A Lightweight Authentication Scheme for Transport System Farecards

Mee Loong Yang*, Ajit Narayanan*, Dave Parry*, and Xiumin Wang †
*School of Engineering, Computer and Mathematical Sciences, AUT University, Auckland, New Zealand
Email: {byang, ajnaraya, dparry}@aut.ac.nz
†College of Information Engineering, China Jiliang University, Hangzhou, China

***Abstract*—Proximity Integrated Circuit Cards (PICC) are widely used for public transport fare collection. The stored contents in the card can only be accessed or modified after the card is able to authenticate the Proximity Coupling Device (PCD) or reader using a shared secret key.**

We propose a new authentication scheme that is not based on shared secret keys. Instead, authentication is based on the card and reader being able to compute an identical pairwise key using their own private keying material obtained from the same source. The computation is done off-line and does not require the participation of a third party. It uses simple modular arithmetic operations over a small binary extension field, achieving fast computation speed using the limited resources in cards. In addition, should the keys be stolen from the cards or readers, the security of the other parts of the system cannot be compromised.

I. Introduction

Fare collection for public transport systems such as trains, buses and ferries need to be fast, convenient, and able to cater for the large number of users at peak periods. Cashless payment using contactless stored value cards, a type of Proximity Integrated Circuit Card (PICC) also referred to simply as "cards", are widely used for this purpose. Examples include the Oyster card in London, Octopus card in Hong Kong, and Charlie card in Boston [1]. Commuters pre-purchase credits which are stored in the cards. When entering and exiting the transit system, the user touches or "taps" the card against a Proximity Coupling Device (PCD) (referred to as a "reader"), which performs read/write operations on the card. At the entry, the gate is opened if there is sufficient stored value, and at the exit, the fare is computed and deducted.The transactions must be fast so as not to impede entrance and exit, allowing a person to walk past the reader without stopping, typically within 300 - 500 ms although some transit agencies require transaction of at most 300 ms [2]. The fast transaction time and sheer volume of users at peak times means that the transactions should are done off-line i.e., locally at the reader which may be fixed or mobile such as in buses. Transaction data may be offloaded to the central servers at appropriate intervals for off-line processing, audit, accounting, and fraud detection.

RFID technologies for contactless fare payments typically implement the ISO/IEC 14443 standard for contactless smart cards operating at 13.56 MHz where the cards are intended to operate within 10 cm of the reader [3]. The MIFARE classic and DESFire farecards are examples of cards used in transit systems.

The cards are passively powered and have limited computational resources. To perform authentication fast enough, the authentication algorithm uses symmetric key cryptographic methods based on a secret key shared between the card and the reader, e.g. in the MIFARE cards [4]. The reader and card prove that they possess the same secret key in order to authenticate each other.

The shared secret key becomes the valued target for attackers. The cards are easily available and the attacker has full control of it once it is obtained. To protect the secret key, it is stored in tamper proof area in the card and cannot be read from outside the card. Nevertheless there has been reports that the shared key can be stolen. For example, it was demonstrated in [5] where, by exploiting weaknesses in the CRYPTO1 algorithm used in the MIFARE Classic card, it was possible to obtain the key by reversely engineering the chip. The more advanced MIFARE DESFire EV1 card which uses the AES algorithm is also vulnerable as demonstrated in [6]. Here the authors used a side channel attack where, by measuring the electromagnetic emanations during cryptographic operations, they were able to recover the secret 3DES key. Once the key is obtained, the cards can be cloned as demonstrated in [7].

There are also reports of invasive attacks using various tools to strip away the physical protective layers and analyse the silicon die to obtain the contents of flash, ROM, FPGAs, etc. [8].

Our Contribution: We propose a new mechanism for authentication between the card and reader that does not use a shared secret key. Instead, the reader and card prove that they each possess unique secret keying material obtained from the same trusted source. It is an off-line process and does not require the participation of a third party. The two devices only need to use their counterpart's *ID*s with their own private keying material to compute a pairwise key which, if identical, authenticates each other. The computations use simple modular multiplications and additions over a small extension field, $GF(2^b)$ where typically, $b = 5$. The scheme can be easily implemented in hardware making it extremely fast.

