calculation and comparison.
- `GET RANDOM NUMBER`: We use a fixed number `00000000` instead of random numbers.
- `GET BALANCE`: Return the balance of the card. Note that since the card is reused by many users, therefore, the balance is fetched from the cloud when the app starts and it is updated periodically together with the fake entrance data.

The Relayed Part. The debit command is protected by TK and requires mutual authentication (as we mentioned in Section 3.2). Therefore, the debit command is relayed to the cloud server. We do not implement this command in an emulated card. We respond to the terminal whatever the cloud server returns.

In order to *minimize* the expected fare, we need to falsify the entrance data of the closest station. To achieve a better user experience, we will not ask the user her destination. Instead, we use the Android API to locate the user via the Cell-ID and Wi-Fi. We upload the user's location every two minutes. In each HTTP request, we send the user's coordinate and get the balance, the card number (see Fig. 4: card number is required to generate the TK, SK, and MAC), as well as the fake entrance data accordingly.

3.3.2 Cloud-Side Implementation

The configuration of the deployed server is: 2 $\times$ 4-core Xeon CPU E5-2609 @2.50 GHz, 8 GiB memory, 500 GiB 10K-RPM SAS disk, and a 100 Mbps network. The system on the cloud side is implemented in Akka, which is a JVM-based concurrent system.

Fake Entrance Generator. PostGIS [14], which is a spatial extender for PostgreSQL object-relational database, is used to find the nearest station. Since we are targeting a relatively small area and City X is not located in high latitude, we choose to use Cartesian distance to measure the distance rather than the spherical distance for a better performance.

"Proxy"-Card. We use ACR122u contactless smart card readers to communicate with the AFC cards. In the 100-user test, we prepared 5 readers and 5 *physical* cards. The server itself maintains the usage of different cards. We use an LRU dispatching algorithm to select a card from the cards that were not used in the past two minutes when receiving a request. Each card is set to the state `IN USE` for 2 minutes once we send the card number to the app. After a successful

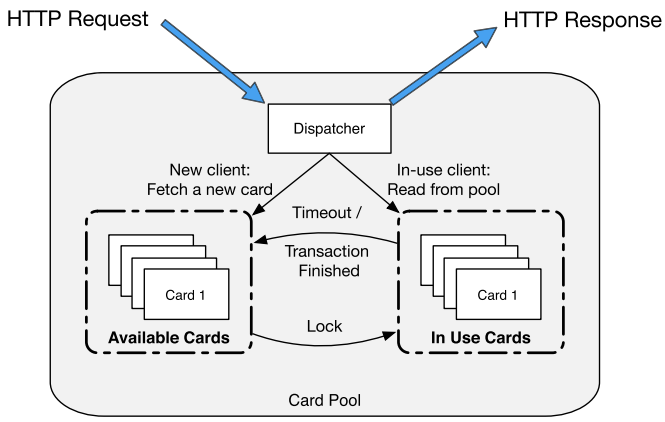


Fig. 7. Card pool scheduler.

transaction or timing out, the state is set to `AVAILABLE` again. The scheduler is shown in Fig. 7.

4 Performance Evaluation

This section evaluates LessPay through attacking real-world AFC systems in City X. In the evaluation, we aim to answer the following three questions:

- How much money users can "save" through using LessPay (in Section 4.2)?
- What is the overhead of using LessPay (in Section 4.3)?
- Whether LessPay can support a large number of users (in Section 4.4)?

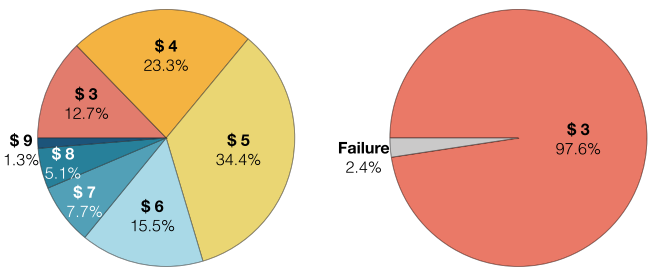
4.1 Experimental Setup

We recruited 100 volunteers to use LessPay. These users are equipped with HCE Android smartphones. The phone models we used are Samsung Galaxy S5, Huawei Mate 7, Moto XT1095, and LGE Nexus 5X. 62 users use LTE-TDD network, and the others use LTE-FDD network.

The experiment lasted for three months (from Jan. 10th to Apr. 10th, 2016). Each user was asked to use LessPay 40 times per month, with a total of 12,000 tests performed.

4.2 How Much We Can Save?

We now answer the first evaluation question: how much money users can "save"? The metro fares in City X vary from \$3 to \$9 (in the local currency) according to the distance. During the 12,000 tests, the "legitimate" fares are presented in Fig. 8a. The average fare that users should pay is \$5.03. By using the LessPay app, all users only need to pay \$3 instead of



(a) Users should pay the fares from \$3 to \$9.

(b) Except for 2.4% failures, users actually paid only \$3.

Fig. 8. The fares that users should pay and actually paid.

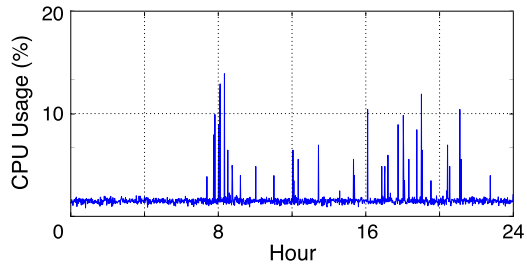


Fig. 9. CPU overhead of the cloud-side server.

the original fare (i.e., without using LessPay). This is clear using LessPay enables users to pay less than the users should pay. \$25,181 in total is "saved" (see Fig. 8b).

As shown in Fig. 8b, we also noticed that among these tests, there are 2.4 percent cases that do not succeed, which means these 2.4 percent attacks fail to "save" the money. According to the log, we found that the reason for attack failures is the poor network connection—the `DEBIT` command requires a relatively good quality connection.

4.3 System Overhead

We evaluate the overhead of LessPay based on two aspects: client-side overhead and cloud-side overhead. The former one means the overhead on smartphones, while the latter one means the overhead on the cloud server side.

Client-Side Overhead. The client-side overhead of LessPay comes from three sources: memory, network traffic, and battery usage. The typical memory usage is 20 MiB, which is moderate.

In terms of bandwidth overhead, our measured results show that the size of a single request is 48 bytes (16-byte location and 32-byte user token). The size of a single response is 20 bytes (6-byte card number, 4-byte balance, and 10-byte entrance data). Including TCP handshakes, and TCP/HTTP headers, the total network traffic cost is less than 1 KB. The cumulative distribution function (CDF) of network traffic consumed in these 12,000 tests are shown in Fig. 10. The average network traffic in all tests is 21.8 KB, which costs only cents. For 80 percent users, the network traffic cost is less than 36 KB. The average total traffic cost in a month (calculated over 40 trips) is less than 1 MB.

