Journal of Transportation Technologies, 2017, 7, 50-69
http://www.scirp.org/journal/jtts
ISSN Online: 2160-0481
ISSN Print: 2160-0473

Touch-and-Go Mobile Payment System

Peter Rulić, Bojan Kotnik, Saša Klampfer, Amor Chowdhury

Margento R & D, Maribor, Slovenia
Email: peter.rulic@margento.com, bojan.kotnik@margento.com, sasa.klampfer@margento.com, amor.chowdhury@margento.com

How to cite this paper: Rulić, P., Kotnik, B., Klampfer, S. and Chowdhury, A. (2017) Touch-and-Go Mobile Payment System. *Journal of Transportation Technologies*, 7, 50-69.
http://dx.doi.org/10.4236/jtts.2017.71004

Received: November 7, 2016
Accepted: January 8, 2017
Published: January 11, 2017

Abstract

Contactless mobile payment devices are fast, convenient and user friendly means for executing small value business transactions. This kind of transactions is preferred to be executed conveniently and on the fly by just tapping a mobile payment device to a mobile terminal. Mobility, convenience and fast transaction execution are very important payment system properties in potentially crowded places such as in public transportation. These properties are in practice causing transaction execution problems, such as: delays in offline transaction execution, transaction atomicity failures caused by mobile data communication drops, lack of central control on mobile transaction system components and increased vulnerability risk caused by easy physical access to the mobile transaction system components. This paper describes a holistic transaction processing model for resolving mentioned problems in micro-payment mobile touch-and-go transaction systems. Defined processing model was implemented on real public transportation mobile payment system where it was proven as a robust solution for execution of small value touch-and-go mobile ticketing transactions.

Keywords

Mobile Transaction System, Business Transaction, NFC, Mobile Payment

1. Introduction

Basic general requirements for an arbitrary transaction system are today known as ACID, which stands for atomicity, consistency, isolation and durability. ACID term was first used by [1], and it is widely used for describing transactions [2] [3] [4]. ACID properties of a mobile transaction system are often challenged, because of frequent disconnections on mobile data transmission channels. Moreover, real mobile micropayment systems have to meet the following additional requirements:

- *Fast business transaction execution by users*. In practice, the so-called small value or micropayment touch and go transactions have to be executed in negligible time, where the user does not have to confirm the transaction execution or enter a pin number. Quick execution of business transactions reduces congestions in crowded places and it is necessary in various cases, such as public transportation check-ins, where transactions are executed quickly with a single tap of contactless devices to the point of sale terminal.
- *User-friendly and always available execution of business transactions*. Business transaction system has to be available for transaction execution regardless of various disconnections on mobile data transmission channels. Users have to be able to conveniently and autonomously execute business transactions on mobile terminals with various contactless devices, such as contactless smartcards or smart phones [5] [6].
- *Reliable automated central control on mobile transaction system components*. Mobile terminals can temporary work in offline mode, during disconnections on mobile data transmission channels. Offline operation makes various transaction failures and failures on mobile transaction system equipment hard to detect, because of missing real-time operational data. Operator of mobile business transaction system has automated full control on mobile parts of transaction system such as mobile terminals.
- *Enhanced transaction system security*. Practical transaction system security has to be based on strong cryptographic techniques for data protection and maintained with appropriate transaction system security procedures. Such type of security has to provide automatic monitoring on business transaction system for potential security breaches based on business transaction rules. Therefore potential security breaches can be rapidly detected and rehabilitated, which is essential for damage mitigation. Another practical aspect for enhanced security in mobile micropayment systems is transaction execution without confirmation with pin number.

We will abbreviate these special requirements as FUCS (F for fast transaction execution, U for user friendly execution of transactions, C for reliable control on the transaction system components and S for the enhanced security). In this paper we will present a processing model of the mobile micropayment near real-time business transaction system, based on ISO/IEC 14443 or its extension Near Field Communication (NFC) [7] compatible contactless devices, such as contactless smartcards, NFC enabled mobile phones or similar devices [8] [9] [10] with ACID and FUCS requirements.

1.1. Description of the Paper Structure

This paper is composed of six main sections and end literature section and is organized as follows: The second section describes implemented mobile transaction system architecture and provides basic insight into the mobile environment for execution of business transactions. The detailed business transaction logic and the problem of ACID and FUCS requirements are defined in the third section.

We continue with the central processing system description in fourth section, where business account synchronization, managing NFC disconnections and enhanced control and security are defined. In fifth section the practical implementation case of public transport City Card Urbana is presented, meanwhile sixth section concludes the paper.

1.2. Research Motivation

The motivation for this research was a gap between requirements and practical problems, which emerged with implementation of micropayment NFC mobile business transaction system called Urbana, which is Margento mobile payment system [11] [12] used for public transportation ticketing and mobile payments. Urbana is required to operate with intermittent mobile data transmission channels with ACID and FUCS properties.

Special challenge represents resolving unverified commit operations caused by single tap NFC transactions (see third chapter, where mathematical model of terminal transaction processing is described) on a MTD (Mobile Transaction Device). Unverified MTD data commit operations lead to transaction atomicity failures. These failures in practice occur when data communication between terminal and MTD is disconnected during commit verification of a single tap transaction.

Additional challenge is ensuring security and control over mobile components of the transaction system. Terminals and MTDs are mobile components, which can operate offline and are therefore vulnerable for potential malfunctions or system intrusions.

1.3. State of the Art-Related Work

Our transaction system relates to three basic transaction system designs: transaction processing systems, real-time systems and batch processing systems [2].

The transactions of transaction processing systems are initiated on terminals with terminal programs, where user requests are converted into executable transactions. Transactions are transferred to central processing system, where they are completed. Transaction data is finally stored in database system. Real-time systems are online systems, with limited time for transaction execution. Ideally, mobile transaction system is a real-time online system, where transactions are executed online and committed in negligible time. Unreliable mobile data transmission channels are preventing real-time transaction execution. In contrast to real-time systems, execution time is not critical for batch processing systems, where transactions are packed in batches and saved. Batches with transactions are processed at a later time.

Disconnections between mobile and fixed hosts can generally be managed with the basic idea behind a batch processing system: mobile hosts can operate offline and locally store transaction data while mobile connection is unavailable. At a later time when connection is available, the transaction data is transferred between local and fixed hosts and processed. This concept is a base for various relaxed ACID transaction models such as: pro-motion [13], pre-commit [14], zippering [15], where with relaxing one or more ACID properties, the mobile transaction can be successfully executed, despite disconnections between mobile and fixed hosts. The same basic idea is used in our mobile transaction system, where disconnection problems between terminals and central processing system are solved with offline terminal operation where transactions are packed in batches on terminals. Fast transaction execution is required in our business transaction system, that's the reason why our transactions are committed on MTDs immediately at the time of execution. Such transactions cannot be canceled, which is similar as in the pre-commit transaction model.

Managing presented disconnections on NFC data transmission channel requires transaction failure recovery, similar to [16] [17], where different strategies for transaction repairing are used.