II. Related work

Light weight methods: RFIDs, being extremely constrained in resources need to use lightweight authentication methods including the above mentioned shared secret key method. In addition, a family of authentication schemes based

on the Hopper-Blum protocol [9] have been proposed, e.g. [10]. The main benefits are in the simple computations using modular arithmetic on binary data. However, the tag and reader also share the same secret key and are subject to the weakness that, if the key is cracked, the system is broken.

PKI ECC methods: RFID smart cards which have Public Key Cryptographic (PKC) functions built into the chips are available and are used in most credits cards. The PKC schemes use a public-private key pair for encryption, decryption, and signature generation and verification in the authentication process. If the private key is stolen, only that particular card is affected. PKC algorithms involve computationally expensive operations including exponentiations and multiplications and for security, use large integers of hundreds or thousands of bits. Usually the more efficient Elliptic Curve Cryptography (ECC) methods are used as they use integers of only hundreds of bits. For security strength equivalent to 80-bit symmetric keys ECC-163 using 163-bit numbers is used, and for 128-bit security ECC-283 is required. ECC-163 have been demonstrated to be suitable for implementation in RFID hardware. The important considerations for RFID application are computation time, digital circuit complexity denoted as Gate Equivalent (GE), and the energy required. The work in [11] showed that it is possible to implement ECC-163 point multiplication which took 300,000 clock cycles requiring 11,904 GE, using 8.57 μW. At 13.56 MHz, this is about 22 ms. The more recent work in [12] designed an ECC-163 processor which took 176,700 clock cycles which at 13.56 MHz, takes about 13.1 ms. It required about 13.8 K GE and consumed 253 μW of power. Authentication using PKC methods require several ECC operations for signing and verifying signatures. The devices also need to exchange their public keys and signatures which requires transmission of hundreds of bits. The work in [13] designed an ECC authentication scheme using Schnorr's signature, and encryption using the linear feedback shift register (LFSR). They simulated the implementation using an FPGA. They did not give figures on the overall performance but presented results for one scalar EC scalar multiplication which took 170 K cycles, required 208.4 μW, and required 21.8 K GE. At 13.56 MHz, this would take about 12.5 ms for one multiplication. As more than one EC multiplication is required as well as other operations for generating and verifying signatures, encryption, decryption, etc., this may not be sufficiently fast for farecard transactions.

III. BYka Authentication for RFID

Our proposal uses the newly developed Blom-Yang key agreement scheme (BYka) [14] as the authentication mechanism between the reader and card.

A. Operation Scenario

The transport company obtains blank cards from a manufacturer with all the algorithms built into the fare card chip. Each card has a unique *ID* hardwired into the chip during manufacture and this cannot be altered. This feature is commonly available, for example the MIFARE DESFire fare cards has a unique 7 byte serial number (*UID*) in locked write-protected non-volatile memory during manufacture [15]. The cards are "uninitialised" and do not have any keying material stored in them. The company has a trusted Key Authority (*KA*) which generates all the keying material and stores them separately in a secure server database. When the user purchases a fare card, the ticket vendor takes a blank card and "initialises" it by retrieving the keying material matching the card *ID* from the server, and writing it into the secure storage area in the card. The storage area should be unreadable from outside the card with access to it broken to prevent reinitialisation or further editing. After this the card can be used to enter and exit the transit system gates for trains, buses, ferries, etc., in the company's system. The user can purchase more credit by presenting the card at the vendor which uses a PCD reader to update the stored value in modifiable memory.

Card usage and operations: All transactions between the card and reader takes place off-line locally at the gates. These include authentication, fare computation, and updating the stored values. Transaction records including card and reader *ID*s, time, stored values, fare collection, etc. are stored locally in the reader and may be uploaded when appropriate to a central server for other purposes such as auditing, accounting, fraud detection, etc.

B. Definition of terms

The following defines some of the terms used in this paper.

Genuine: A card or reader is considered *"genuine"* if it is issued by the company, and all its contents are written using the company's authorized PCD's readers only.

Fake: A card or reader is considered a *"fake"* if some or all of its contents are written by non-company readers, for example, an attacker's reader. The card may have previously been a genuine one but its contents have been altered by the attacker. It may also be an uninitialised card stolen from the store and written with contents by an attacker. In all these cases, the *ID*s of the cards are unchanged.