To understand the overhead of LessPay on battery life, we record the battery power consumption in these tests. As shown in Fig. 11, the average power consumption per trip is 3.4 mWh, which is extremely low given that the battery capacity of popular smartphones lies between 5-20 Wh [15].

Cloud-Side Overhead. Fig. 9 illustrates the CPU utilization of the server on a typical day. The web service is not a

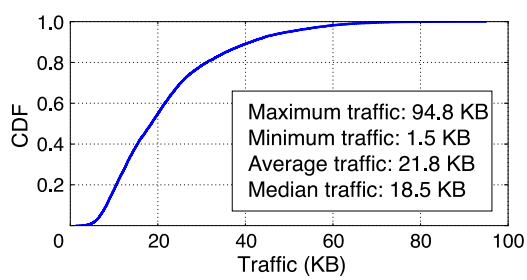


Fig. 10. The network traffic consumed in LessPay.

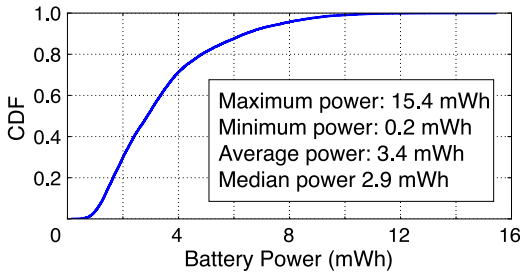


Fig. 11. The battery power consumed in LessPay.

CPU-bound application. In most time, the CPU usage is as low as 1 $\sim$ 2 percent. Even in rush hours (e.g., 7–9 A.M.), the CPU usage is below 15 percent.

The inbound/outbound bandwidth for the cloud-side server is quite low. There is no network traffic when no users turn the app on. As we pointed out, the traffic in each round-trip is less than 1 KB. As a result, network with 100 Mbps bandwidth is able to serve hundreds of thousands of users.

4.4 Scalability

We now explore whether LessPay can scale to a large number of users. The scalability of LessPay depends on the number of physical cards in the cloud-side card pool. In other words, more physical cards can make LessPay support more users. In order to evaluate the scalability of LessPay, we conducted a simulation study. The simulation assumes: 1) users use LessPay during rush hours, 2) all the users use LessPay within two hours, and 3) users' arrivals follow the Poisson distribution. The user can be denied service if she has to wait for longer than 15 seconds. We present the simulation results—the relationship between the number of users LessPay can support and the number of physical cards in the card pool—in Fig. 12. We also choose different service denial rates (0.1 and 0.2) to evaluate the scalability of LessPay under different environments. As shown in Fig. 12, even during rush hours, maintaining a card pool size of 150 will satisfy 10,000 users' need, which means LessPay can serve much more users by simply adding a few more cards to the pool. Thus, we conclude that LessPay scales well to a large number of users by only maintaining a moderate-sized AFC card pool at the cloud-side.

5 Countermeasures

In order to defend against the constructed relay attacks, this section proposes four types of countermeasures: 1) limiting frame waiting time, or FWT (Section 5.1); 2) protecting the entrance data (Section 5.2); 3) computing fare based on

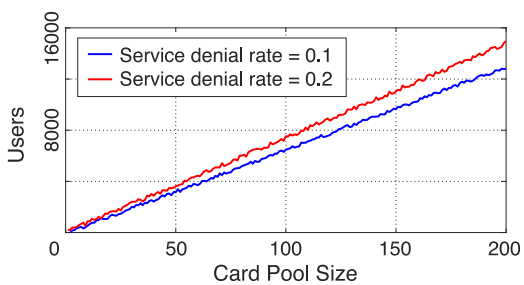


Fig. 12. The number of users that the card pool can support.

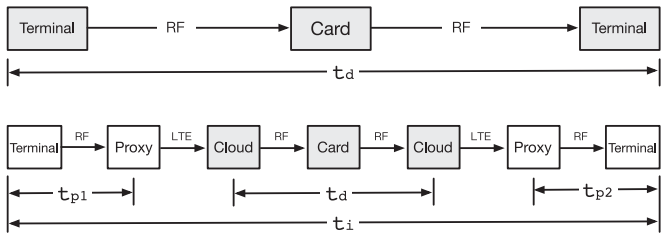


Fig. 13. The upper part shows a direct communication. The lower part shows a relayed communication.

online data-sharing (Section 5.3); and 4) dynamic QR code to replace AFC cards.

For each type of countermeasures, we first present the design and workflow of the countermeasure. Then, we describe the implementation, real-world deployment and experiment of the countermeasure. Finally, we discuss the countermeasure's advantages, disadvantages, and feasibility. Note that in the second type of countermeasure, we present four defenses corresponding to different levels of protection techniques: appending message authentication code to the entrance data (Section 5.2.1), encrypting the entrance data (Section 5.2.2), and ISO/IEC 7816-4 secure messaging (Section 5.2.3).

At the end of this section, we summarize the advantages and disadvantages of all the countermeasures (Section 5.5) and show the comparison results in Table 3.

5.1 Limiting Frame Waiting Time (FWT)

As shown in Fig. 1, we notice during the debit step, LessPay introduces an extra delay, which is caused by the additional wireless communication channel and HCE. In particular, Fig. 13 shows the communication difference between a normal case and a relay-attack case.

Driven by the insight shown in Fig. 13, we propose the first type of countermeasure that can quantify the delay in the debit step, thus detecting suspicious relay attacks. We call this countermeasure as limiting FWT approach.

Deployment and Experiments. We implement a limiting FWT-based approach which can measure the round-trip times of different commands using a commercial contactless card reader (HD-100) directly (i.e., t_d in Fig. 13), and indirectly (i.e., t_i in Fig. 13). Fig. 14 presents our experiments that measure the round-trip times (in milliseconds) of the four selected commands in our implementation. Specifically, these four commands include:

(1) `GET BALANCE` aims to read the balance stored in EEPROM. The response of this command is 6 bytes long.
(2) `LOAD` is responsible for computing a message authentication code. The response of this command is 18 bytes long.
(3) `INTERNAL AUTHENTICATION` is used to encrypt data through 2-key 3DES algorithm [16], and the response of this command is 10 bytes long. The reason that our experiments used 2-key 3DES algorithm (rather than AES) is that prevalent AFC systems keep using 2-key 3DES algorithm in practice due to historical reasons and deployment overhead. Given the fact that we are targeting currently in-use AFC