The presented related work is mainly focused at resolving ACID problems caused by intermittent connections in mobile transaction systems. Related work does not cover practical problems of mobile payment systems. Similar prototype transaction system for mobile payment, which was presented in [18] [19] is focused mainly on user's acceptance and opinion on convenience of NFC payments. It does not cover ACID and FUCS problems in big mobile payment systems. Our mobile payment system presents solution to FUCS requirements, which are not covered in surveyed literature on mobile transaction systems.

2. Transaction System Architecture

The basic architecture of our transaction system is shown in **Figure 1**.

Business transactions represent interactions between businesses and users. Business transactions are initiated on mobile terminals with ISO/IEC 14,443 or NFC compatible devices referred as MTDs. MTDs are secured NFC hardware devices capable of secure data storing and communicating with terminals. In the following text business transactions will also be noted simply as transactions. Terminals are hardware devices capable of secured running of terminal programs and storing the transaction data. Terminals are connected with the transaction processing center over a mobile network, which supports internet protocol [20] such as GPRS [21] [22], UMTS [23] or similar. Transaction processing center uses relational SQL database [24] [25] for transaction data storage.

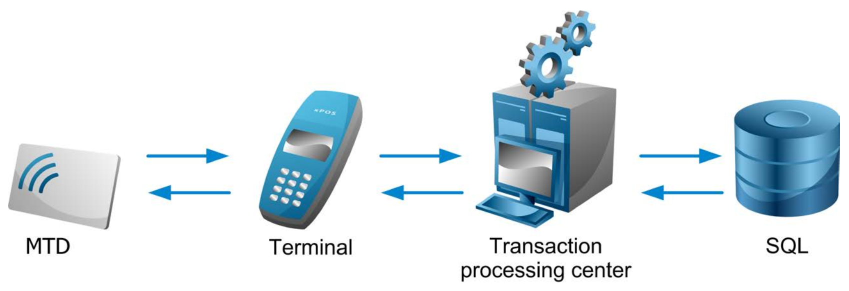

Figure 1. Architecture of the NFC transaction system.

Secure data storing on terminals and MTDs and encrypted data transmission between terminals and transaction processing center; allow transactions to be securely conveyed from terminals to the central processing system.

Transaction data is stored in the database system as presented in **Figure 2**.

The first is a mobile spread database, where users' accounts data is stored on their MTDs. The mobile database has the following properties: A single MTD contains the data of a single transaction system account. The data on a MTD can only be accessed by terminals during transaction execution.

These properties impede centralized and random transaction data access on terminals, because the data on MTDs is available only during transaction executions on terminals.

The second database is central SQL database, which contains near-real time copy of the mobile spread database. User account copies in the central database are synchronized with user accounts on MTDs with transactions executed on terminals and transferred to the central side of the transaction system. Such database system provides centralized system control and random data access.

We can observe the whole mobile transaction system from the top layer where the transactions take place. In **Figure 3** the similarity between the terminal side and the central side of the transaction system is presented in the sense of transaction processing and data storage. Transactions on the terminal side are performed on multiple terminals, where user account data is stored on MTDs. While the central side uses single software component called offline-processor for transaction processing and SQL database for data storage. Such transaction architecture enables user-friendly and always available transaction execution.

Transactions can be executed on mobile terminals offline, without fixed connection to the central processing system. Executed transaction data is temporary stored on terminals in transaction batches. When mobile connection between a terminal and the central side is established, transaction batches are sent from a terminal to the central side of transaction system, where they are processed to synchronize the SQL database with the mobile spread MTD database.

3. Developed Mathematical Model of Terminal Transaction Processing

We will present basic business transaction logic with the definition of terminal transaction processing. Terminal transaction processing is executed when a user's MTD is taped to a terminal.

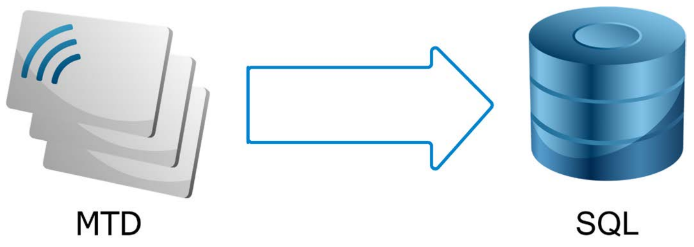

Figure 2. Data storage in mobile business transaction system.

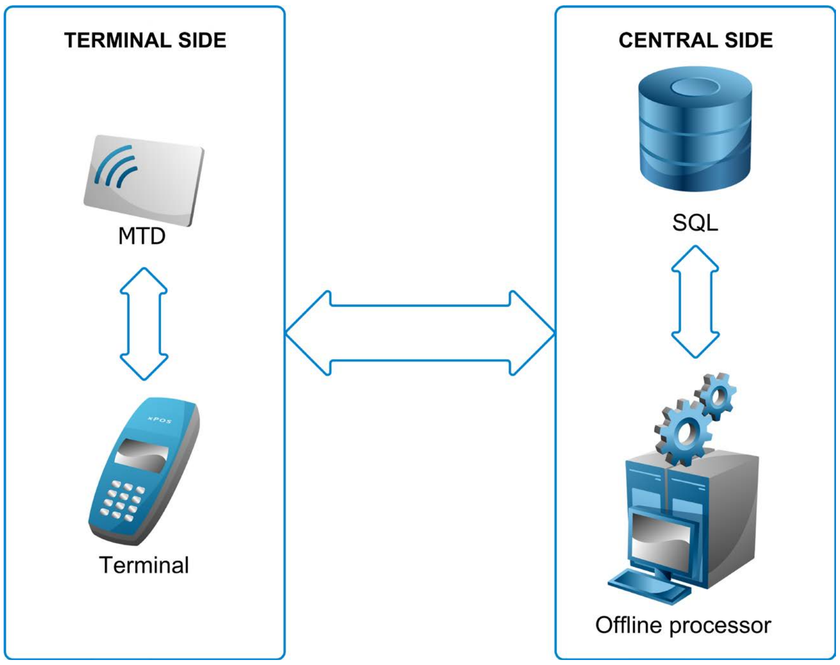

Figure 3. Transaction processing and data storage on terminal and central side.

Every user has a business account stored on a MTD. We denote the business account, also called just the account, as A_i, where index i represents i-th user. Business account contains electronic wallet, which can be topped up with credit for cashless payments. The account also allows purchases and validations of various products such as electronic tickets or vouchers. An arbitrary account contains the following data:

$$(I, W, P, L, Z, H) \in A \tag{1}$$

where I represents a unique identification number of the account A, which corresponds to i-th user. W is the account wallet balance credit. P represents details of purchased products available for use to the user. P is a set of purchased products $(p_1, p_2, \cdots p_N) \in P$, where index N represents the number of products. The time of last transaction execution in the account A is represented by L. Identification of the terminal on which last transaction was executed is represented by Z. All executed transactions on the account A are collected in a set of transactions H, $(B_1, B_2, \cdots B_M) \in H$. Index M represents the number of executed transactions.

When transaction is successfully executed, the account data is changed. We denote the states of an account between two successive transactions with successive upper script index integers such as A^m and A^{m+1}. Similarly we denote the execution order of successive transactions with notations: B^m and B^{m+1}.