Clone: A *"clone"* is defined as a hardware or software device constructed with an exact copy of all the components of a genuine device including its *ID*, private keys, etc., and is able to fully to function just like the genuine device. For example, it may be an card or reader emulator constructed of hardware and software using stolen keys and *ID*s. In addition, it may also be able to implement without detection, other nefarious functions not normally done in genuine devices.

C. Attacker Capabilities

We assume that the attacker is able to obtain genuine fare cards, for example by purchasing them from the counter. It is also able to obtain genuine uninitialised cards, for example by theft or other means, from the manufacturer's or the company's stores. However, the attacker cannot modify the cards' unique *ID*s.

It is now accepted that, using invasive or non-invasive techniques, the attacker can extract all the cards' contents including sensitive cryptographic material. The attacker is also

able to implement all the operations, algorithms, etc. just like a genuine device. As a result, the attacker is able to modify the contents of genuine cards, fabricate fake cards, construct clones to emulate as cards, readers, etc. The feasibility of such an attack has been demonstrated in [7].

While the readers installed in stations, buses, etc. can be monitored, we also assume that they can be stolen and their private keys obtained.

IV. The BYka scheme

Our scheme for authentication between the reader and card uses the BYka scheme [14] which enables these two devices who have not previously communicated with each other, to compute a shared secret pairwise key after obtaining each other's *ID*s. This is only possible if each device has a private key-set derived from its own *ID* and the master key kept by the company. In this way, all devices belonging to the company are able to compute pairwise keys with each other as they share a common "ancestor" – the company's master key. Each private key-set is unique and if stolen, affects only the compromised device.

The following notations are used in our scheme:

N – number of master key matrices
m – dimension of matrices and vectors
η – number of *ID* vectors per device
p – an irreducible polynomial
b – the degree of p
$\mathbf{V}$ – public key-set, an $\eta \times m$ matrix
$\mathbf{M}$ – master key, an $m \times m$ matrix
$\mathbf{K}$ – private key-set, an $N\eta \times m$ matrix
$\mathbf{P}$ – pairwise key-set, an $N\eta \times \eta$ matrix

A. System setup

a) Administrator: An administrator is one or any number of entities responsible for setting up the whole system including initialising the PCD readers and PICC cards with their keying material, stored values, and other administrative data.

*b) Public key-set **V**:* Each device has a unique *ID*. A known algorithm $g(\cdot)$ is used with the *ID* to generate η column vectors called the public key-set comprising η $(m \times 1)$ column vectors $\mathbf{V}_{i,\cdots,\eta}$, and each element is in $GF(2^b)$. For example, device *A* with ID_A has the public key-set,

$$\mathbf{V}_A = g(ID_A) = \left[\begin{array}{cccc} \mathbf{V}_{A_1} & \mathbf{V}_{A_2} & \cdots & \mathbf{V}_{A_\eta} \end{array} \right] \quad (1)$$

The public key-set vectors satisfy the following conditions:

1. Each $\mathbf{V}_{x_i}$ is unique across the whole system
2. They are linearly independent

The *ID*s of the cards and readers, and the algorithm $g(\cdot)$ are publicly known.

c) Key Authority, KA: A Key Authority (*KA*) is responsible for generating and safeguarding the system's N master key-set, $\mathbf{M} = \{\mathbf{M}_1, \mathbf{M}_1, \cdots, \mathbf{M}_N\}$. Each $\mathbf{M}$ is an $m \times m$ symmetric matrix with random elements in $GF(2^b)$. For additional security, the *KA* can comprise several separate entities, each one responsible for independently generating and storing one or more of the N master keys.

*d) Private key-set **K**:* This is a set of $N\eta$ row vectors, secret and unique for each device generated by the *KA* using its master key-set and the device's *ID*. The *KA* first generates the public key-set for the card, e.g. *A* using (1). Then the *KA* computes the private key-set $\mathbf{K}_A$ as follows,

$$\begin{array}{rcl} \text{for } i = 1, \cdots, N \quad \text{and} \quad j = 1, \cdots, \eta & & \\ \mathbf{K}_{A_{ij}} & = & \mathbf{V}_{A_j}^T \cdot \mathbf{M}_i \pmod p \end{array} \quad (2)$$

Each $\mathbf{K}_{A_{ij}}$ is a $(1 \times m)$ row vector. They are stored in the private key-set matrix in a random order.

$$\mathbf{K}_A^T = \left[\begin{array}{cccc} \mathbf{K}_{A_{11}}^T & \mathbf{K}_{A_{12}}^T & \cdots & \mathbf{K}_{A_{N\eta}}^T \end{array} \right]$$

The private key-set matrix needs to be loaded into the card before it can be used. This can be done either during manufacture or when it is purchased.