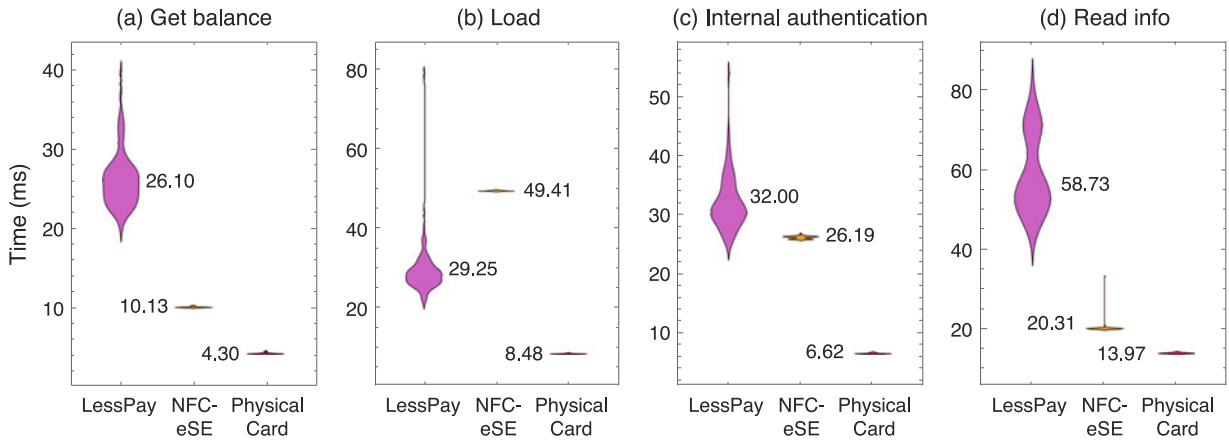


Fig. 14. RTT (in milliseconds) of different commands and targets.

systems, our experiments employ 2-key 3DES, despite the fact that AES works better than 3DES [17].

(4) `READ INFO`'s main purpose is to read the data stored in EEPROM. The response of this command is 62 bytes long.

In our experiments, we select a physical card, our LessPay app,[5] and a well-applied NFC-eSE based card for comparison. The physical card contains a microcontroller (MCU) and a crypto coprocessor. The NFC-eSE based card is embedded in Xiaomi MI 5, and the NFC part is supported by NXP PN66T, which employs NXP SmartMX as the Java Card VM. Our experiment result is collected from 100 tests. The density of distribution is recorded and presented based on the width of each RTT, and the time on the top-right of each plot is the average RTT in each case.

Based on the RTT of the relay (Fig. 14), we observe that the time consumed in LessPay is typically much longer when accessing EEPROM (i.e., command `GET BALANCE` in (a) and command `READ INFO` in (d)). However, when commands require cryptographic-related operations, the NFC-eSE based card also slows down. In the load command case, it takes more time than LessPay. We believe this is caused by the low performance of Java Card VM. As a result, it is possible to detect the attack by enforcing stricter timing restrictions. But by doing so, the legitimate NFC-eSE solution is also banned.

We also notice that deploying the FWT countermeasure may have some practical issues. First, a restrictive timeout value can lead to valid transactions rejected, especially with the NFC-eSE based cards. Second, for a standardized terminal, the maximum interval between the end of a frame sent by the terminal and the start of the response frame from the card, i.e., FWT, is determined by the card. The FWT is calculated from the frame waiting time integer (FWI) using the formula

$$FWT = \left(256 \times \frac{16}{f_c}\right) \times 2^{FWI},$$

where $f_c = 13.56$ MHz, varying from 302s (FWI $= 0$) to 4949 ms (FWI $= 14$). FWI is specified during initialization and anti-collision. Therefore, the response delay can be up to nearly 5 s for a standardized terminal, which makes it difficult to cope with the problem. Therefore, in this countermeasure, the terminal should not follow the FWT indicated by a card.

To evaluate the effectiveness of this countermeasure, we set up an experimental AFC environment. We employed the following equipment:

- LANDI APOS A8 POS terminal
- C-Union PSAM card
- Physical C-Union traffic card
- NFC-eSE C-Union traffic card

C-Union is a widely-applied standard in China, with hundreds of millions of cards issued. In the experiment, we choose different FWTs instead of following the FWT claimed by the card, then we conduct 100 purchases for the physical C-Union traffic card, the NFC-eSE C-Union traffic card, and LessPay, and measure the accept rate of each case. The complete purchase process is shown in Fig. 15, and the FWT is limited in Step 8 because in this step, the cryptographic algorithm is applied, while in other steps, LessPay is able to prereplay the required data and no relay is required.

The accept rates of various FWTs are shown in Fig. 16. Under the experimental condition, selecting a reasonable FWT is impossible. There is no significant boundary to distinguish LessPay and NFC-eSE. However, in metro stations, the RTT of LessPay is usually twice or triple of RTT in the

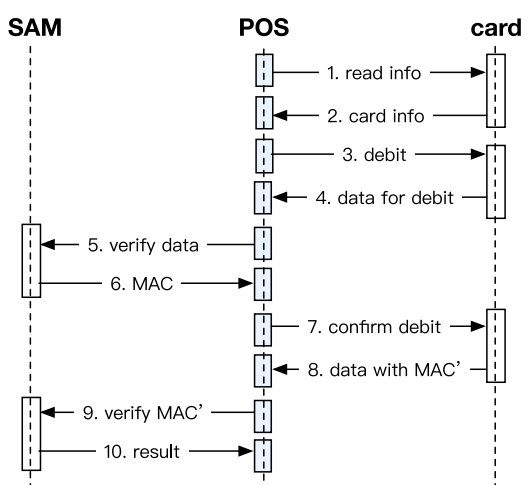


Fig. 15. The debit process of C-Union.

5. Under a good network condition; the network in the metro station is slower.

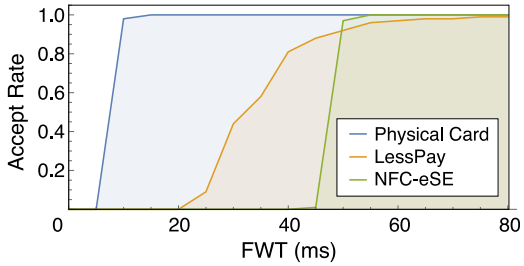


Fig. 16. Accept rates of various FWTs.

experimental environment, which varies in different stations. It seems that in metro stations, setting FWT to 60 ms is a reasonable choice. Unfortunately, due to the rapid change of network quality, this FWT may not apply in the future. To dynamically change the FWT, we may install cellular modules to measure the network quality. In particular, we measure the RTT of certain hosts (e.g., the DNS servers of the ISP) periodically, and set the FWT to the minimum value between the measured RTT and 60 ms (the upper bound of communication time using NFC-eSE).

Discussions. An obvious advantage of the limiting FWT approach is the approach can be directly used to detect relay attacks without changing AFC cards or AFC transaction protocol. In other words, it is easy to deploy in practice. However, as shown in our experiment results, such a countermeasure cannot work very well because some parameters of AFC terminals, e.g., restrictive timeout and time interval, significantly affect the accuracy.