A transaction of i-th user account in m-th state is executed in a logical order, where the account state A_i^m is changed to A_i^{m+1} with the transaction B_i^m. The account state on a MTD can be changed with execution of transaction B, which is represented as a message or log generated by a terminal. Transaction B consists of transaction elements denoted as:

$$(f,c,t,l,y,z,a,w,p_n,s) \in B \tag{2}$$

Symbols representing the transaction elements are described as:

- *f*—Transaction type is used for classification of transactions; We use four transaction types: *f* = {*refill, debit, purchase, validation*}. The transaction type is set on a terminal during transaction execution, according to the purpose of the transaction.
- *c*—MTD identification number is used for identification of user accounts. It is unique for each MTD device and used for the identification of the accounts.
- *t*—Time of transaction execution is current time on a terminal.
- *l*—Time of the last successfully executed transaction is stored on account from previous successful transaction execution.
- *y*—Unique terminal identification number on which a transaction was executed.
- *z*—Terminal identification of the last successfully executed transaction is stored on account from previous successful transaction execution.
- *a*—Transaction amount is set on a terminal according to a business price. It can take positive or negative value.
- *w*—Wallet balance after an executed transaction is taken from the account.
- p_n—Details of *n*-th purchased product where n = 1… *N* are chosen on a terminal, if transaction type *f* = *purchase*.
- *s*—Data commit status on a MTD.

With transaction execution the following account values are modified:

Wallet credit $W_i \in A_i$ can be increased or decreased by the value $a_i \in B_i$, or left unchanged:

$$W_i^{m+1} = W_i^m + a_i^m \tag{3}$$

New *n*-th purchased product element p_n is added to user's purchased products set $P_i \in A_i$ with the transaction type *purchase*. This is expressed with equation

$$f = \text{purchase} \Rightarrow P_i^{m+1} = P_i^m \leftarrow P_n^m \tag{4}$$

where the symbol ← represents insertion of purchased product details p_n into the set P_i.

Last successful time $L_i \in A_i$ is set as the current time $t \in B_i$ and last terminal identification $Z_i \in A_i$ is set as a current terminal identification:

$$L_i^{m+1} = t_i^m \tag{5}$$

$$Z_i^{m+1} = y_i^m \tag{6}$$

New transaction is added to the set of transactions *H*:

$$H_i^{m+1} = H_i^m \leftarrow B_i^m. \tag{7}$$

The success of a transaction depends on the success of a MTD data commit operation. When a transaction *B* is executed, equations from (3) to (7) change the account state, only if data commit operation on a MTD is *successful*. Data commit operation on a MTD can be either *successful* or *unsuccessful*; therefore a transaction is *successful* or *unsuccessful*.

Terminal verifies success of the commit operation in the final stage of transaction execution. The commit operation result is represented with the MTD data commit status s_i^m.

$$s_i^m = \{\text{successful, unsuccessful, unverified}\} \tag{8}$$

Verified data commit operation yields *successful* or *unsuccessful* data commit result. The *unverified* data commit result was introduced because in practice a terminal cannot always verify the success of data commit operation on a MTD. Disconnection can occur before a terminal verifies the commit operation. Failed reconnection or timeout enables a terminal to set *unverified* data commit status s_i^m = *unverified* and continue executing next transaction.

The *unverified* data commit result causes atomicity failure. Meanwhile consistency and durability are assured with data commit operation on a MTD. Although the commit result in transaction *B* is set as *unverified* as presented with Equation (8), the actual state of an account *A* on a MTD is only changed with *successful* commit operation, so account *A* on a MTD remains consistent after executed transaction. The isolation transaction property is assured with a logical terminal rule, where only one transaction can be executed on one terminal at a time. Transaction executions are independent of each other, thus multiple transactions can be executed simultaneously on multiple terminals with isolation property.

The presented terminal transaction processing logic provides always available, fast and user friendly execution of transactions. But the defined transaction processing on terminals alone, cannot provide atomic transaction execution. Moreover, transaction processing logic on terminals does not provide central control on executed transactions or random user account data access. The solution to these drawbacks is provided in the next sections, where details of the central processing system are described.

4. Developed Mathematical Model of Central Processing System

After execution the transaction data *B* is temporary stored in batches on terminals. When mobile connection is available, transaction batches are sent to central processing system where they are processed by the offline-processor interacting with the SQL database. SQL database contains copies of all user accounts on MTDs. Central side processing mechanisms synchronize the original business accounts on MTDs with the central side copies.

Similarly as on the terminal side, the central processing system also uses the equations from (3) to (7) for transaction processing. The elements of an arbitrary account A_i has to have the same values on the terminal and on the central side after every transaction B_i has been processed. To make further formulation more understandable, we denote central account values with a right upper script dash line, e.g. $(I', W', P', L', Z', H') \in A'$. Ideally, an account on the central side is processed the same way as on the terminal side.

Processing of transactions on the central side is not as straightforward as on the terminals. The reason for this is irregular transaction sequence. Transactions on the terminal side are always executed in an incremental order. The time of every executed transaction on a terminal $t_i^m \in B_i^m$ is always greater than the time of the previously last executed transaction $t_i^{m-1} \in B_i^{m-1}$, and smaller than the time of the first next execution time $t_i^{m+1} \in B_i^{m+1}$. For sequential states of an arbitrary account on MTD $\cdots A_i^{m-1}, A_i^m, A_i^{m+1} \cdots$, the respective sequential transactions are generated $\cdots B_i^{m-2}, B_i^{m-1}, B_i^m \cdots$ always in an incremental relation:

$$t_i^{m-2} < t_i^{m-1} < t_i^m \tag{9}$$

This is in practice achieved by terminal real-time clock synchronization with central database time.

Problems with central transaction processing emerge because transactions cannot be processed in the same time sequential order as on the terminals. Transactions on terminals are stored in batches, before they are transferred to the central processing system. Time delays between transaction initiations on terminals and central processing can cause a transaction B_i^m to arrive to the central side before transaction B_i^{m-1}. These problems are managed with algorithms for synchronization of user accounts in central processing system.

Transaction parameter chaining is useful transaction property obtained from Equations (5) and (6). We are using two parameters: *t*-time of transaction execution and *y*-unique terminal identification number for chaining two sequential successful transactions on the MTD as shown in Figure 4.

Transaction chaining mechanism confines successful transaction execution on a MTD and enables central processing system to effectively verify consistency of transactions executed on mobile terminals.

In the following section we will present algorithms for transaction processing in the central processing system.

4.1. Business Account Synchronization Mathematical Model

Two approaches are possible for account synchronization in central SQL database. First is based on a delayed synchronization, where transactions are processed after enough time has passed since their execution on terminals. This delay time assures us that all executed transactions of *i*-th user have reached the central processing system and transactions can be processed in execution-sequential order as on terminals. Such approach yields simple synchronization, but it has a drawback. The central accounts are not synchronized fast or near real-time. To achieve near real-time synchronizations, we have chosen the approach, where transactions are processed near real-time or just after they are transferred to the central processing system.

