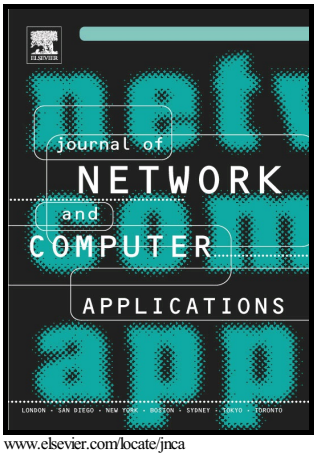
# Trustworthy and Effective Person-to-Person Payments over Multi-hop MANETs

Barbara Carminati, Elena Ferrari, Ngoc Hong Tran




www.elsevier.com/locate/jnca

PII: S1084-8045(15)00283-0
DOI: http://dx.doi.org/10.1016/j.jnca.2015.11.011
Reference: YJNCA1508

To appear in: *Journal of Network and Computer Applications*

Received date: 29 April 2015
Revised date: 3 September 2015
Accepted date: 24 November 2015

Cite this article as: Barbara Carminati, Elena Ferrari and Ngoc Hong Tran Trustworthy and Effective Person-to-Person Payments over Multi-hop MANETs *Journal of Network and Computer Applications* http://dx.doi.org/10.1016/j.jnca.2015.11.011

# Trustworthy and Effective Person-to-Person Payments over Multi-hop MANETs

Barbara Carminati[1], Elena Ferrari[1], Ngoc Hong Tran[1]*

{*firstname.lastname*}*@uninsubria.it*

*DiSTA, University of Insubria*

*Via Mazzini, 5, Varese, Italy*



**Abstract**

Due to the rapid development of mobile technologies, nowadays mobile devices are not expensive and almost every person can easily possess a mobile device. This fact boosts investments in mobile applications, among which are the person-to-person mobile payment applications. These applications are pretty sensitive in that they are related to monetary transactions, thus involving strict security and privacy requirements. To this purpose, we propose a secure protocol leveraging online social network connections to help users enforce their trust preferences locally to make a money transfer. The protocol exploits mobile ad-hoc network as a communication means. To improve the network performance by still preserving data security and user privacy, we also propose some optimization strategies to decrease the number of tokensets sent over mobile ad-hoc network. The experimental results demonstrate the effectiveness of our proposal.

*Keywords:* Mobile person-to-person payment, trust preferences, mobile ad-hoc network, homomorphic encryption.

*Corresponding author

*Email address:* ngoc.tran@uninsubria.it (Ngoc Hong Tran[1]*)

[1]Authors are listed in alphabetical order according to authors' surnames.

## 1. Introduction

Hardware techniques for mobile devices have been developed with a rapid speed[2,3] with the result of reducing the cost of mobile devices. Today owning a mobile device is no longer a problem. Consequently, the market of mobile applications has been growing up tremendously to take advantage of this increasing number of mobile users[4,5]. Among mobile applications, mobile-based person-to-person (P2P) payments have been rapidly growing. As Gartner forecasts[6], in 2016 there will be 448 million mobile payment users in a market that worths 617 billion USD. Despite the success of these new digital P2P payment methods, we believe that, to fully enable this rise of digital wallets, relevant security and privacy challenges need to be addressed. At this purpose, in [1] we have proposed a secure protocol, namely SmartPay, to exploit the available social network connections to support users in making decisions behind mobile P2P payments. In particular, SmartPay leveraged social connections to help the payers, i.e., those people who give credit to someone, to judge if the person asking money can be considered trusted. More precisely, SmartPay exploits social networks to verify how payer and payee are connected in the social graph. Based on this information and the payer trust preferences, SmartPay suggests whether the money transfer should be authorized or not. A key aspect of SmartPay is the decentralized protocol exploited to gather information on the social path connecting the payer and the payee, which is then encoded into a data structure, called *tokenset*. Furthermore, SmartPay exploits a light cryptographic algorithm, namely binary Elliptic Curve Cryptography (ECC) algorithm [2] to protect the privacy of user relationship information in the traversed path. This makes SmartPay able to privately aggregate information on the traversed paths (i.e., depth, trust, relationship type), without revealing any information on traversed edges. However,

[2] http://www.dayintechhistory.com/news/evolving-cell-phone-1973-2014/.
[3] http://www.businessinsider.com/the-evolution-of-the-cell-phone-2013-1?op=1.
[4] http://wearenative.in/news/smartphones-still-room-for-growth/
[5] http://www.smartinsights.com/mobile-marketing/app-marketing/mobile-app-statistics/
[6] http://www.gartner.com/newsroom/id/2028315