5.2 Protecting the Entrance Data

Besides the communication delay during the debit phase, another major difference between the normal case and the relay attack case should be the entrance data. This is because LessPay cannot create an entrance data as an official terminal does. Inspired by the above observation, we propose the second group of countermeasure that protects the entrance data based on security techniques.

In this type of countermeasure, we present three subgroups corresponding to different levels of protection techniques: appending message authentication code to the entrance data (Section 5.2.1), encrypting the entrance data (Section 5.2.2), and ISO/IEC 7816-4 secure messaging (Section 5.2.3).

5.2.1 Appending MAC to the Entrance Data

To protect the entrance data, we first propose a solution that checks for the entrance data based on a message authentication code along with the entrance data. This countermeasure should include two important components.

- First, it is necessary to adopt one key for each card strategy through key derivation technique; otherwise, simply appending MAC cannot defend against relay attacks, because attackers can build a Peer-to-Peer network to share the valid entrance data together with the MACs from all the terminals. Specifically, in our one key for each card strategy, the derived key DK should be $DK = Encrypt(MK, SN)$, where MK is the master key stored in the terminal or the SAM, SN is the card number, and $Encrypt(.)$ is the 2-key 3DES encryption algorithm [18], [19]. DK is used as the key of the 2-key 3DES-MAC in CBC mode, which differs in different cards. The reason for using 2-key 3DES-MAC in CBC mode is to keep compatible with the old system so that the SAM can be applied directly. The process which conducts a SAM card is shown in Fig. 17.
- The second key component is the validation of entrance time (i.e., step 5 in Fig. 17). Note that the one key for each card strategy itself cannot prevent attackers from replaying the entrance data produced by themselves.

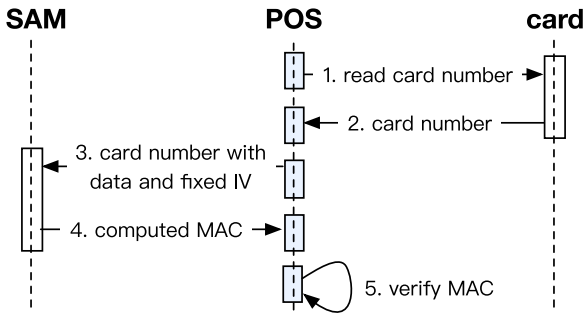


Fig. 17. The procedure of verifying MAC using SAM.

Deployment and Experiments. We have implemented the above countermeasure, i.e., appending MAC to the entrance data, based on two existing techniques, respectively, for comparison purpose.

(1) *MCU.* We conduct a widely-deployed MCU—STM32F103, which is ARM Cortex-M3 based and running at 72 MHz—to implement the 3DES-MAC algorithm.
(2) *SAM.* A secure access module is usually used to store secret keys and execute cryptographic algorithms. We conduct an Android-based POS terminal with a COTS SAM card and run the same test.

The experimental results are presented in Fig. 18. Because there is no operating system when using MCU, the running time of 8 bytes input data is only 0.928 ms (based on STM32 cryptographic library and `O1` compilation optimization), whose cost can be ignored (since a typical transaction costs several hundreds of milliseconds). While the average running time of 8 bytes input data is 59.36 ms, running on the Android-based terminal. It is much higher than the former test but still acceptable considering the transaction time.

We also notice that a practical limitation for this countermeasure is that MAC is stored statically. As a consequence, reading partial data, e.g., the entrance time, using `READ BINARY` command is possible but the terminal cannot determine whether it is modified.

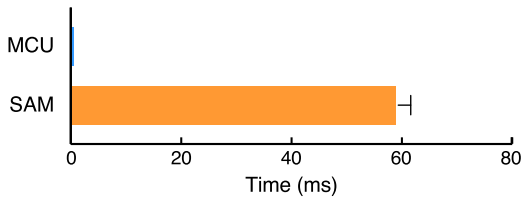


Fig. 18. Computation time of 3DES-MAC.

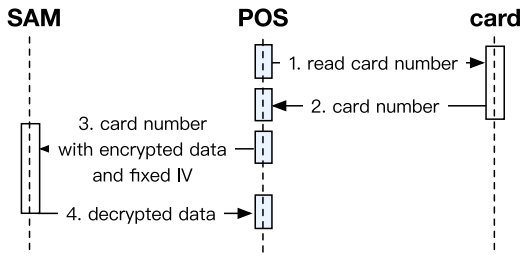


Fig. 19. The procedure of decrypting data using SAM.

Discussions. Appending MAC is effective against relay attacks, and does not need to change anything on the AFC cards or the back-end service. Despite the effectiveness, this countermeasure has the following two disadvantages. First, the AFC transaction protocol has to be changed, which will lead to a lot of additional efforts on the AFC terminal side. To save time and effort as much as possible, we suggest using the over-the-air (OTA) programming to upgrade the firmware remotely, which is adequate for applying the first two group of countermeasures. Second, because MAC is appended, additional computational overhead is added to the protocols, which introduces a slight delay.

5.2.2 Encrypting the Entrance Data

Similar to the approach of appending MAC, another way to protect the entrance data is to encrypt the entrance data. Fig. 19 shows an encryption-based approach (adopted in a SAM card) against relay attacks, which has no significant difference from MAC. The SAM in this countermeasure checks the information offered by AFC cards and sends the decrypted data to the POS terminal.

Deployment and Experiments. We implemented the encryption-based countermeasure based on 3DES-CBC scheme. We conducted the same experiments as what we did for the appending MAC approach in Section 5.2.1. The experiments use the same condition and configuration. Fig. 20 presents our experiment results.

There is no significant difference in cost between encrypting the entrance data and appending MAC to the entrance data. Both methods keep the proofs that indicate no modification is made since the whole entrance data is written. In this countermeasure, reading partial data is no longer possible, due to the property of 3DES-CBC scheme. In addition, encrypting data makes it more difficult to analyze the usage of data in different files. But it also sacrifices the possibility to parse data from the card without keys. After encrypting data, apps like E-Card Tapper can be only provided by the operator and used online due to data has to be decrypted remotely.

Discussions. An encryption-based approach is another effective countermeasure against relay attacks. However, given the fact that its principle is similar to MAC-based countermeasure, the disadvantages are the same. One of the disadvantages is this approach also needs to modify the protocol, potentially introducing additional labors. However, compared with MAC-based countermeasures, the encryption-based countermeasure should be more expensive, because most of the encryption operations may introduce more computational overhead than MAC computation.

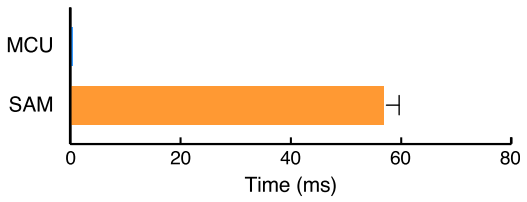


Fig. 20. Computation time of 3DES.