With this approach central accounts are synchronized regardless of the terminal sequential order of transactions. These way transactions can be processed as soon as they are transferred to the central processing system. Elements of an account are synchronized in the following way:

Synchronization is unnecessary for account element *I*, because it represents constant identification number. Synchronization is trivial for set of transactions

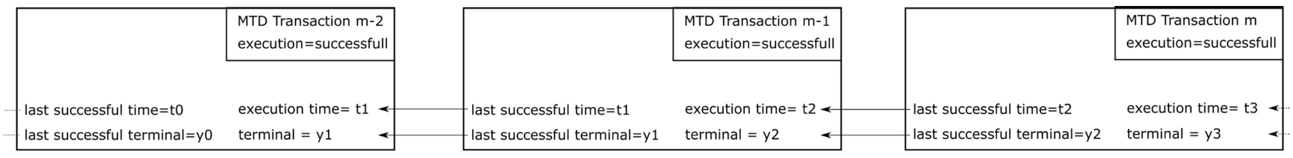

Figure 4. Sequential successful transaction chaining.

H', which are just copies of transactions H sent from terminals in batches. Account elements L' (the time) and Z' (the terminal identification) of last successfully executed transaction are also easily synchronized, by just taking values from the last successful transaction transferred to the central processing system.

Synchronization of purchased products P' is straightforward. User can purchase a product $p_n \in P$ for a use within its predefined time period. Every product has to be validated before it is used. Therefore product purchase transaction on a terminal is executed first and validations latter. Disconnection between the terminal and the central processing system might cause product purchase transaction to be delayed and product validations might arrive to the central processing system before purchase. In this case processing conflict occurs in the central processing system-validation cannot be processed, because there is no referenced product purchase. To deal with delayed product purchases, special pending status to product validations is introduced (see Section 4.2.).

4.2. Algorithm for Managing Pending Validation

The algorithm for managing pending validations is presented with flowchart on **Figure 5**. Pending validation status can be set only for arbitrary limited time. If this time has expired, the transaction system alert is set. Alert notifies transaction system administrators that validation without purchased product was executed. This indicates either a potential security breach or a system malfunction.

4.3. Algorithm for Wallet Credit Synchronization

The remaining account element to be synchronized is wallet credit. Synchronization of the wallet credit is performed with help of the auxiliary-element set X_i', which completes the account A_i' on the central side of the transaction system. The auxiliary set X_i' is presented as a record in the central database table used for synchronization. The auxiliary set contains the following elements:

$$\left(I', U', D'\right) \in X' \tag{10}$$

The first element denoted as I' is used for identification of the account being synchronized $I' \in X' = I' \in A' = I \in A$. The second element represents time and is denoted U'. It represents the synchronization time. The last element is synchronization amount denoted D'.

After a transaction B_i is executed on the terminal, it is transferred to the central side of the transaction system. Synchronization of a wallet credit value W_i' on the central side is performed with the algorithm presented with flowchart on **Figure 6**.

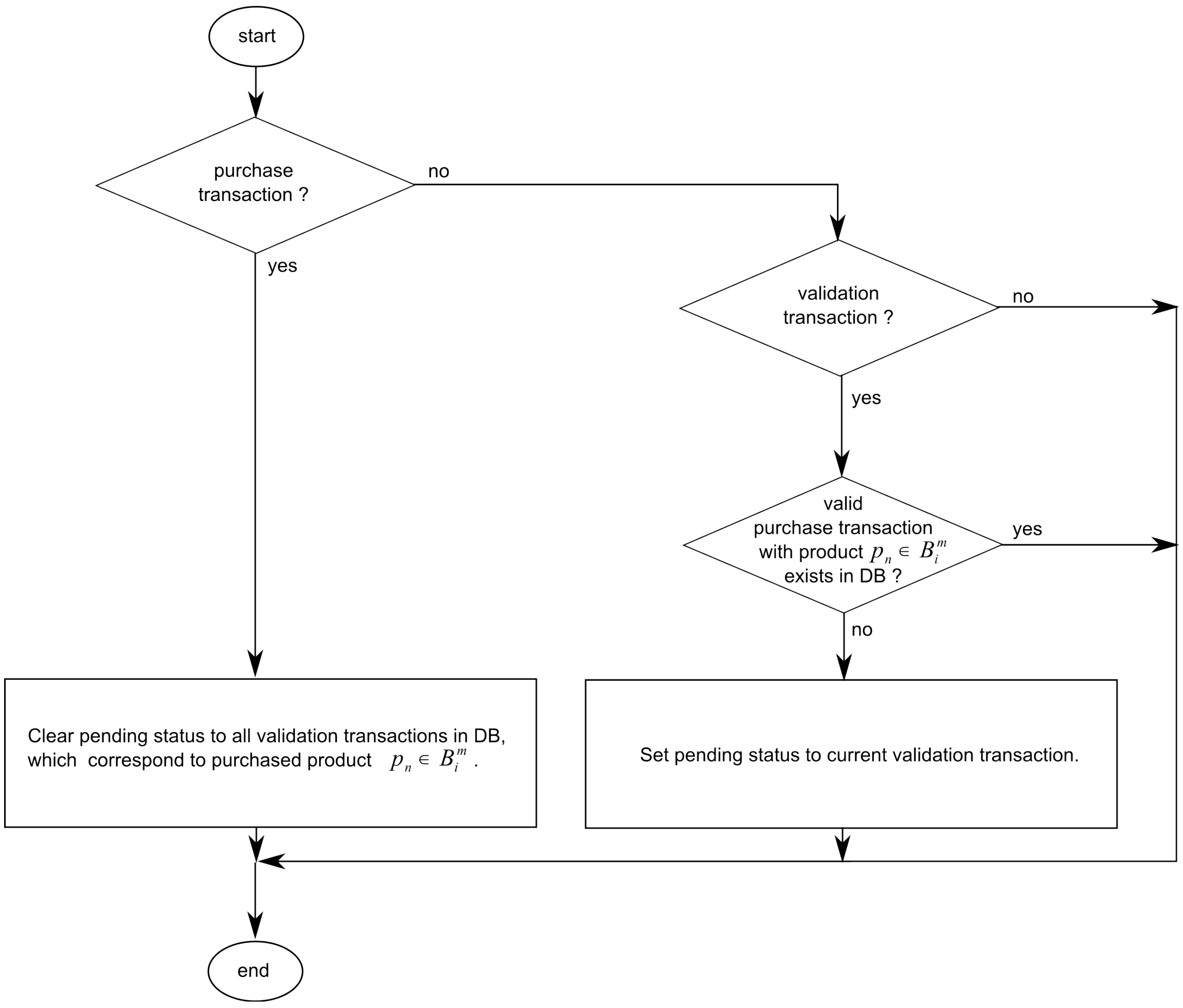
start
purchase transaction ?
no
yes
validation transaction ?
no
yes
valid purchase transaction with product $p_n \in B_i^m$ exists in DB ?
yes
no
Clear pending status to all validation transactions in DB, which correspond to purchased product $p_n \in B_i^m$.
Set pending status to current validation transaction.
end

Figure 5. Algorithm for processing pending validations.

Synchronization amount $D_i^/$ represents indicator of the synchronization status. Synchronization amount $D_i^/$ is inserted into the auxiliary set table $X^/$ only when the terminal A_i and the central side $\mathrm{A}_\mathrm{i}^/$ accounts are not synchronized. When all transactions of i-th user have been processed in the central processing system, every record $X_i^/$ in the auxiliary table has a synchronization amount $D_i^/$ value equal zero. In this case the synchronization is completed. If some values of $D^/$ exist, which are different than zero, this indicates that all transactions performed on terminals by i-th user, have not yet been processed due to transaction transfer delays. In this case the synchronization is not completed. With ideal functioning of the transaction system the auxiliary records $X^/$ always have synchronization amount values $D^/$ equal zero, after some maximal time has passed since the transaction execution on a terminal. Transaction system alert is set when arbitrary maximal time is exceeded and the respective synchronization amount $D^/$ is still not zero. This indicates either a potential security breach or a system malfunction.