*e) Pairwise key-set **P**:* When two devices, say reader *A* and card *B* need to obtain their pairwise key, they first exchange their *ID*s. Then, using each other's *ID*, they derive their counterpart's public key-sets from the *ID* using (1). The reader obtains $\mathbf{V}_B$ and the card obtains $\mathbf{V}_A$. Each device then computes their pairwise key-set $\mathbf{P}$ as follows.

Reader *A*:

$$\mathbf{P}_{AB} = \mathbf{K}_A \cdot \mathbf{V}_B \pmod p \quad (3)$$

$\mathbf{P}_{AB}$ is an $N\eta \times \eta$ matrix with elements $(\mathbf{V}_{A_j}^T \mathbf{M}_i)\mathbf{V}_{B_k}$ where $i = 1, \cdots, N$ and $j, k = 1, \cdots, \eta$.

Similarly, card *B*:

$$\mathbf{P}_{BA} = \mathbf{K}_B \cdot \mathbf{V}_A \pmod p \quad (4)$$

The elements of $\mathbf{P}_{BA}$ are similarly $\mathbf{V}_{B_j}^T \mathbf{M}_i \mathbf{V}_{A_k}$. Consider an element and its transpose, $(\mathbf{V}_{B_j}^T \mathbf{M}_i \mathbf{V}_{A_k})^T = \mathbf{V}_{A_k}^T \mathbf{M}_i^T \mathbf{V}_{B_j}$. Since $\mathbf{M}_i$ is symmetric, and the matrix $\mathbf{V}_{A_j}^T \mathbf{M}_i \mathbf{V}_B$ is scalar, the two matrices $\mathbf{P}_{AB}$ and $\mathbf{P}_{BA}$ have identical elements but located in different positions. Each device use these elements to construct a pairwise key of the desired length using some algorithm $f_k(\cdot)$, i.e.,

$$\begin{array}{rcl} \text{Reader R: } K_{AB} & = & f_k(\mathbf{P}_{AB}) \\ \text{device B: } K_{BA} & = & f_k(\mathbf{P}_{BA}) \\ \text{and} \quad K_{AB} & = & K_{BA} \end{array}$$

For example, a possible algorithm for $f_k(\cdot)$ is to multiply all the non-zero elements together modular a suitable irreducible polynomial to give the required key size. Finally, a joint session key K_s can be constructed by using another function $f_s(\cdot)$, for example a hash function, with K_{AB} or K_{BA} and n, a random number as inputs. How the number n is shared will be describe as follow.

B. Authentication protocol

The ISO/IEC-14443-3 protocol has 3 phases. The initialisation and anti-collision phase commences when a card comes into proximity with a reader. At the end of this phase, the reader has obtained the *ID* of the card, and the card is in an active state. The authentication phase follows where the reader

and card authenticates each other using random challenges encrypted using their shared secret key. Three exchanges are required. Then the application phase commences where commands are sent to the card.

Our scheme implements all the operational sequences of the above ISO/IEC-14443-3 protocol. The only difference is the use of a computed pairwise session key in the authentication phase.

Initialisation phase: When the user presents his/her card denoted as *B*, by bringing it near a reader denoted as *A*, the initialization and anti-collision phase proceeds as in ISO/IEC protocol. On completion, the reader has obtained the ID_B of the card, and the card is now in an active state.