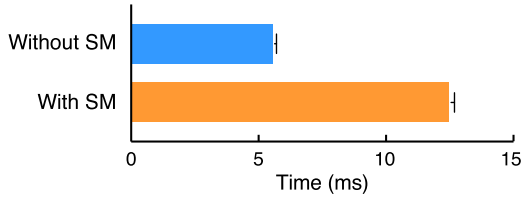


Fig. 21. Communication time (milliseconds) of reading 8 bytes data.

5.2.3 ISO/IEC 7816-4 Secure Messaging

ISO/IEC 7816-4 provides a type of mechanism for secure messaging. It allows encrypted data to be transmitted between the card and the terminal. The basic securing messaging is also a challenge-response procedure:

(1) The terminal sends a `GET RANDOM` command, and the card replies the random number *rnd*.
(2) The terminal sends a `READ BINARY` command, and the card replies data in ciphertext or plaintext (according to the card operating system) and uses *rnd* as initial vector (IV) to encrypt or calculate MAC.

Deployment and Experiments. We developed this countermeasure based on the above design, and measure the communication time of reading 8 bytes with and without secure messaging (SM) by 100 experimental runs. Fig. 21 shows the evaluation results.

The cost, which is only less than 10 ms, is quite small. However, it also faces the same threat we mentioned in Section 5.2.1 that attackers may replay the random number. As a result, the entrance time must be verified carefully. And the operator may still suffer from a potential attack. Furthermore, not only does applying this countermeasure require upgrading the terminals, but it also requires replacing or reprogramming the cards, which will cost millions of dollars.

Discussions. Compared with appending MAC and encrypting data, challenge-response authentication can provide dynamic data protection, which has been demonstrated more secure. However, this countermeasure needs to modify both AFC cards (reprogram or replace) and terminals, which need to a lot of efforts.

5.3 Computing Fare via Online Data-Sharing

Due to the poor network connection in the late 20th and early 21th centuries, the AFC network decided to use the offline solution. However, the success of contactless payment in financial systems like Visa payWave [20] and MasterCard Contactless [21] reveals that the network condition is no longer a problem.

In online fare calculation systems, when a user taps her card at the entrance terminal, the terminal simply verifies the card *offline*. Once the card contains a valid digital signature or MAC, the doors open and the entrance data will be

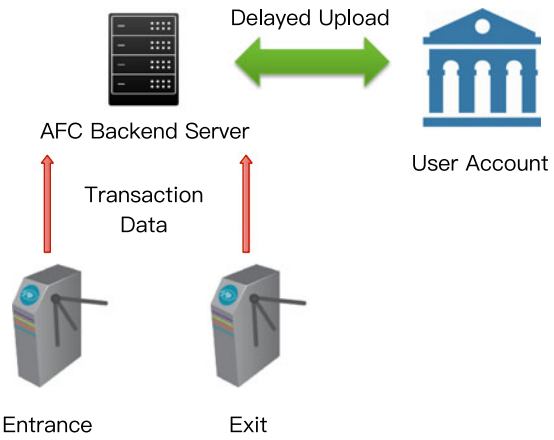


Fig. 22. Online fare calculation.

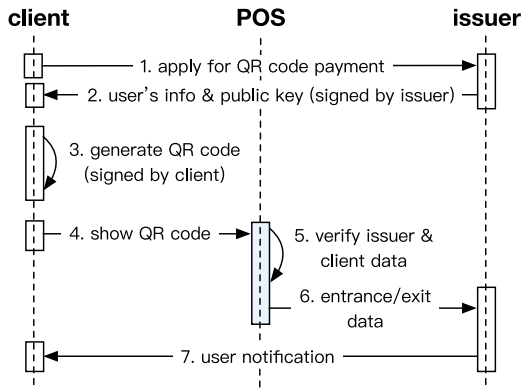


Fig. 23. The Quick Response (QR) code-based payment method without using an AFC card.

uploaded to AFC back end. When this user taps her card at the exit terminal, the terminal verifies the card again. If it passes the verification, the user will be billed then. The process is shown in Fig. 22. Because all the data (including entrance and exit information of the same AFC card) is uploaded and stored in the AFC back end, no external attackers can modify the information.

Except for its security, online fare calculation also has several advantages compared with traditional procedures:

- *No Top-Up Needed*. Since the fare is debited from users' accounts, the top-up is no longer needed.
- *Easy Applied Marketing Strategies*. Discounting for transfers and multiple rides is quite easy to apply because all of the fares are calculated after rides daily or even weekly.

One of the most typical examples of online fare calculation is in London. Contactless payments were launched on London's buses in 2012. And it costs 2 years to accept contactless payments in London transport. In London, users are billed at the end of the day instead of exiting stations.

Migrating the current AFC system to online fare calculation system may cost a huge amount of money. For example, each terminal costs $200 and a large city owns 20,000 buses, 150 metro stations. If there are 2 terminals on a bus and 40 terminals in a station, the total number of terminals is 46,000 and the total cost should be around 9.2 million US dollars.

In order to reduce the cost, we propose a procedure to utilize the currently in-use cards. Particularly, a user should associate a payment account with her AFC card, and the existing challenge-response mechanism can be used for authentication. Then the fare can be calculated later.

Besides, the AFC system may introduce contactless bank cards since they usually support offline data authentication using the Public Key Infrastructure (PKI) technology, and accordingly, the fare can be directly credited from users' bank cards.

Discussions. The online fare calculation can offer the highest level of security because the computation of this approach is performed on the cloud side. Thus, any external attacks, e.g., the attackers who want to perform relay attacks, cannot bypass the terminal checking with modified data. However, all the terminals and the back-end system need to be updated, which is a big cost.

5.4 Using Dynamic QR Codes Instead of AFC Cards

Motivated by the popularity of Quick Response code scanning payment in China, we propose a possible solution using dynamic QR code for public transportation, targeted at the AFC environment where almost everything works offline.

The designed process of using a QR code for public transportation is shown in Fig. 23: with no secure element involved, there is no such balance in the mobile app. Before using a QR code for public transportation, a user must provide his or her payment account to the issuer for the delayed payment. The issuer then generates a set of data including the identification of the user and the expiration of QR code. It also generates a pair of asymmetric keys (stored in the mobile phone for offline data signing). The issuer should sign these data (but the private key) to avoid being forged. We call them *online generated data*. Note that the signature is generated using the issuer's private key.

To support offline use, the client needs to generate the QR code containing *online generated data* along with the current time (for preventing replaying) and the signature of these data using the client's private key. Note that the client is always possible to generate new QR code offline before the expiration of *online generated data*. Once the POS terminal scans the QR code, it verifies the signature of *online generated data* by holding the public key of the issuer and the signature signed by the client using the pre-verified public key which is contained in the *online generated data*. After these steps, the terminal allows the user to enter or exit and then uploads the data of QR code to the server for further notification and payment.