in [1] we did not cope with the communication means for exchanging tokensets among SmartPay users. It is only assumed that the communication means is Internet (e.g., using emails or sockets). In this paper, we want to explore alternative communication means. In particular, we propose to exploit Mobile Ad-hoc NETwork (MANET). MANET has been investigated since the 1990's and recent predictions about the future research trends in MANET determined that it has potentials to be continuously developed [3], [4]. Since MANET's properties fit the realistic requirements of many scenarios, such as availability, cost saving, self-organized and infrastructure-less architecture, the applicability of MANET is extremely large. For instance, MANET applications have been developed on tactical networks, emergency, as well as education, context aware services, entertainment, military services [5]. Additionally, several applications have been deployed over MANET. For instance, available locations of users on their mobile devices have been exploited, among others, by Foursquare[7] and Gowalla[8], which are location-based social networking services for mobile devices, and by *Last.fm Festival*[9] that suggests a list of music festivals to users near the event locations. As further relevant examples, *TerraNet*[10] supports mobile phone calls without a connection through a cell tower; Mobile Chedar [6] is a middleware prototype with a mobile peer-to-peer learning environment application using Bluetooth; AdSocial [7] is a software platform supporting social network applications in ad hoc networks targeting small-scale scenarios, such as friends playing a game on the train or co-workers sharing calendar information, as well as, conference participants establishing voice-video calls, chat, or play games. In addition, we can cite MobiClique [8], a mobile social software, that allows people to maintain and to extend their online social networks through opportunistic connections between neighboring devices, and What's Up [9] an application providing spontaneous social networks in ad hoc networks, such as conferences and expositions.

[7] https://foursquare.com/about
[8] http://gowalla.com/
[9] http://www.last.fm/festivals
[10] http://terranet.se/

As witnessed by the above mentioned services, plenty of MANET applications have been deployed successfully, and particularly for scenarios where local networks with a high density of users are available (e.g., a conference room, library, supermarket, stadium, park, university campus, company buildings). This typical MANET scenario fits well the one in which P2P mobile payment applications can be used. Hence, we strongly believe that there is a need of deploying P2P payments over MANET. For this reason, in this paper, we investigate how to deploy SmartPay over MANET. To our knowledge, this work is the first one exploiting social network relationships for P2P payments over MANET.

Despite the benefits, this new communication means has open challenges that need to be addressed [4] [10]. Among them, the most relevant are: *(1) privacy and security*, *(2) energy efficiency*, and *(3) network performance*. The deployment of SmartPay over MANET has to cope with all these issues. Regarding privacy and security, the exchange of tokensets through MANET rises new challenges w.r.t. of those investigated in [1]. Indeed, according to the MANET protocol, a tokenset might be forwarded through several mobile users before reaching the destination. Those users are not involved in the path traversal and thus do not have to infer any information about the payment transaction as well as aggregate relationship information. In addition, in [1] the payer identity was not kept private, as the proposed solution exploits the payer's public key. In contrast, in this paper we present a solution suitable to MANET communication that provides a more secure protocol than the one in [1]. Regarding issue (2), we use cryptographic algorithms on elliptic curves, that is, binary ECC [2] and Elliptic Curve Digital Signature Algorithm (ECDSA) [11]. These binary cryptographic algorithms make the mobile devices able to perform efficient computations and consume less energy. With respect to (3), SmartPay adopts a strategy inspired by $k$-anonymity to optimize network performance. However, selecting a reasonable value of $k$ to make the network consumption effective is not easy. In particular, in case $k$ is large, network performance is similar to the one of broadcasting techniques, but privacy is preserved more strongly. In case $k$ is small, the bandwidth consumption is low but privacy

cannot be preserved effectively. To overcome this problem, in this paper, we propose an optimization strategy exploiting secure comparison algorithms so as to reduce the flooding of tokensets in MANET. In particular, our algorithm allows to evaluate encrypted relationship information inside a tokenset to be forwarded in the network.

The remainder of the paper is organized as follows. Background knowledge is introduced in Section 2. Section 3 describes related work. An overview of SmartPay is reported in Section 4. Section 5 presents issues and possible solutions to deploy SmartPay over MANET. Section 6 describes the expanded SmartPay protocol over MANET. Section 7 presents the optimization strategy. Experimental results are reported in Section 8. Security properties are discussed in Section 9. Finally, Section 10 concludes the paper.

## 2. Background

### *2.1. MANET (Mobile Ad-hoc NETwork)*

MANET is a collection of many devices equipped with wireless communications and networking capabilities that make them able to communicate with each other [12]. This can be done through a direct communication, if the destination of the message is within the *radio range* of the sender. In general, the radio range depends on the communication means. For instance, a Bluetooth device has a radio range of 10m, whereas a Wi-Fi device has a radio range of 100m [13]. In case two devices are out of their radio ranges, a protocol is initialized aiming at forwarding the message through intermediate nodes until reaching the final destination. For this purpose, each device, hereafter *node*, keeps track of the set of neighbors that fall into its radio range. Since nodes might leave or enter the other's radio range, the MANET topology can change in a very dynamic way. These changes might split the network into different subnetworks, called partitions, among which there might not exist a possible routing path. To support the communication among nodes in different partitions, large-scale MANETs make use of connectors. These are proxies responsible for forwarding

messages from a partition to another. A connector can be a fixed device (such as an Access Point (AP), a server, or a switch that functions as a gateway) [14], or mobile nodes/robots [15]. When a node joins the network, it receives from its neighbors information about available connectors. When a node wants to transmit a message to a node in another partition, it forwards the message to connectors in its connectors list.