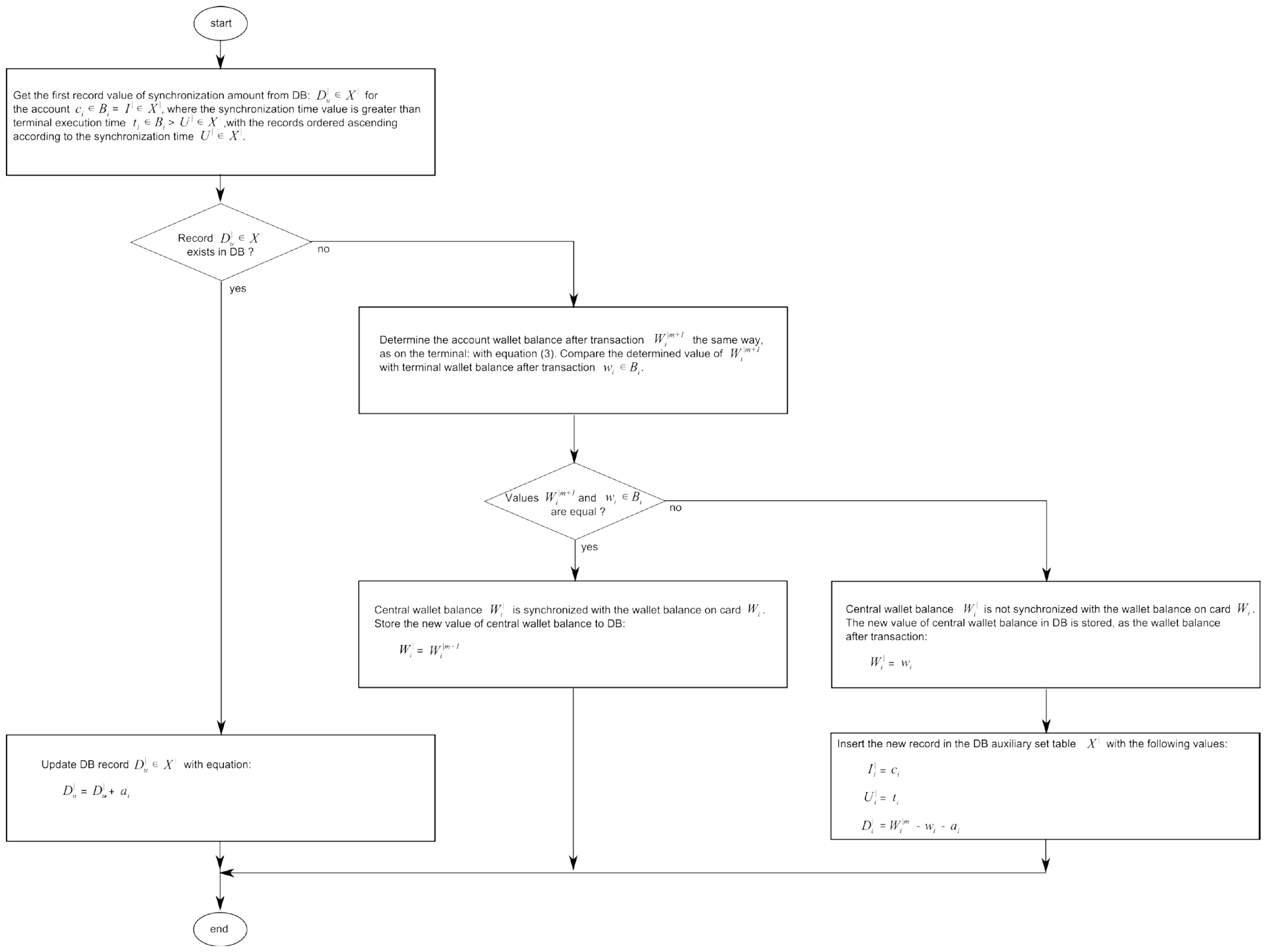
start
Get the first record value of synchronization amount from DB: $D_u^{|} \in X^{|}$ for the account $c_i \in B_i = I^{|} \in X^{|}$, where the synchronization time value is greater than terminal execution time $t_i \in B_i > U^{|} \in X^{|}$,with the records ordered ascending according to the synchronization time $U^{|} \in X^{|}$.
Record $D_u^{|} \in X^{|}$ exists in DB ?
no
yes
Determine the account wallet balance after transaction $W_i^{|m+1}$ the same way, as on the terminal: with equation (3). Compare the determined value of $W_i^{|m+1}$ with terminal wallet balance after transaction $w_i \in B_i$.
Values $W_i^{|m+1}$ and $w_i \in B_i$ are equal ?
no
yes
Central wallet balance $W_i^{|}$ is synchronized with the wallet balance on card W_i. Store the new value of central wallet balance to DB:
$W_i^{|} = W_i^{|m+1}$
Central wallet balance $W_i^{|}$ is not synchronized with the wallet balance on card W_i. The new value of central wallet balance in DB is stored, as the wallet balance after transaction:
$W_i^{|} = w_i$
Update DB record $D_u^{|} \in X^{|}$ with equation:
$D_u^{|} = D_u^{|} + a_i$
Insert the new record in the DB auxiliary set table $X^{|}$ with the following values:
$I_i^{|} = c_i$
$U_i^{|} = t_i$
$D_i^{|} = W_i^{|m} - w_i - a_i$
end

Figure 6. Algorithm for wallet credit synchronization.

4.4. Managing Disconnections on NFC Data Transmission Channel

Transaction execution between a terminal and a MTD is not always atomic because of disconnections on NFC data transmission channel, as described earlier in text. To overcome this problem we use the time of the last successfully executed transaction *L*.

Transaction B_i^m can arrive to the central side of the transaction system with the MTD commit status $S_i^m \in B_i^m$ set to *unverified*. In this case central processing system has to verify if transaction B_i^m was executed successfully or unsuccessfully. Successful transaction B_i^m can be verified with the parameter $l_i \in B_i$ -time of last successfully executed transaction using the transaction time chaining relations:

$$t_i^m = l_i^{m+1}, s_i^m = \text{successful}, s_i^{m+1} = \{successful, unverified\} \tag{11}$$

where any couple of successive transactions B_i^m and B_i^{m+1} (executed by *i*-th user) are connected with time stamp values t_i^m and l_i^{m+1}. Start time of previously executed successful transaction t_i^m is always equal to the next transactions last successful time l_i^{m+1}, if the next transactions execution status s_i^{m+1} is *successful* or *unverified*. Defined transaction time chaining can undoubtedly clarify unverified commit operation on a MTD and preserves the atomicity. Figure

7 represents the algorithm based on the transaction time chaining, which is used for managing disconnections on NFC data transmission channel.