Authentication phase:

Step 1. PCD Reader: The reader with ID_A having obtained the ID_B of card *B* derives the public key-set $\mathbf{V}_B = g(ID_B)$. Using this, it computes the pairwise key-set $\mathbf{P}_{AB}$ using Eqn. (3) from which it generates a pairwise key $K_{AB} = f_k(\mathbf{P}_{AB})$. It also generates two random numbers: n which is used with K_{AB} to obtain the joint session key $K_s = f_s(K_{AB}, n)$, and c_A which is used as a challenge. It then constructs the message, $M_1 = \{ID_A \parallel n \parallel E(c_A \parallel n)_{K_s}\}$ and sends it to the card.

$$1: \quad A \rightarrow B : M_1 = \{ID_A \parallel n \parallel E(c_A \parallel n)_{K_s}\}$$

Step 2. PICC Card: The card obtains ID_A, derives the public key-set $\mathbf{V}_A = g(ID_A)$, computes the pairwise key-set $\mathbf{P}_{BA}$ using Eqn. (4) and then obtains $K_{BA} = f_k(\mathbf{P}_{BA})$ and then the joint session key $K_s = f_s(K_{BA}, n)$. The decrypted n is compared to the clear text n, and if verified, it retrieves the challenge c_A and computes the required response $r_A = c_A + 1$. It then generates its own random challenge c_B, constructs the message $M_2 = E(r_A \parallel c_B)_{K_s}$, and sends it to the reader.

$$2: \quad B \rightarrow A : M_2 = \{E(r_A \parallel c_B)_{K_s}\}$$

Step 3. Reader Verification: The reader decrypts M_2 using the session key K_s, checks that $r_A = c_A + 1$. If verified, the reader is assured that card with ID_B possess the private key-set computed from ID_B by the *KA*, and is thus genuine. It computes the response $r_B = c_B + 1$ and updates its challenge $c'_A = r_A$. It is now ready to move to the application phase, for example to read the stored value in the card by issuing the command cmd. It constructs an encrypted message $M_3 = E(c'_A \parallel r_B \parallel cmd)_{K_s}$ and sends it to the card.

$$3: \quad A \rightarrow B : M_3 = \{E(c'_A \parallel r_B \parallel cmd)_{K_s}\}$$

The card decrypts M_3 and after verifying the response $r_B = c_B + 1$, the card is assured that the other party is indeed one with the claimed ID_A which also obtained its private key-set from the trusted *KA*. After this, it transits to the application phase.

Application phase: The operations of the application phase, such as reading and updating contents, etc., proceeds as usual without any change. All transactions are encrypted using the session key K_s and each message includes a new challenge and a response to the previously received challenge. In this way all messages are part of a stream initiated by the reader.

C. Security of the scheme

We consider attacks on the scheme on two fronts – the card itself, and the protocol. For the attacks on the card, it is assumed that the attacker is able to obtain as many genuine cards as necessary. We assume that the attacker is able to extract the private key-sets from cards even though these are stored in a secure unreadable area in the card. We also assume that the PCD readers can be stolen from unattended buses, stations, etc. and their keys obtained.

Attacks on card: The attacker can attempt to:

1. Recover the master key matrices
2. Fabricate fake cards
3. Clone cards and readers

1. Recovering the master keys: If the master key matrices can be recovered, the attacker would be able to clone readers, fabricate fare cards, etc., and compromise the entire system. This is equivalent to compromising a globally shared secret key. Various attacks to recover the master keys from captured private keys were analysed in [16] in which it was shown that the BYka scheme was resilient against these attacks. The main features enabling the scheme to be resilient are briefly described as follows.

The BYka scheme is based on Blom's scheme [17]. Here, there is one master key $\mathbf{M}$ which is a symmetric $(m \times m)$ matrix, and each device has one public key $(m \times 1)$ vector $\mathbf{V}$ and one private key $(1 \times m)$ row vector $\mathbf{K} = \mathbf{V}^T\mathbf{M}$ (mod n), a prime number. After devices *A* and *B* obtained each other's public key vectors, they compute their pairwise key $K_{AB} = (\mathbf{V}_A^T\mathbf{M})\mathbf{V}_B$ (mod n) and $K_{BA} = (\mathbf{V}_B^T\mathbf{M})\mathbf{V}_A$ (mod n), respectively. The key size is the same size as n. The Blom's scheme is unconditionally secure if all the public key vectors are linearly independent, and no more than $(m-1)$ private key vectors are stolen [18]. Otherwise, if non-linearly independent public key vectors are used, the attacker would be able to fabricate new valid public and private key vectors using linear combinations of the captured ones. More drastically, if m or more private key vectors are obtained, the attacker can use them to construct a system of linear equations whose solution is the master key matrix itself.