The issuer can adjust the expiration to achieve the trade-off between the duration of offline usage and the potential risk.

Implementation and Experiments. As a totally different method, we have implemented this countermeasure on Android (Xiaomi 5 with Qualcomm Snapdragon 820 processor @ 1.8 GHz) side for generating QR code, MCU (STM32F103) side for verifying QR code, and server (in Section 3.3.2) side for generating secret keys. The complete data elements are listed in Table 2. We use Ed25519 [22] as the signing and verifying algorithm.

The QR code example is shown in Fig. 26, in which the expiration of the online generated data is set to 2018-2-15 08:00:00 (UTC), and the QR code generation time is set to

TABLE 2
Data Elements Contained in the QR Code

	#	Element	Length	Description
Online generated data	1	User ID	8	Identification of user
	2	Expiration	4	Expiration of online generated data
	3	User's public key	32	Different in each generation
	4	Signature	64	Signature of online generated data (element 1-3, using issuer's private key)
Offline generated data	5	Timestamp	4	Timestamp of QR code creation
	6	Signature	64	Signature of the entire data elements (using user's private key)

2018-2-14 22:00:00 (UTC). To verify the QR code, the issuer's public key (`0FD7E339ED16FEE6CAC84E300B01E4-F39AAC962E1E68443545E119CEBF8E6103`) is installed in the MCU. It first verifies `signature1` using this public key. Once the verification is passed, the public key (#3) of the user can be retrieved, and consequently, `signature2` can be verified.

According to the design, Step 1 is only needed for the first time a user registers; Step 2 shows up occasionally (new *online generated data* will be generated only after its expiration); Step 3-7 exist in every usage. To measure the performance of this design, we conducted 100 tests. In these tests, the client downloads *online generated data* every time (which may happen in low frequency uses), and then generates the corresponding QR code. We use a commercial QR code decoder that is connected to the MCU via serial port, and use the decoder to scan and decode the QR code on the mobile phone. Finally, the QR code is verified in the MCU and it notifies our server via an ethernet module. The result is shown in Figs. 24 and 25, which indicates that it is a relatively traffic-saving solution. Note that the time consumed in Step 4 is not included in Fig. 24 because it depends on the position of how we show the screen to the terminal. For daily use, Step 2 is not involved. Therefore, we also measure the running time of signing data (Step 3) and verifying data (Step 5). During the 100 tests, the signing time on the mobile phone is less than 1 ms. The verifying time on the MCU is, however, as high as 856 ms, which indicates that such low-performance MCU should not be used to perform ECC calculation.

Discussions. Using dynamic QR code, which achieves the lowest cost on the passenger side, entirely replaces the medium of AFC systems. However, all the terminals and the back-end system need to be upgraded. Besides, there are also several disadvantages: 1) the user must open app manually, which may reduce the speed of entrance or exit; 2) there is no secure element involved, therefore writing a secure app becomes a challenge.

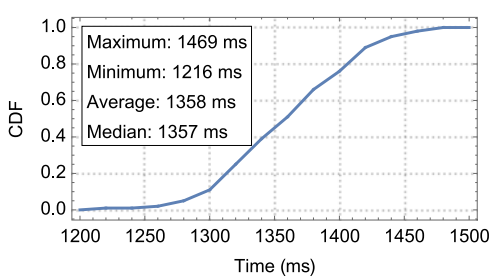


Fig. 24. Time consumed in QR code-based payment.

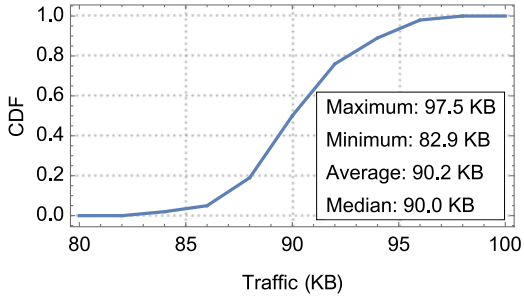


Fig. 25. Network traffic consumed in QR code-based payment.

5.5 Summary

Table 3 presents a summary of the six countermeasures (in four groups) proposed in this section. As the only one countermeasure that does not need to replace both AFC cards and protocols, limiting FWT approach offers the lowest effectiveness. On the contrary, the online fare calculation approach and the dynamic QR code approach provide the best effectiveness but need to upgrade lots of components, which leads to big additional efforts. Between the two extreme cases, using MAC, encryption and secure messing can also provide reasonable defense against relay attacks, and at the same time introduce additional overhead and protocol modification.

To sum up, which countermeasure can be used in practice relies on specific scenarios and purposes at hand. We believe our proposed six countermeasures have provided enough choice spaces for defending against the LessPay attack.

6 Related Work

This section reviews previous studies on relay attack and attacks on contactless payment, smart card, and AFC system.

Relay Attack. Attackers have been trying to implement a relay attack using various approaches. Initially, researchers built specific hardware to relay the communication between a smart card and a terminal. Hancke et al. [23] used a self-built hardware to increase the distance up to 50 m. They also deeply reviewed relay attacks in [24], discussing relay resistant mechanisms.

With the development of NFC, recent works have focused on relay attacks using mobile phones. Nokia 6131

0000000012345678**(user ID)**5A853E00**(expiration)**062C6
C7336106367DC89A5093B5C2855BF8EDA16F2168C913CBBF66
86207CCB0**(public key)**A5424E8D2F12A4AE4AF536D8E8C9E
484FC88745A30FB49B13E9DA22C79796FC78E3F9705092B2F5
78483BFA67356210D045C25B409EBD3833FF80604E18CE701
(signature1)5A84B160**(generated time)**39C32E866B06AA
9F5A80FA35A1F55C25D48F0B2DE51F3447A3D04C462E41857B
F93598BFC1A5E4F721005A05CD91057305ABEA2675CF4BA0FC
97A5FD8F630B08**(signature2)**

Fig. 26. An example of the QR code generated by our implementation, as well as its implications.

TABLE 3
Summary of Countermeasures

#	Countermeasures	Pros.	Cons.	Effectiveness	Cost
1	Limiting FWT	No replacement of cards required. The reading/writing protocol keeps unchanged.	Ticket tokens vary from types, which makes it hard to determine FWT.	Low	Low
2.1	Appending MAC	No replacement of cards required. Apps that can parse data (e.g., transaction records) still work.	MAC can be replayed. Protocol must be modified due to the extra verification.	Medium	Medium
2.2	Encrypting data	No replacement of cards required. Hard to replay.	Protocol must be modified because of the extra verification. A proprietary app is required to parse data.	Medium	Medium
2.3	Secure messaging	Challenge-response based authentication. Dynamic data protection.	Protocol must be modified because of the extra verification. Cards need to be replaced or reprogrammed. Random number can be replayed.	Medium	Medium to high
3	Online fare calculation	Impossible to falsify data. Operators may run various sales strategy.	Hard to ensure high network coverage.	High	High
4	Dynamic QR code	Impossible to falsify data. Low cost on the passenger side.	Terminals and back-end service need to be upgraded.	High	High

was the first phone ever produced with NFC capability. Francis et al. [25] revealed the possibility to perform a relay attack using COTS devices. In [25], [26], [27], researchers performed relay attacks using Nokia mobile phones and discussed the feasibility of some countermeasures, such as timing and distance bounding.