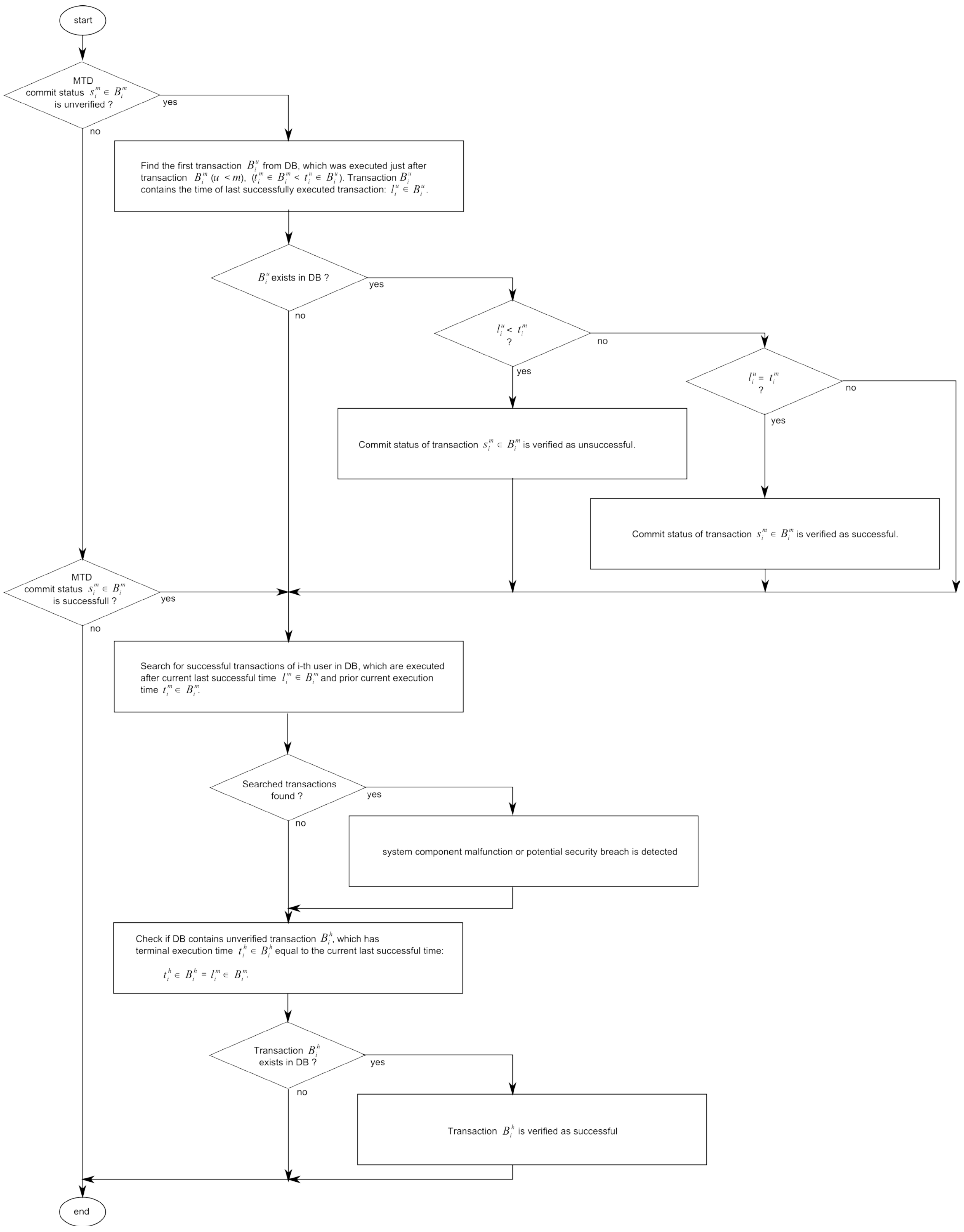
start
MTD commit status $s_i^m \in B_i^m$ is unverified ?
yes
no
Find the first transaction B_i^u from DB, which was executed just after transaction B_i^m $(u < m)$, $(t_i^m \in B_i^m < t_i^u \in B_i^u)$. Transaction B_i^u contains the time of last successfully executed transaction: $l_i^u \in B_i^u$.
B_i^u exists in DB ?
yes
no
$l_i^u < t_i^m$?
no
yes
$l_i^u = t_i^m$?
no
yes
Commit status of transaction $s_i^m \in B_i^m$ is verified as unsuccessful.
Commit status of transaction $s_i^m \in B_i^m$ is verified as successful.
MTD commit status $s_i^m \in B_i^m$ is successfull ?
yes
no
Search for successful transactions of i-th user in DB, which are executed after current last successful time $l_i^m \in B_i^m$ and prior current execution time $t_i^m \in B_i^m$.
Searched transactions found ?
yes
no
system component malfunction or potential security breach is detected
Check if DB contains unverified transaction B_i^h, which has terminal execution time $t_i^h \in B_i^h$ equal to the current last successful time:
$t_i^h \in B_i^h = l_i^m \in B_i^m$.
Transaction B_i^h exists in DB ?
yes
no
Transaction B_i^h is verified as successful
end

Figure 7. Algorithm for managing disconnections on NFC data transmission channel.

When i-th user opens an account, a MTD is initialized. With MTD initialization, the time of the last change in an account state $L_i \in A_i$ is set to the current initialization time. This enables the first transaction to be verified. By tracking the chained sequence of successful transactions, we can effectively verify the MTD commit operation. The presented algorithm assures atomic transaction execution even in the cases of *unverified* MTD data commit status on terminals.

The drawback of described method for preserving atomicity is non-real-time verification. Unverified transactions can only be verified with the next executed transaction by a user. This represents a relaxation of atomicity in the transaction system, where atomicity of disconnected transactions is preserved for the cost of transaction execution delay.

The atomicity drawback is in practice reduced to minimal occurrence with terminal feedback prompting the user to re-tap MTD when commit operation is unverified. However, the user can ignore the terminal feedback, refuse or fail to re-tap the MTD to the terminal. In this case we are relaxing the atomicity property and allow the transaction execution to be delayed until the user executes next transaction.

4.5. Control and Security

Transaction time chaining was utilized for managing disconnections on NFC data transmission channel. It is also used for control and security purpose, in combination with the terminal identification chaining.

Practical concern in the mobile transaction system is random malfunction, when some random terminal didn't send an executed transaction to the central processing system. To verify the transaction execution, we need reliable central control on such failures. This is achieved with transaction time chaining and terminal identification chaining.

A transaction contains the element of terminal identification $y \in B$ and the element of terminal identification of the last successfully executed transaction $z \in B$. Any couple of successive transactions B_i^m and B_i^{m+1} executed by i-th user are chained with terminal identifications as:

$$y_i^m = z_i^{m+1}, s_i^m = successful, s_i^{m+1} = \{successful, unverified\} \quad (12)$$

where terminal identification of previous successful transaction y_i^m is always equal to the next transactions last terminal identification z_i^{m+1}, if the next transactions execution status s_i^{m+1} is *successful* or *unverified*.

Automated central control on transaction system is based on detection of consistency failures during transaction processing such as:

- missing transactions during account synchronizations,
- missing or invalid timestamps in transaction time chaining defined with Equation (11),
- other invalid transaction parameters.