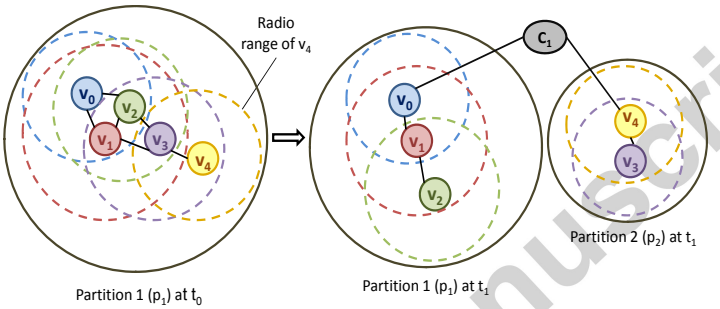



Figure 1: MANET partitions and radio ranges at different times

**Example 1.** Figure 1 illustrates a MANET at two different time instants, i.e., $t_0$ and $t_1$. Each node has a radio range, denoted by the surrounding dotted circle. Partitions are denoted by a solid circle. When a node enters in another dotted circle, the two nodes can communicate directly using Wi-Fi standards, i.e., IEEE 802.11. At time $t_0$, $v_0$ can directly contact $v_1$ since it is in $v_1$'s radio range and vice versa. This direct communication is reported as an edge connecting $v_0$ and $v_1$, denoted as $e(v_0, v_1)$. In contrast, $v_0$ can contact $v_3$ indirectly by a protocol forwarding the message through a routing path, e.g., $v_0 \rightarrow v_2 \rightarrow v_3$. As reported in Figure 1, at time $t_0$ all mobile nodes $v_0, v_1, v_2, v_3, v_4$ set up a partition, say $p_1$, since all of them can communicate with each other directly or indirectly. Figure 1 reports that $v_3$ and $v_4$ leave $p_1$ at time $t_1$ and form the second partition, say $p_2$. The two partitions, i.e., $p_1$, $p_2$, communicate with each other through the connector, that is, $C_1$.

### 2.2. Homomorphic Encryption

Homomorphic encryption [16] is a form of encryption where all computations are carried out on ciphertexts. In our protocol, we use the homomorphic Elliptic Curve Cryptography ($ECC$) [2] based on the binary finite field $F_2^m$. As such, we assume that each node $v_i$ using $SmartPay$ over MANET has a pair of a public key, i.e., $(k_i \cdot B)$ and a private key, i.e., $k_i$, where $B$ is the base point on the elliptic curve ($E$). Let us remind $ECC$ encryption/decryption operations by assuming that Alice wishes to send a message $M$ to Bob, and Bob has a pair of keys ($k_{Bob}$, $k_{Bob} \cdot B$). To encrypt $M$, Alice generates $E(M) = (Y1, Y2)$, $Y1 = r_{Alice} \cdot B$, $Y2 = M + r_{Alice} \cdot (k_{Bob} \cdot B)$, where $r_{Alice}$ is a random integer generated by Alice. Given $E(M)$ Bob uses his private key $k_{Bob}$ to decrypt $M$ as follows: $D(E(M)) = Y2 - k_{Bob} \cdot Y1$.

## 3. Related work

In recent years, many P2P payment systems have been deployed. For instance, Bitcoin [17] and Zerocoin [18], which are decentralized e-cash systems. However, it has been showed that Bitcoin has some privacy issues [19] due to the adopted pseudonym technique [20] used to avoid the disclosure of real user identities. Zerocoin proposes a crypto-based solution to guarantee the anonymity of Bitcoin transactions. However, the goal of SmartPay is different from the one of Bitcoin and Zerocoin in that SmartPay does not cover an e-cash transaction. Rather, it helps to judge whether a payment transaction should or should not be authorized based on the relationship path between payer and payee.

[21] proposes a secure electronic payment system over MANET, called PayFlux. PayFlux applies SPKI (Simple Public Key Infrastructure) to improve the performance of processing encryption/decryption. However, PayFlux only works with direct connections (i.e., one hop) between the payer and the payee. In [22], [23], authors proposed a secure payment protocol for a vehicular ad hoc network between a client and a merchant. [22] uses symmetric cryptography whereas [23] exploits ECC and hash functions. These aim to improve the performance, while

still preserving user privacy and data security. However, [21] [22] [23] focus on direct connections and e-cash transfers, whereas, our work considers both direct and indirect connections and makes trust preference enforcement available for users to decide a money transfer.

[24] addresses P2P payments in multiple hop wireless networks as well. Each node in the protocol is assigned a trust value. A packet of payment request is routed through the network according to these trust values. It implies that the packet moves through nodes having the highest trust values. After every successful transaction, these trust values are updated. Whereas, our protocol works on relationship information of every user retrieved from their social networks. Beside, [24] deployed their decentralized-based framework in wireless networks, however, they still need a third party for data authentication. In contrast, we do not need the support of any third party, our proposal is fully decentralized. Moreover, they use only hash function for data authentication while our proposal makes data more confidential by using both hash functions and encryption algorithms. Additionally, in our proposal intermediate nodes do not know the source and the destination node to avoid private information leakage from a malicious intermediate node while the intermediates in [24] can know the source and the destination nodes.