More recently, researchers focused on relay attacks with Android mobile phones. Roland et al. [28], [29] described relay attack equipment and procedures on Android phones. Lee [30] demonstrated an open-source software NFCProxy that proxies transactions using Android phones. Korak [31] compared timing on relay attacks using different communication channels. Still, some other work relates to privacy or human interaction issues [32], [33], [34].

Contactless Payment. Extracting information from the transaction communication between a credit card and a POS terminal using eavesdropping is possible. Haselsteiner and Beitfuß [35] showed a possible way to eavesdrop NFC. They suggested that, while normal communication distances for NFC are up to 10 cm, eavesdropping is possible even if there is a distance of several meters between the attacker and the attacked devices. However, this information (mainly credit card numbers, and expiration) can be obtained directly via NFC or even through social engineering. Paget [36] showed the process and later encode this information and wrote to magnetic stripe cards. This attack is also known as downgrade attack, which may not apply nowadays, due to banks refusing magnetic stripe cards and migrating to *Chip and PIN*.

Smart Card and AFC System. Originally, the MIFARE chip, which is a memory card chip, was developed as a solution for AFC. In 1994, an AFC system based on MIFARE was first deployed in Oslo, Norway. Ten years after its introduction, the MIFARE Classic was seen as the major candidate for AFC systems. In 2008, however, researchers discovered a serious security flaw in MIFARE Classic cards [37], [38], [39]. In particular, the cipher algorithm used in MIFARE Classic, known as CRYPTO1, has been reversed and reconstructed in detail, and a relatively easy method to retrieve cryptographic keys was revealed. Since then, the AFC cards have been gradually replaced by processor cards globally.

According to a public report, in Dec. 2010, two engineers from Qihoo exploited the flaw of MIFARE Classic chip to crack four Beijing Municipal Administration Traffic Cards and modified the balances. Ref. [40] Beijing had stopped issuing the MIFARE Classic card since then. The newly issuing cards are processor cards, which are the cards we used in our attack model.

7 Conclusions

Today's AFC systems have been globally adopted and billions of AFC cards have been issued all over the world. Among these systems, ISO/IEC 14443 is the main protocol

used worldwide, being near universal in East Asia and Europe, and in its early adoption in the rest of the world.

Under the above background, this paper proposes a new relay attack on AFC systems, which enables users to pay much less than actually required by providing fake entrance data. The relay attack is scalable and invisible to AFC system operators. We have developed an HCE app, named LessPay, based on our proposed and reported attack, and evaluated the LessPay app through real-world experiments. The evaluation results demonstrate the feasibility, practicality and scalability of our approach.

To handle the constructed relay attack, we finally propose four types of countermeasures against the constructed relay attack. We implement, deploy and evaluate these countermeasures, and also provide the analysis of these approaches.

Acknowledgments

This work is supported by the National Key R&D Program of China under grant 2018YFB1004700, the High-Tech R&D Program of China ("863–China Cloud" Major Program) under grant 2015AA01A201, the National Natural Science Foundation of China (NSFC) under grants 61471217, 61432002 and 61632020. M. Li is supported by the Singapore MOE Tier-1 grant RG125/17, Tier-2 grant MOE2016-T2-2-023, and NTU CoE grant M4081879. Aziz Mohaisen is supported by NSF under grant CNS-1643207 and NRF under grant NRF-2016K1A1A2912757.

References

[1] E. Barker and N. Mouha, *Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher*. Gaithersburg, MD, USA: United States National Institute of Standards and Technology, 2017.

[2] NIST-FIPS, *Announcing the Advanced Encryption Standard (AES)*. Gaithersburg, MD, USA: United States National Institute of Standards and Technology, 2001.

[3] *Information Technology–Security Techniques–Message Authentication Codes (MACs) Part 1: Mechanisms Using a Block Cipher*, ISO/IEC Standard 9797-1:2011.

[4] *Identification Cards Integrated Circuit Cards - Part 4: Organization, Security and Commands for Interchange*, ISO/IEC Standard 7816-4:2013.

[5] *City Union Card of Digital City–General Technology Requirements*, China Standard GB/T 31 778-2015.

[6] *Specification for Contactless ePurse Application (CEPAS)*, Singapore Standard SS 518:2014.

[7] *Contactless Pre-Paid/Post Pay IC Card User Card*, Korea Standard KS X 6924:2009.

[8] *CIPURSE V2 - Operation and Interface Specification*, OSPT Standard 2.0.

[9] W. Rankl and W. Effing, *Smart Card Handbook*. Hoboken, NJ, USA: Wiley, 2010.

[10] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Examining smart-card security under the threat of power analysis attacks," *IEEE Trans. Comput.*, vol. 51, no. 5, pp. 541–552, May 2002.

[11] F. Dang, P. Zhou, Z. Li, E. Zhai, A. Mohaisen, Q. Wen, and M. Li, "Large-scale invisible attack on AFC systems with NFC-equipped smartphones," in *Proc. IEEE INFOCOM*, 2017, pp. 1–9.

[12] E-Card Tapper. [Online]. Available: http://www.wandoujia.com/apps/com.siodata.uplink, Accessed on: Jul. 20, 2016.

[13] Apktool—A tool for reverse engineering android APK files. [Online]. Available: https://ibotpeaches.github.io/Apktool/, Accessed on: Jul. 21, 2016.

[14] PostGIS–Spatial and Geographic Objects for PostgreSQL. [Online]. Available: http://postgis.net/, Accessed on: Jul. 21, 2016.

[15] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proc. USENIX Conf. USENIX Annu. Tech. Conf.*, 2010, pp. 21–21.

[16] C. J. Mitchell, "On the security of 2-key triple DES," *IEEE Trans. Inf. Theory*, vol. 62, no. 11, pp. 6260–6267, Nov. 2016.

[17] E. Barker, *Recommendation Key Manag. Part 1: General*. Gaithersburg, MD, USA: United States National Institute of Standards and Technology, 2016.

[18] P. Karn, P. Metzger, and W. Simpson, *The ESP triple DES transform, RFC 1851*, (1995). [Online]. Available: http://tools.ietf.org/html/rfc1851

[19] EMVCo, *Integrated Circuit Card Specifications for Payment Systems-Book 2: Security and Key Management*, (2011). [Online]. Available: https://www.emvco.com/document-search/

[20] Visa payWave. [Online]. Available: https://usa.visa.com/pay-with-visa/featured-technologies/visa-paywave.html, Accessed on: Jul. 20, 2016.