Detected irregularities can be supplemented with the terminal identification chaining, defined with Equation (12). With terminal identification chaining the malfunctioned terminal can be rapidly detected, even if the terminal is disconnected from the central processing system.

Security is another important concern in the mobile transaction system. Mobile transaction system components such as terminals and MTD's are particularly vulnerable, because mobility enables offline transaction execution without connection to the central processing system. Moreover, potential attackers have easy physical access to these mobile components. Generally it is hard to predict all possible attack methods on MTDs [26] [27] therefore we are proposing a consistency verification method for fraud detection. Comprehensive transaction system security is the topic beyond the scope of this text. We are presenting a basic approach for security hardening in the mobile payment system.

Encryption represents the basic defense against attacks on mobile transaction system. Mobile transaction system data is protected with encryption of accounts on MTDs, secured transaction execution on terminals and encrypted data transfer between terminals and central processing system. If an attacker manages to break the encryption, it would be trivial to execute unauthorized transactions. Breaking encryption on a MTD or terminal could lead to:

- unauthorized electronic wallet credit modifications,
- unauthorized electronic product purchases,
- copying and duplication of a MTD account.

Presented central transaction processing mechanisms offer enhanced security with rapid detection of potential data breaches on the terminal side of the transaction system. Security breach on the terminal side can be detected with defined consistency verification during account synchronization or managing NFC disconnections on the central side. Every inconsistency between terminal and central side of an arbitrary account represents potential security breach. For example: unauthorized product purchases can be detected with defined synchronization of purchased products, unauthorized wallet balance modification can be detected with the wallet credit synchronization. Copying or duplication of a MTD account can be detected with the following transaction consistency relation:

$$l_i^u \neq l_i^m,\ u \neq m,\ s_i^m \in B_i^m = \text{successful}, s_i^u \in B_i^u = \text{successful} \tag{13}$$

where any couple of successful transactions B_i^u and B_i^m, executed by *i*-th user, always have different values of last successful time. Copy and duplication of account on a MTD is detected by comparing last successful time of processed transaction with last successful time of any previous transactions executed by *i*-th user and stored in the central database. If two successful transactions on a MTD with the same last successful time are detected, this indicates potential unauthorized copying and duplication of a MTD account. Such transaction consistency confinement provides the effective detection of miscellaneous attacks on the terminal side, and increases attack difficulty. This is because undetectable attack requires simultaneous account data modification on the terminal side and on the central side of the transaction system.

5. Implementation Case City Card Urbana

In this section we will present practical operational data of the defined transaction processing model. The presented processing model was implemented in real big mobile transaction system called "City card Urbana" or simply Urbana. Results are obtained from real transaction data gathered from Urbana in December 2015, which represents typical operation. Urbana is used in the city of Ljubljana as a stored value card business transaction system for purchasing small value public city transportation services, city parking services, public library services, city tourist services, etc. Urbana contains approximately 860,000 active user accounts and 1600 mobile terminals.

The chart in Figure 8 shows daily number of successful transactions. On a typical work day Urbana exceeds 200,000 successful transactions. While on weekends, national holidays, school holidays there are notably smaller number of transactions. This is expected daily distribution, since city services are used more frequently during work days.

We performed the measurements on the transaction system operational data in December 2015, which are presented in Table 1.

These measurements are:

- Average frequency of disconnections between terminals and central processing system during transmission of transaction batches.
- Average daily number of disconnections on NFC data transmission channel.
- Averagetime for transaction execution on terminals for contactless smartcards.
- Average time for transaction execution on terminals for NFC compatible mobile phones.

Urbana terminals use GPRS mobile data transmission channel for transferring transaction data to the central processing system. Frequency of disconnections in Table 1 is calculated as a ratio between number of disconnected and number of successfully transmitted transaction batches from terminals to the central processing system. In December 2015 there were 1,319,516 successfully transmitted transaction batches and 26,015 disconnections. The measurement of 0.02 represents approximately one disconnection on 50 successfully transmitted transaction batches. Defined algorithms for business account synchronization successfully managed all disconnected transactions assuring ACID and FUCS properties.

On work week days Urbana receives averagely 226 NFC disconnections, which affect transaction atomicity discussed earlier. Although the number of such disconnections is relatively small in comparison to the total daily number of successful transactions (more than 200,000), the preservation of atomicity is essential for the transaction system. Processing mechanisms for managing NFC disconnections enable preservation of atomicity; however atomicity is preserved with relaxation, where transaction execution on the central side is delayed until a user executes next transaction. Table 2 presents number of these delays in accordance to time for NFC disconnections that occurred during December 2015.

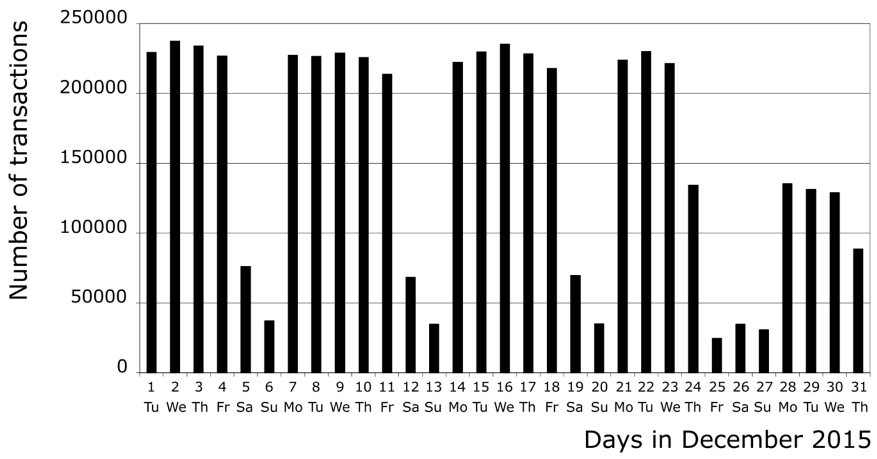

Figure 8. Daily number of successful transactions in December 2015.

Table 1. Measurements in Urbana for December 2015.

Average frequency of batch transmission disconnections	0.02
Average daily number of NFC disconnections	226
Average time for transaction execution with contactless smartcards (in seconds)	0.35
Average time for transaction execution with NFC compatible mobile phones (in seconds)	1.0

(Measurements were performed on work week days-without weekends and national holidays).

Table 2. Delays for managing disconnections on NFC during December 2015.

Number of disconnected transactions	Verification time in days (24 hours) since transaction execution
3953	less than 1 day
305	from 1 to 2 days
69	from 2 to 3 days
56	from 3 to 4 days
38	from 4 to 5 days
28	from 5 to 6 days
17	from 6 to 7 days
30	from 7 to 8 days
18	from 8 to 9 days
4	from 9 to 10 days
257	more than 10 days

(Number of all disconnected transactions = 4775).

Table 2 shows that majority of disconnections on NFC are resolved within one day since the execution of disconnected transaction. The disconnected transaction is verified with the next transaction after NFC disconnection. Resultsin Table 2 indicate that majority of users use the transaction system regularly every day. Table 2 also shows that 257 disconnected transactions need more than 10 days for verification. Some of those transactions might never get verified or verification time will be very long. In the worst case scenario, some transaction on a MTD gets disconnected and the same MTD is never used again. Such transaction will never be verified.