The BYka scheme however, is able to remain resilient against any number of private key-sets being stolen by using multiple keys, random storage of the private key-set vectors, and with operations over a small finite field. Here, the *KA* has a master key-set comprising N master key matrices. Each device has a public key-set comprising η vectors used in permutation to obtain the $N\eta$ vectors in the private key-set. Now, before the stolen private key-set can be used to construct the system of equations, each private key-set vector must first be associated with the particular public key vector and master key matrix used to compute it, i.e. discover the private-public-master key association (PPMka). As the the public key vectors are unique, the private key-set vectors storage are in a random order, and

the operations are over a very small finite field of b bits, the PPMKa cannot be easily found. The analysis in [16] showed that by selecting suitable values of N, η, m, b the number of possible PPMka's from captured keys using the most efficient method can be made so large that it requires an infeasible number of attempts, e.g. 2^{80} or 2^{128}. Based on the notion of security strength in the NIST Standard [19], we consider these to be of security strengths 80-bits and 128-bits respectively.

2. Fabricate fake cards: Assume that the attacker is able to obtain blank cards which have not been initialised, for example by stealing from the company's store or from the manufacturer. Assume also that the attacker is able to obtain all the keying material from card *B*. The attacker is able to write the private key-set $\mathbf{K}_B$ into a blank card with ID_X, creating a fake card since the *ID* cannot be changed. However this card is unusable as the private key-set is unrelated to the fake card's ID_X. When this fake card is presented to a genuine reader, say *A*, the reader obtains ID_X and computes the pairwise key-set $\mathbf{P}_{AX} = \mathbf{K}_A \cdot \mathbf{V}_X$. Meanwhile, the card obtains ID_A and computes a different pairwise key-set $\mathbf{P}_{BA} = \mathbf{K}_B \cdot \mathbf{V}_A$. Since the fake card *B* and reader C are not able to obtain the same session key, the authentication fails. In this way, stolen blank cards are useless unless the private key-sets matching the stolen card *ID*s are stolen from the secure server as well.

The attacker can attempt to construct a valid private key-set for ID_X using suitable linear combinations of compromised private key-sets, a possibility in the original Blom's scheme. However, in the BYka scheme, the attacker must first discover the PPMka's which is infeasible.

3. Clone cards and readers : Using stolen keying material from a card, the attacker is able to construct a (hardware or software) reader or card emulator using the matching *ID* and private key-set. The card emulator can be used as a fare card at genuine readers. By repeatedly cloning new cards, the attacker can obtain unlimited travel.

The attacker may also attempt to build a reader emulator and use it to modify stored values in genuine cards. If this is successful, the attacker would be able to masquerade as a vendor to collect payments and use the reader to recharge genuine cards. Our scheme has a simple mechanism to prevent PCD reader cloning attacks using stolen PICC card keys. In our scheme, the cards and readers are assigned *ID*s from different ranges. For example, card *ID*s are 7 bytes long, while reader *ID*s are only 6 bytes. Genuine PICCs are designed and built with hardware to handle only 6 byte-*ID*s, i.e. only from genuine readers. The reader cloned with keying material derived from the 7-byte *ID* of the compromised card cannot be authenticated by a genuine card.

If the keys are stolen from PCD readers, they can be used to clone readers. Our scheme is able to support suitable countermeasures. As each transaction is possible only if both parties use their real *ID*s related to their private key-sets, the reader *ID*, together with time, location, etc., can be recorded in the card at each transaction. These records can be captured by genuine readers when the card is used. The information can be used by other off-line fraud detection systems to detect cloned PCDs readers and cards.

Attacks on the protocol: It is assumed that the attacker is able to record messages between the reader and a genuine card using some device. The attacker can attempt to replay messages with the aim to reset the value of the card, or report a false stored value at entrance gates, etc.

Consider that an attacker purchases new credit for a genuine card at an unattended recharging station, and all messages between the card and reader during the transaction were recorded. When the card value is depleted, the attacker would attempt to send the recorded messages to reset the card with the previously purchased credit. Our scheme protects against replay attacks by requiring each message received by the card to contain the appropriate response to the challenge contained in the previous message sent. As a new random challenge is generated for each new session, a replayed message would have only a very small chance of having the correct response to the current challenge and would be rejected. Similarly, replaying an old message containing a larger stored value to a reader would be easily detected.