[21] MasterCard Contactless. [Online]. Available: http://www.mastercard.com/contactless/, Accessed on: Jul. 21, 2016.

[22] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," *J. Cryptographic Eng.*, vol. 2, no. 2, pp. 77–89, Sep. 2012.

[23] G. P. Hancke, "A practical relay attack on ISO 14443 proximity cards," Comput. Lab., Univ. of Cambridge, Cambridge, UK, Tech. Rep., vol. 59, pp. 382–385, 2005.

[24] G. P. Hancke, K. E. Mayes, and K. Markantonakis, "Confidence in smart token proximity: Relay attacks revisited," *Comput. Secur.*, vol. 28, no. 7, pp. 615–627, Oct. 2009.

[25] L. Francis, G. Hancke, K. Mayes, and K. Markantonakis, "Practical NFC peer-to-peer relay attack using mobile phones," in *Proc. Int. Workshop Radio Freq. Identification: Secur. Privacy Issues*, 2010, pp. 35–49.

[26] L. Francis, G. Hancke, K. Mayes, and K. Markantonakis, "Practical relay attack on contactless transactions by using NFC mobile phones," in *Proc. Workshop RFID IoT Secur. Asia*, Nov. 2012, pp. 21–32.

[27] R. Verdult and F. Kooman, "Practical attacks on NFC enabled cell phones," in *Proc. Int. Workshop Near Field Commun.*, Feb. 2011, pp. 77–82.

[28] M. Roland, J. Langer, and J. Scharinger, "Relay attacks on secure element-enabled mobile devices," in *Proc. IFIP Int. Inf. Secur. Conf.*, 2012, pp. 1–12.

[29] M. Roland, J. Langer, and J. Scharinger, "Applying relay attacks to Google wallet," in *Proc. Int. Workshop Near Field Commun.*, Feb. 2013, pp. 1–6.

[30] E. Lee, "NFC hacking: The easy way," presented at DEFCON 20 Hacking Conf., Las Vegas, USA, 2012.

[31] T. Korak and M. Hutter, "On the power of active relay attacks using custom-made proxies," in *Proc. IEEE Int. Conf. RFID*, Apr. 2014, pp. 126–133.

[32] F. Dang, P. Zhou, Z. Li, and Y. Liu, "NFC-enabled attack on cyber physical systems: A practical case study," in *Proc. IEEE INFOCOM Workshops*, 2017, pp. 289–294.

[33] W. Gu, L. Shangguan, Z. Yang, and Y. Liu, "Sleep hunter: Towards fine grained sleep stage tracking with smartphones," *IEEE Trans. Mobile Comput.*, vol. 15, no. 6, pp. 1514–1527, Jun. 2016.

[34] X. Chen, X. Wu, X. Y. Li, X. Ji, Y. He, and Y. Liu, "Privacy-aware high-quality map generation with participatory sensing," *IEEE Trans. Mobile Comput.*, vol. 15, no. 3, pp. 719–732, Mar. 2016.

[35] E. Haselsteiner and K. Breitfuß, "Security in near field communication (NFC): Strengths and weaknesses," in *Proc. Int. Workshop Radio Freq. Identification: Secur. Privacy Issues*, 2006, pp. 12–14.

[36] K. Paget, "Credit card fraud—The contactless generation," in *ShmooCon*, 2012. [Online]. Available: http://www.tombom.co.uk/Paget-shmoocon-credit-cards.pdf

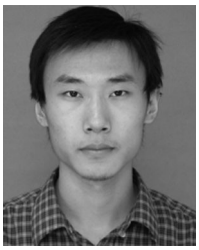
[37] F. D. Garcia, G. de Koning Gans, R. Muijrers, P. van Rossum, R. Verdult, R. W. Schreur, and B. Jacobs, "Dismantling MIFARE Classic," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2008, pp. 97–114.

[38] G. de Koning Gans, J.-H. Hoepman, and F. D. Garcia, "A practical attack on the MIFARE classic," in *Proc. Int. Conf. Smart Card Res. Adv. Appl.*, 2008, pp. 267–282.

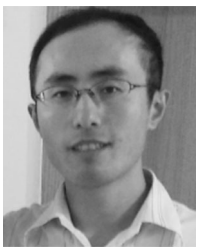
[39] N. Courtois, K. Nohl, and S. O'Neil, "Algebraic attacks on the Crypto-1 stream cipher in MiFare classic and oyster cards," *IACR Cryptology ePrint Archive*, vol. 2008, 2008, Art. no. 166.

[40] The engineers in Qihoo 360 cracked BMAC. [Online]. Available: http://tech.sina.com.cn/i/2011-09-28/17166123872.shtml, Accessed on: Jul. 20, 2016.

Fan Dang received the BE degree from the School of Software, Tsinghua University, in 2013. He is currently working toward the PhD degree in the School of Software, Tsinghua University. His research interests include mobile computing and security.



Ennan Zhai received the MPhil and PhD degrees from Yale University, in 2014 and 2015, respectively. He is currently an associate research scientist with the Computer Science Department, Yale University. His research interests mainly include distributed system, applied cryptography, and software verification.



Zhenhua Li received the BSc and MSc degrees from Nanjing University, in 2005 and 2008, and the PhD degree from Peking University, in 2013, all in computer science and technology. He is an assistant professor with the School of Software, Tsinghua University. His research areas cover cloud computing/storage/download, big data analysis, content distribution, and mobile Internet. He is a member of the IEEE.

Pengfei Zhou received the BE degree from the Automation Department, Tsinghua University, in 2009 and the PhD degree from Nanyang Technological University. He is currently working as a postdoctoral fellow with Tsinghua University. His current research interests include mobile computing and security, localization, cellular network communications, and NFC.

Aziz Mohaisen (M'05-SM'15) received the PhD degree from the University of Minnesota, in 2012. He is currently an associate professor with the Department of Computer Science and the Department of Electrical and Computer Engineering, University of Central Florida. His research interests include the areas of systems, security, privacy, and measurements. He is a senior member of the IEEE.

Kaigui Bian received the PhD degree in computer engineering from Virginia Tech, Blacksburg, in 2011. He is currently an associate professor in the Institute of Network Computing and Information Systems, School of EECS, Peking University. His research interests include mobile computing, cognitive radio networks, network security, and privacy. He is a member of the IEEE.

Qingfu Wen received the BE degree from the School of Software, Tsinghua University, in 2015. He is currently working toward the MSc degree in the School of Software, Tsinghua University. His research interests include big data and mobile computing.

Mo Li (M'06) received the BS degree from the Department of Computer Science and Technology, Tsinghua University, in 2004, and the PhD degree from the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, in 2009. He is currently an assistant professor with the School of Computer Engineering, Nanyang Technological University. His current research interests include wireless sensor networking, pervasive computing, mobile and wireless computing, and etc. He is a member of the IEEE.