In [25], a local secure scheme supports nodes in evaluating the trust between two nodes, node $i$ and node $j$, in an ad hoc network, when node $i$ requests node $j$ for a communication. In particular, node $i$ retrieves a set of common connected nodes of node $i$ and node $j$, then requests common nodes to send to node $i$ their reputation recommendation on node $j$. To retrieve this intersection and to let each node in the protocol unaware of the set of nodes of the other side, [25] adopted the homomorphic Paillier's encryption and polynomial intersection calculation. In contrast, we address the more general issue of calculating trust between two indirect nodes and we apply the homomorphic Elliptic Curve Cryptography algorithm to aggregate the trust values on a path.

## 4. SmartPay Overview

### *4.1. Trust-driven mobile person-to-person payment*

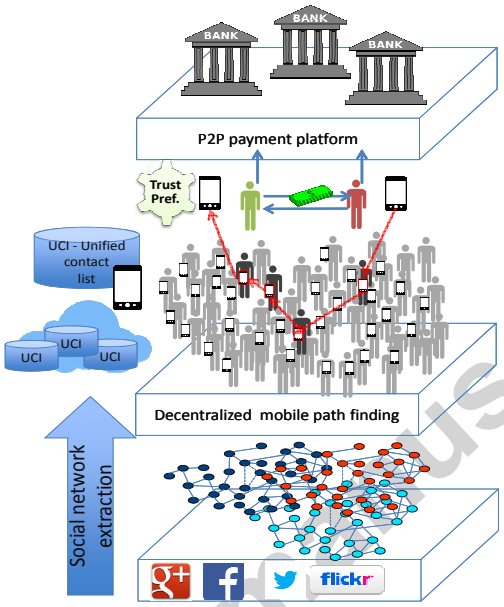



Figure 2: Trust-driven mobile Person-to-Person payments

In [1], we proposed a light cryptographic protocol for mobile P2P payment systems, called SmartPay. SmartPay is able to preserve user privacy, to protect data security and to optimize the performance for mobile devices. As introduced in Section 1, SmartPay operates over available social networks. An overview of the architecture is presented in Figure 2. SmartPay periodically synchronizes with user's social network accounts so as to extract their contact lists. These lists are merged into a unique list, called Unified Contact List (UCL), which can be either locally stored or stored into a cloud public storage service. SmartPay also provides users with the possibility of further classifying their relationships by specifying, for each of them, the type of the relationship, e.g., parent, colleague, classmate, etc., defined according to a standard vocabulary, like FOAF [27],[11]

[11] We uses FOAF in this work since FOAF is an ontology dictionary of named properties

and a trust value, representing the strength of the relationship.

Payer's trust preferences are expressed as conditions on the relationship the payee has to have in order to be considered enough trusted by the payer. Hence, each SmartPay user locally specifies one or more *trust preferences*, defined as $TP = (P, TC)$, where $P$ indicates the role (i.e., *payer* or *payee*) of the considered user, whereas $TC$ denotes a set of trust conditions $tc$. Each trust condition $tc$ is defined as a tuple $(rt, d_{max}, t_{min})$ describing the trust requirements. More precisely, $tc$ states that between the trust preference owner and the other party there should exist a direct or indirect relationship[12] of type $rt$ with a maximum depth $d_{max}$ and a trust value greater than or equal to $t_{min}$.[13]

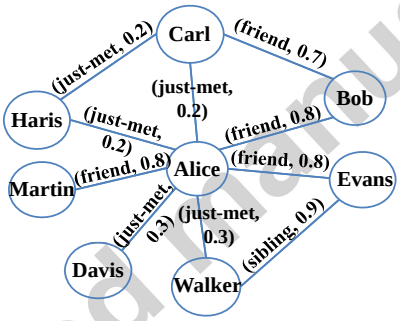



Figure 3: An example of a portion of a social graph

**Example 2.** Consider the social graph depicted in Figure 3, where edges are labeled with the trust level and the type of the relationship existing between the connecting nodes. Assume that *Evans* sets a trust preference *TP = (payer, Friend, 3, 0.6)*, stating that he accepts to be *payer* for users that are connected with him by a direct or an indirect relationship of type *"Friend"* with maximum

and classes of social relationship using W3C's RDF technology, an open Web standard, as well as, is promoted for decentralized approaches [28].

[12] An indirect relationship connects two nodes by a path including more than one edge, all with the same relationship type.

[13] Literature offers several algorithms to compute the trust value between two indirectly connected nodes [29]. Here, we assume that the trust value of an indirect relationship is computed as multiplication of the trust values of each single edge in the path connecting the two nodes.

depth 3, and minimum trust 0.6. Suppose that Bob and Haris ask Evans a small amount of money. Let us first consider the case of Haris. As Figure 3 reports, between Bob and Haris there does not exist any indirect relationship of type "Friend", since there is no connecting path between them with all edges labeled as Friend. As such, Haris does not satisfy Evans trust preference, and therefore the money transfer is denied. In contrast, between Bob and Evans there exists an indirect friend relationship (i.e., the path Bob $\rightarrow$ Alice $\rightarrow$ Evans in Figure 3). Moreover, the trust and the depth of the path are respectively 0.64 and 2. Hence, Bob satisfies the trust preference defined by Evans.