V. Discussion

Transmission: The authentication process requires the devices to send their *ID*s to their counterparts. This is a basic requirement for transactions in general, and by just exchanging these few *ID* bits, our scheme enables the reader and card to derive a pairwise session key for authenticating each other.

Performance and GE: The design and hardware implementation of our scheme is left to a future work. Here we only sketch out the expected performance and GE required by comparing with other work, and by conducting some preliminary design investigations.

Computation time: On top of the time required in the ISO/IEC-14443-3 scheme for generating, encrypting, decrypting and verifying the challenges, ours requires additional time to compute the pairwise key. This involves generation of η public key-set vectors, $N\eta^2 m$ modular multiplications of b-bit operands, and derivation of the pairwise key using a suitable method, for example by multiplying the $N\eta^2$ pairwise key-set elements together modular a 128-bit irreducible polynomial.

The most intensive operation is the modular multiplication. This can be efficiently implemented in hardware using, for example, the Montgomery modular multiplication algorithm. There have been a lot of studies done on efficient hardware implementation of the Montgomery multiplier, especially for use with large finite fields. As an indication, the work in [20] reported its implementation in hardware in which the modular multiplication took at most $\lfloor \frac{2}{3}(b+2) \rfloor + 3$ cycles with operands of $b = 128, 256$, and 512 bits. With $b = 5$ in our case, each multiplication would require at most 8 clock cycles, and with $N = 9, \eta = 5, m = 26$ for 128 bit security, the total time to obtain the pairwise key would be 49.2 ms @ 13.56 MHz.

Gate Equivalent: Our scheme requires one 5×5-bit multiplier and one 5×128-bit modular multiplier. Considering that the ECC-163 implementation in [12] required 13.8 K GE, our requirement should be less.

Preliminary design investigations: Our modular multiplications using small operands of $b = 5$ bits allow the possibility of implementations using a lookup table or the schoolbook shift and add method.

The lookup table method is extremely area consuming but very fast. Our preliminary design for the 5-bit modular multiplication table using decoders and encoders would require around 10,025 GE, and each look up takes only a couple of clock cycles.

The schoolbook modular multiplication using the shift and add method requires at most 15 clock cycles to do one multiplication. Overall, it would take around 110 K cycles to obtain the 128-bit pairwise key using $N = 9, \eta = 5, m = 26$ and $b = 5$. At 13.56 MHz this amounts to around 8.1 ms. Only a few 5-bit registers and 128-bit registers are used, requiring a total of around 1836 GE.

Private key storage: The private key-set consists of $N\eta mb$ bits, for example 5850 bits. Its large size also serves as a deterrent against attempts to steal it using brute force or other attacks such as those involving physical invasive methods.

VI. Conclusion

Proximity farecards play a vital role in modern public transport systems. However, the low computational power of the embedded chips, the requirements for fast transaction at the entry and exit gates, and the large number of devices in use tend to limit the authentication methods to those using shared secret keys. A new scheme in which the reader and farecard computes their pairwise key for authentication is proposed. It operates off-line and does not require the participation of a third party. It has minimum communication overheads and the two devices only need to exchange their *ID*s in order compute their pairwise key. As it uses simple modular arithmetic operations over a small binary extension field, it is fast and requires little computing resources making it very suitable for transportation system farecards. An important feature is that, even if the private keying material of a large number of farecards are stolen, the impact is limited and cannot compromise the whole system.

Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grant No. 61379027.

References

[1] K. Markantonakis, K. Mayes, D. Sauveron, and I. G. Askoxylakis, "Overview of security threats for smart cards in the public transport industry," in *e-Business Engineering, 2008. ICEBE '08. IEEE International Conference on*, Oct 2008, pp. 506–513.