FUCS requirements in Urbana are ensured with the defined processing architecture. Transaction execution is available on mobile terminals regardless of the availability of mobile transmission channel. Transactions are executed fast and user friendly. Generally a transaction is executed in a short time after the user just puts a MTD in front of a terminal: currently in approximately one third of a second for contactless smartcards and one second for NFC compatible mobile phones.

6. Conclusions

We have defined a holistic transaction processing model, with algorithms for mobile account synchronization, verification of transactions, establishing enhanced transaction system control and security. Defined processing model is capable of resolving common problems in mobile micro payment touch-and-go transaction systems. These problems are offline transaction executions, transaction atomicity failures, central control on mobile transaction system components, and supervision on potential security risks.

The processing model was tested in real mobile transaction system environment, where it operates very well. Presented transaction chaining mechanism was in practice proven as a good solution for ensuring requirements in real mobile payment system. FUCS properties of the payment processing system can be further improved in the future, by adding additional parameters for transaction chaining such as: transaction serial number on a terminal or transaction serial number on a MTD. A challenge remains in further reduction of NFC transaction disconnections and in decreasing delays for managing these disconnections.

References

[1] Haerder, T. and Reuter, A. (1983) Principles of Transaction Oriented Database Recovery. *ACM Computing Surveys*, **15**, 287-317. https://doi.org/10.1145/289.291

[2] Bernstein, P.A. and Newcomer, E. (2009) Principles of Transaction Processing 2nd Edition, Morgan Kaufmann, Burlington.

[3] Le, H.N. (2006) A Transaction Processing System for Supporting Mobile Collaborative Works. Doctoral Dissertation, Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim.

[4] Weikum, G. and Vossen, G. (2002) Transactional Information Systems, Theory, Algorithms, and the Practice of Concurrency Control and Recovery. Morgan Kaufmann, Burlington.

[5] Chen, J.J. and Adams, C. (2004) Short-Range Wireless Technologies with Mobile Payments Systems. 6*th International Conference on Electronic Commerce*, Delft, 25-27 October 2004, 649-656. https://doi.org/10.1145/1052220.1052302

[6] Dewan, S.G. and Chen, L.-D. (2005) Mobile Payment Adoption in the USA: A Cross-Industry, Cross-Platform Solution. *Journal of Information Privacy & Security*, **1**, 4-28. https://doi.org/10.1080/15536548.2005.10855765

[7] Finkenzeller, K. (2010) RFID Handbook: Fundamentals and Applications in Con-

tactless Smart Cards, Radio Frequency Identification and Near Field Communication. 3rd Edition, John Wiley & Sons, Hoboken. https://doi.org/10.1002/9780470665121

[8] Carr, M. (2007) Mobile Payment Systems and Services: An Introduction. Mobile Payment Forum.

[9] Ondrus, J. and Pigneur, Y. (2007) An Assessment of NFC for Future Mobile Payment Systems. *ICMB International Conference on the Management of Mobile Business*, Toronto, 9-11 July 2007, 43-53. https://doi.org/10.1109/ICMB.2007.9

[10] Ondrus, J. and Pigneur, Y. (2006) Towards a Holistic Analysis of Mobile Payments: A Multiple Perspectives Approach. *Electronic Commerce Research and Applications*, **5**, 246–257. https://doi.org/10.1016/j.elerap.2005.09.003

[11] Klampfer, S. and Chowdhury, A. (2015) The Proposed Planning Method as a Parallel Element to a Real Service System for Dynamic Sharing of Service Lines. *ISA Transactions*, **57**, 403-417. https://doi.org/10.1016/j.isatra.2015.02.010

[12] Svečko, J., Kotnik, B., Mezgec, Z. and Chowdhury, A. (2010) The Margento Automated Fare Collection System Solution for Public Transport. *Proceedings of the 7th International Conference on Logistics & Sustainable Transport*, Maribor, 24-26 June 2010.

[13] Walborn, G.D. and Chrysanthis, P.K. (1997) Pro-Motion: Management of Mobile Transactions.11*th ACM Annual Symposium on Applied Computing*, San Jose, 28 February-2 March 1997, 101-108. https://doi.org/10.1145/331697.331718

[14] Madria, S.K. and Bhargava, B. (1998) A Transaction Model for Mobile Computing. *Proceedings of the International Database Engineering and Applications Symposium*, Cardiff, 8-10 July 1998.

[15] Keller, A.M., Densmore, O., Huang, W. and Razavi, B. (1998) Zippering: Managing Intermittent Connectivity in DIANA. *ACM/Kluwer Journal on Mobile Networks and Applications*, **2**, 357-364. https://doi.org/10.1023/A:1013661523328

[16] Pedregal-Martin, C. and Ramamritham, K. (2002) Support for Recovery in Mobile Systems. *IEEE Transactions on Computers*, **51**, 1219-1224. https://doi.org/10.1109/TC.2002.1039847

[17] VanderMeer, D., Datta, A., Dutta, K., Ramamritham, K. and Navathe, S.B. (2003) Mobile User Recovery in the Context of Internet Transactions. *IEEE Transactions on Mobile Computing*, **2**, 132-146. https://doi.org/10.1109/TMC.2003.1217233

[18] Ferreira, M.C., Dias, T.G. and Falcao e Cunha, J. (2014) Design and Evaluation of a Mobile Payment System for Public Transport: The MobiPag STCP Prototype. 4*th International Conference on Mobile Services, Resources, and Users*, Paris, 20-24 July 2014.

[19] Rodrigues, H., José, R., Coelho, A. and Melro, A., Campos Ferreira, M., Falcao e Cunha, J., Pimenta Monteiro, M. and Ribeiro, C. (2014) MobiPag: Integrated Mobile Payment, Ticketing and Couponing Solution Based on NFC. *Sensors*, **14**, 13389-13415. https://doi.org/10.3390/s140813389

[20] Stevens, W.R. (1994) TCP/IP Illustrated, Volume 1: The Protocols. Addison-Wesley, Upper Saddle River.

[21] Halonen, T., Romero, J. and Melero, J. (2003) GSM, GPRS and EDGE Performance Evolution towards 3G/UMTS. 2nd Edition, John Wiley & Sons, Hoboken. https://doi.org/10.1002/0470866969

[22] Heine, G. and Sagkob, H. (2003) GPRS: Gateway to Third Generation Mobile Networks. Artech House Publishers, Norwood.

[23] Sanchez, J. and Thioune, M. (2007) UMTS, ISTE.

[24] Codd, E.F. (1970) A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, **13**, 377-387. https://doi.org/10.1145/362384.362685

[25] Date, C.J. (1995) An Introduction to Database Systems. 6th Edition, Addison Wesley, Upper Saddle River.

[26] Đurić, Z., Marić, O. and Gašević, D. (2007) Internet Payment System: A New Payment System for Internet Transactions. *Journal of Universal Computer Science*, **13**, 479-503.

[27] Pasquet, M., Reynaud, J. and Rosenberger, C. (2008) Secure Payment with NFC Mobile Phone in the Smart Touch Project. *Proceedings of International Symposium on Collaborative Technologies and Systems CTS*, Irvine, 19-23 May 2008.