Trust preference evaluation at the payer side requires to discover a path connecting the two involved users. Because the relationship information is locally stored in every node, in [1] we proposed a distributed protocol executed through a collaboration among the traversed nodes in the social graph to find a path from the payer to the payee. When finding a path, the protocol collects the relationship information (i.e., relationship type, trust, depth) as well. The collected information is stored locally in a data structure, called *tokenset*. In [1], the protocol is assumed to be launched by the payer for the sake of simplicity. The tokenset is first sent to the payer's contacts at which it continues to be aggregated and propagated. This step is repeated until the tokenset reaches the payee. In the next section, we introduce the SmartPay protocol in further details.

### *4.2. Decentralized path finding protocol*

Let us first consider the path trust value and assume that a node $v_i$ which partially constitutes a traversed path $p = \{v_0, \cdots, v_i, \cdots, v_p\}$ from the payer (i.e., $v_0$) to the payee (i.e., $v_p$). We recall that in [1] every node in the protocol knows the public key parameters of the payer's needed for the encryption. A summary of the protocol in [1] is presented in what follows:

1. Once the payer receives a request of a money transfer from a payee, the payer initializes a tokenset including distinct tokens, one for each of the

path property to be verified (i.e., trust token, and relationship type token, respectively). The payer then forwards the tokenset to his/her contacts.

2. When an intermediate node $v_i$ receives the tokenset, it updates every token in a privacy preserving way to add the information on depth and trust of the edge being traversed. As an example of trust aggregation done on the social path connecting node $v_0$ and node $v_i$, that is, $E_{(0)(i)} = \Sigma_{\alpha=0}^{i-1} E_{(\alpha)(\alpha+1)}$ where $E_{(\alpha)(\alpha+1)}$ is the encryption of the trust value between node $v_\alpha$ and node $v_{\alpha+1}$, and $E_{(\alpha)(\alpha+1)} = (r_{(\alpha)(\alpha+1)} \cdot B, P_{Tr_{(\alpha)(\alpha+1)}} + r_{(\alpha)(\alpha+1)} \cdot k_{v_p} \cdot B)$ where $r_{(\alpha)(\alpha+1)}$ is a random number generated locally at node $v_\alpha$ and $P_{Tr_{(\alpha)(\alpha+1)}}$ [14] is a point on the elliptic curve mapped from a trust value between node $v_\alpha$ and node $v_{\alpha+1}$, i.e., $Tr_{(\alpha)(\alpha+1)}$. Thus, $E_{(0)(i)} = (\Sigma_{\alpha=0}^{i-1}(r_{(\alpha)(\alpha+1)} \cdot B), (\Sigma_{\alpha=0}^{i-1} P_{Tr_{(\alpha)(\alpha+1)}} + \Sigma_{\alpha=0}^{i-1}(r_{(\alpha)(\alpha+1)} \cdot k_{v_p} \cdot B)))$. Thus, we can see that the trust on edges of the path from the payer to node $v_i$ are aggregated into the trust token. The process for the type aggregation is similar. Then, $v_i$ forwards the updated tokenset to its contacts.
3. The process in step 2 is repeated until the tokenset reaches the payee, which, in turn, sends the tokenset back to the payer.
4. The payer decrypts tokens in the received tokenset to gain the needed values. As an example of the trust decryption with the private key $k_{v_p}$ of the payer , $D(E_{(0)(i)}) = (\Sigma_{\alpha=0}^{i-1} P_{Tr_{(\alpha)(\alpha+1)}} + \Sigma_{\alpha=0}^{i-1}(r_{(\alpha)(\alpha+1)} \cdot k_{v_p} \cdot B)) - ((\Sigma_{\alpha=0}^{i-1}(r_{(\alpha)(\alpha+1)} \cdot B)) \cdot k_{v_p}) = \Sigma_{\alpha=0}^{i-1} P_{Tr_{(\alpha)(\alpha+1)}}$, a point on the elliptic curve. Then, the payer needs to decode this point and obtain the bit string in the binary field $F_2^m$, that is, $\Sigma_{\alpha=0}^{i-1} Tr_{(\alpha)(\alpha+1)}$.

   The type decryption is made similarly. Then, the payer retrieves the trust preference defined for the payee and validates the trust conditions. If type and trust evaluations succeed, the payer can calculate the depth from the

[14] Each value representing the relationship information (i.e., trust, depth, type) is a bit string in $F_2^m$. Since the computing operators of ECC are done on points in elliptic curves, these bit strings are needed to be mapped onto points on a binary elliptic curve [30] [31] [32]. Due to this reason, for simplicity and the additive property of the mapping, a message $W$ is mapped to a point on $(E)$ as $P = W \cdot B$, where $B$ is the base point.

type token. As in [1], the final aggregate type is indeed a product of the depth (i.e., a number of hops from the payer to the payee) and the type defined in the local trust conditions of the payer. Hence, the depth can be simply computed by dividing the final aggregate type by the type retrieved from the local trust condition. We can do calculation on depth and type because depth is a number while the type before processed is mapped onto a defined number in the binary field $F_2^m$ at the payer side, and this number must be unique. For example, the type $''friend''$ is mapped onto the number $1001110110011_b$ (i.e., $5043_d$).