[2] *Transit and Contactless Open Payments: An Emerging Approach for Fare Collection*, Smart Card Alliance Std. TC-11 002, November 2011. [Online]. Available: http://www.smartcardalliance.org/resources/pdf/Open_Payments_WP_110811.pdf

[3] *Requirements of ISO/IEC 14443 Type B Proximity Contactless Cards, Application Note*, Amtel Corporation, 2005. [Online]. Available: http://www.atmel.com/images/doc2056.pdf

[4] "MIFARE DESFire EV1 AES Authentication with TRF7970A," Texas Instruments, Tech. Rep., December 2014. [Online]. Available: http://www.ti.com.cn/cn/lit/an/sloa213/sloa213.pdf

[5] F. D. Garcia, G. Koning Gans, R. Muijrers, P. Rossum, R. Verdult, R. W. Schreur, and B. Jacobs, *Computer Security - ESORICS 2008: 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6-8, 2008. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, ch. Dismantling MIFARE Classic, pp. 97–114. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-88313-5_7

[6] D. Oswald and C. Paar, *Cryptographic Hardware and Embedded Systems – CHES 2011: 13th International Workshop, Nara, Japan, September 28 – October 1, 2011. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, ch. Breaking Mifare DESFire MF3ICD40: Power Analysis and Templates in the Real World, pp. 207–222. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-23951-9_14

[7] T. Kasper, I. von Maurich, D. Oswald, and C. Paar, "Cloning Cryptographic RFID Cards for 25$," *5th Benelux workshop on information and system security. Nijmegen, Netherlands*, November 2010.

[8] A. Tria and H. Choukri, "Invasive Attacks," *Encyclopedia of Cryptography and Security, 2*, vol. 2011, no. Part 9, pp. Pages 623–629, Nov. 2011. [Online]. Available: http://hal-emse.ccsd.cnrs.fr/emse-00644128

[9] N. J. Hopper and M. Blum, *Advances in Cryptology – ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, ch. Secure Human Identification Protocols, pp. 52–66. [Online]. Available: http://dx.doi.org/10.1007/3-540-45682-1_4

[10] Z. Lin and J. S. Song, "An Improvement in HB-Family Lightweight Authentication Protocols for Practical Use of RFID System," *Journal of Advances in Computer Networks*, vol. 1, no. 1, pp. 61–65, March 2013.

[11] D. Hein, J. Wolkerstorfer, and N. Felber, "ECC is Ready for RFID – A Proof in Silicon," in *LNCS*, ser. LNCS, R. Avanzi, L. Keliher, and F. Sica, Eds., no. 5381, 2009, pp. 401–413.

[12] Z. Liu, D. Liu, X. Zou, H. Lin, and J. Cheng, "Design of an Elliptic Curve Cryptography Processor for RFID Tag Chips," *Sensors*, vol. 14, no. 10, pp. 17 883–17 904, Oct 2014.

[13] D. Liu, Z. Liu, Z. Yong, X. Zou, and J. Cheng, "Design and Implementation of An ECC-Based Digital Baseband Controller for RFID Tag Chip," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 7, pp. 4365–4373, July 2015.

[14] M. L. Yang, A. Al-Anbuky, and W. Liu, "An Authenticated Key Agreement Scheme for Wireless Sensor Networks," *Journal of Sensor and Actuator Networks*, vol. 3, pp. 181–206, 2014.

[15] *Short Form Specification Mifare DESFire MF3 IC D40, Revision 3.0*, Philips Semiconductors, 2004.

[16] M. L. Yang, A. Al-Anbuky, and W. Liu, "Security of the Multiple-Key Blom's Key Agreement Scheme for Sensor Networks," in *ICT Systems Security and Privacy Protection*, ser. IFIP Advances in Information and Communication Technology, N. Cuppens-Boulahia, S. Jajodia, and F. Cuppens, Eds. Springer-Verlag GmbH, Berlin, Heideberg, June 2014, pp. 66–79.

[17] R. Blom, "An Optimal Class of Symmetric Key Generation Systems," Linkopping University, Linkopping, Sweden, Tech. Rep., 1984.

[18] A. J. Menezes, P. C. Oorschot, and S. A. Vanston, *Handbook of Applied Cryptography*. CRC Press, Inc., 2001.

[19] E. Barker, W. B. Barker, W. Burr, W. Polk, and M. Smid, *Recommendation for Key Management – Part 1: General*, National Institute of Standards and Technology Std. Special Publication 800-57, Rev. 3, July 2012.

[20] M. E. Kaihara and N. Takagi, "A hardware algorithm for modular multiplication/division," *IEEE Transactions on Computers*, vol. 54, no. 1, pp. 12–21, Jan 2005.