SmartPay adopted the Tidal trust algorithm [26] to calculate the trust value between two nodes. Moreover, to reduce the bandwidth consumption due to the tokensets propagation, we adopted a solution inspired by $k$-anonymity [33]. At this purpose, we have to note that, to greatly reduce the tokenset propagation, each intermediate node should forward the tokenset through only the edges labeled with the relationship type required by the trust conditions, as the final goal is indeed the discovery of that type of path. However, this naive approach would imply to make each intermediate node aware of the relationship type required by the condition, and thus able to infer that the payer is connected at least with a node with this type of relationship. To avoid this problem, in [1] we assumed that the payer selects a set of $k$ relationship types, denoted as $A_{rel}$, including the type required by the trust condition. Then, each intermediate node has to forward the tokenset only to contacts with which it has a relationship of one of the types in $A_{rel}$. This solution has the advantage of reducing tokenset propagation by at the same time, avoiding the inference of the relationship in the trust condition. Therefore, SmartPay makes nodes able to aggregate trust, depth, and relationship type, while, at the same time, protecting the privacy of this information.

## 5. SmartPay over MANET: Issues and Solutions

To be enforced in MANET, SmartPay needs to be expanded so as to handle MANET communications. In order to distinguish users in SmartPay social network and in MANET, we organize the structure of Smartpay over MANET into two layers, that is, SmartPay Social Network Layer (SSNL) and Smartpay MANET Layer (SMNL). Users in the SSNL are called contacts, whereas users in the SMNL are called nodes. Hereafter, let $v_i^S$ denote a contact $i$, and $v_i^M$ denote a node $i$. One contact in the SSNL is mapped onto one node in the SMNL, and vice versa. SSNL includes contacts operating according to the SmartPay protocol discussed in Section 4. In particular, the SSNL is responsible for aggregating path information and propagating the tokenset to the other contacts. Whereas, the SMNL includes nodes (physical mobile devices) and communications in MANET. MANET communications are set up according to requests on SSNL. The SMNL is in charge of forwarding the tokenset to neighbors, or stopping the tokenset when the destination is reached. It is important to note that while contacts in SSNL are mapped on nodes in SMNL, the same does not hold for edges in the corresponding graphs. Indeed, in a MANET graph, edges are defined based on nodes' radio ranges, and not based on relationship information like in SSNL. As such, the protocol for discovering a path in a social graph defined in [1] cannot be applied as it is in the MANET graph. Indeed, the tokenset flows might follow a different path in the MANET layer, due to the availability and the position of nodes at that moment, as the following example clarifies.

**Example 3.** Let us consider the scenario of a MANET created among mobile devices of spectators of a football match at a big stadium. Let us assume some of them are connected in a social network graph as in Figure 3. As presented in Figure 4, this MANET consists of several partitions, that are connected together by means of a set of connectors (wireless access points) placed around the stadium. Let us consider four friends: Bob, Evans, Carl, and Alice, that come to the stadium to follow the match, but they sit in different stadium areas.

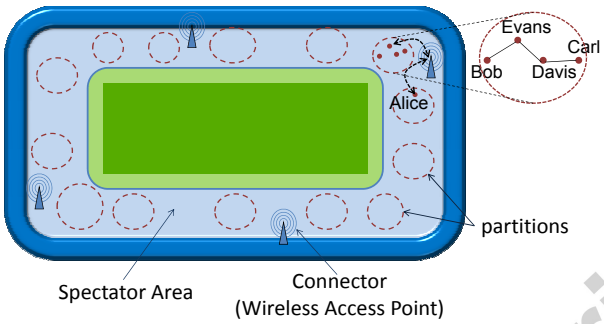



Figure 4: Communication at a stadium

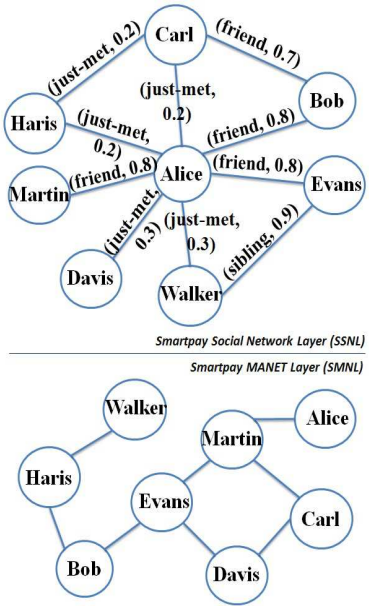



Figure 5: SmartPay protocol over MANET

Assume that Bob is sitting next to Evans, Carl is a little far from Evans and Bob, while Alice is at the farthest place from the others. Based on the distance, they are located into two different partitions (see Figure 4). Evans and Bob can communicate directly because they are in the radio range of each other. If Bob wants to communicate with Carl, he must communicate through two intermediate nodes, that is, Evans and Davis. If Alice wants to communicate with Bob, her message needs to be sent to a connector that then forwards the message to Bob's partition. Continuing with this scenario, let us consider the social graph and MANET described in Figure 5. As depicted in Figure 5, connections in SSNL and SMNL are different. Suppose that Bob wants to buy snack and beverage, and thus he asks Evans for lending him money by making a direct request. Evans and Bob are not connected in SSNL, Evans does not know Bob very well, so he initializes and sends a SmartPay tokenset to Bob through the SmartPay application installed on his cell phone. The tokenset propagates from Evans to Bob through SSNL path Evans→Alice→Bob. Whereas, in the SMNL the tokensets go directly to Bob through the path Evans→Bob.

Due to the SSNL communication layer, the SmartPay protocol as described in [1] cannot be directly applied, as further information needs to be inserted in the tokenset in order to route it in the MANET network. As it will be described in the next section, the revised tokenset has still to ensure privacy and security properties.

## 6. Expanded Smartpay Protocol (ESP)

Let us consider the three main roles with which nodes participate in the protocol, that is, the payer, the payee, and the intermediate nodes.

***The Payee.*** Let us assume that a user $v_{payee}$ wants to borrow an amount of money from another user, say $v_{payer}$. He/she first needs to send $v_{payer}$ a request. By using SmartPay, the payer will generate a tokenset which is propagated in the social graph until it reaches the payee. The tokenset has to contain some information that makes the payee able to determine he/she is the final

destination of the received tokenset or this relates to another execution of ESP. The naive idea is to insert the payee identity in the tokenset. Yet, the payee identity is a personal information. Thus, we use an additional information, called *Validator*, which is generated as the encryption with the public key of the payee, i.e., $(K_{payee} \cdot B)$, of some local information gathered from the payee's mobile device. These are, as an example, the MAC address and the timestamp of the instant when the payee makes a request to the payer. The timestamp is used as a session identity to distinguish the payment request at the payee side, so that the payee can recognize exactly the request relating to the payer. Before encrypting the combination of MAC address and timestamp, to improve the robustness of Validator against adversaries (see Section 9), we take the combination of MAC address and timestamp apart into groups of bytes, then permute randomly positions of these groups of bytes by applying the Fisher-Yates shuffle algorithm [34]. Note that the random arguments used for permutation are held by the payee, so that the payee can reuse them for its identity data recovery. After that, the payee encrypts the permuted combination of local information by its public key to gain Validator. Then, Validator is appended at the end of the ESP tokenset. Also notice that Validator is not modified through the path from the payer to the payee.

**Definition 1.** *(Validator).* Let $v^M_{payee}$ be the node who makes the payment request, $MA_{v^M_{payee}}$ be the MAC address of $v^M_{payee}$, $t_{v^M_{payee}}$ be the timestamp when $v^M_{payee}$ makes the request, $(K_{v^M_{payee}} \cdot B)$ be the public key of $v^M_{payee}$. The Validator made by $v^M_{payee}$ is defined as follows:

$$Validator_{v^M_{payee}} = Enc_{(K_{v^M_{payee}} \cdot B)}(shuffle([MA_{v^M_{payee}} || t_{v^M_{payee}}]))$$

where shuffle() is the Fisher-Yates shuffle algorithm, '||' is an operation concatenating two bit strings.

$v_{payee}$ inserts the Validator into the request and sends the request to $v_{payer}$.

**Example 4.** Let us continue with Example 3 (see Figure 5), and suppose once again that Bob wants to borrow money from Evans. Therefore, he needs to gen-

erate the request along with the Validator and sends them to Evans. Bob first retrieves the request timestamp, i.e., $t_{Bob^M}$, and the MAC address, i.e., $MA_{Bob^M}$, from his mobile device. Then, he combines them together, creates a random permutation on the obtained combination, and then encrypts the resulted permutation with his public key to generate the Validator: $Validator_{Bob^M} = Enc_{(K_{Bob^M} \cdot B)}(shuffle[MA_{Bob^M}||t_{Bob^M}])$. Bob sends then the Validator and the payment request to Evans.

After that, $v_{payee}$ waits for an ESP tokenset from $v_{payer}$ through his/her contacts. When $v_{payee}$ receives an ESP tokenset including his/her Validator, he/she sends that ESP tokenset back to the payer and waits for the payment result.

**Example 5.** Consider Example 4, when Bob receives a tokenset with the Validator $Enc_{(K_{Bob^M} \cdot B)}(shuffle([MA_{Bob^M}||t_{Bob^M}]))$, he can use his private key (i.e., $K_{Bob^M}$) to successfully decrypt the Validator, and verifies that he is the destination of the tokenset. Then, Bob stops forwarding the tokenset to his neighbors.

***The Payer.*** In defining ESP we wish to cope with an open issue in [1]. Indeed, in order to aggregate information into the tokenset, in [1] we assumed that each user knows the public keys of other participants. However, this makes users involved in the SmartPay execution able to infer who is the payer. In ESP, we wish to protect this information as well. Therefore, we assume that the payer, i.e., $v^S_{payer}$, generates a pair of temporary ECC public and private keys, i.e., $(K_{tempv^S_{payer}} \cdot B)$ and $K_{tempv^S_{payer}}$, for each new transaction, to be used instead of its original public key so that intermediate nodes cannot infer the payer identity. Temporary keys have to be protected so as to avoid other contacts to misuse them. Beside, in order to avoid that intermediate nodes do aggregation on the same tokenset received from the same SSNL node in case the sending node can be subject of a replay attack (see Section 9), we insert the information of session identity (ID) between two SSNL nodes, called $sessionID$.

$sessionID$ can include an ID of the receiving node and a number standing for the session order. Beside, to let the receiving nodes know the sending node, we insert the ID of the sending node, namely $ID(v_i)$, into the tokenset. Both $ID(v_i)$ and $sessionID$ should be protected as the temporary public key of the payer. At this purpose, we require that the temporary public key, the $sessionID$ and $ID(v_i)$ are encrypted by the intermediate contacts who have used it to aggregate the relationship information with the official public key of the SSNL contact to which the tokenset has to be sent. The above extra information is encapsulated into a data structure, called *secure key*, and inserted into the head of the ESP tokenset.

**Definition 2.** *(Secure Key).* Let $v_i^S$ be the node processing the ESP tokenset, $v_{i+1}^S$ be a contact of $v_i^S$, $(K_{v_i^S} \cdot B)$ be the public key of $v_i^S$, $(K_{v_{i+1}^S} \cdot B)$ be the public key of the neighbor $v_{i+1}^S$, $sessionID_{(v_i^S, v_{i+1}^S)}$ be the session identity of $v_i^S$ and $v_{i+1}^S$. *SecureKey* between $v_i^S$ and $v_{i+1}^S$ is defined as follows:

$$SecureKey_{(v_i^S, v_{i+1}^S)} = Enc_{(K_{v_{i+1}^S} \cdot B)}(ID(v_i)||(K_{tempv_{payer}^S} \cdot B) ||sessionID_{(v_i^S, v_{i+1}^S)})$$

**Example 6.** Let us continue with Example 4. After receiving the request and Validator from Bob, Evans generates a pair of temporary public and private keys, $(K_{tempEvans} \cdot B, K_{tempEvans})$, to be used for the current payment transaction only. Evans keeps the temporary private key, and propagates the temporary public key to his contacts, that is, Alice and Walker. In particular, Evans encrypts his temporary public key, the session IDs between him and them (i.e., $sessionID_{(Evans,Alice)}$, $sessionID_{(Evans,Walker)}$), and his ID (i.e., $ID(Evans)$) with their public keys, i.e., $(K_{Alice} \cdot B)$ and $(K_{Walker} \cdot B)$ respectively, and obtains two distinguished encryptions. From the above information, Evans creates two secure keys, as follows:

- $SecureKey_{(Evans,Alice)} = Enc_{(K_{Alice} \cdot B)}(ID(Evans)||K_{tempEvans} \cdot B) ||sessionID_{(Evans,Alice)})$

- $SecureKey_{(Evans,Walker)} = Enc_{(K_{Walker} \cdot B)}(ID(Evans)||K_{tempEvans} \cdot B) ||sessionID_{(Evans,Walker)})$

The secure key (Definition 2), the SmartPay tokenset as defined in [1], and the Validator (Definition 1) are then enveloped into a unique data structure, namely, the *ESP tokenset*. Moreover, to guarantee the integrity of the content of elements of ESP tokenset, i.e., no intermediate nodes can modify the public key encryption, a signature of ESP tokenset encryption is made with the private key of the sending contact using the Elliptic Curve Digital Signature Algorithm (ECDSA) algorithm [11]. An ESP tokenset is formally defined as follows:

**Definition 3.** *(ESP Tokenset).* Let $v_i^S$ be a node, $v_{i+1}^S$ be one of $v_i^S$'s contacts, $v_{payer}^S$ be the payer, $v_{payee}^M$ be the payee. Validator is created by $v_{payee}^M$ as in Definition 1. SecureKey is created by $v_i^S$ as in Definition 2, SPTokenset is the tokenset as defined in [1]. Hence, the ESP tokenset is defined as follows:

$$TK_{(v_{payer}^S, v_{i+1}^S)} = [combi||sign_{K_{v_i^S}}(combi)]$$

where

$$combi = [SecureKey_{(v_i^S, v_{i+1}^S)}||(SPTokenSet_{(v_{payer}^S, v_{i+1}^S)} ||Validator_{v_{payee}^M}]$$

and $sign()$ is the function that $v_i^S$ uses for generating the signature of the encryption of ESP tokenset with its private key.

**Example 7.** Let us continue with Example 6. After creating the secure key, Evans detaches Validator from the received request from Bob, and creates two tokensets for his two contacts on SSNL, i.e., Alice and Walker, as follows:

- $TK_{(Evans,Alice)} = combi||sign_{K_{Evans}}(combi)$, where $combi = [SecureKey_{(Evans,Alice)} ||SPTokenset_{(Evans,Alice)}||Validator_{Bob}]$.
- $TK_{(Evans,Walker)} = combi||sign_{K_{Evans}}(combi)$ where $combi = [SecureKey_{(Evans,Walker)} ||SPTokenset_{(Evans,Walker)}||Validator_{Bob}]$.

where $SecureKey_{(Evans,Alice)}$, $SecureKey_{(Evans,Walker)}$ are like in Example 6, whereas Validator is like in Example 4.

$v_{payer}$ sends the initial ESP tokenset to its contacts and waits for the final ESP tokenset from $v_{payee}$. Once received, $v_{payer}$ decrypts the SmartPay tokenset, and enforces the trust preferences according to the Smartpay protocol (as described in Section 4).

***Intermediate nodes.*** Before presenting in details the steps executed by an intermediate node in the ESP protocol, let us remind that, when nodes move around, MANET can be grouped into many partitions. In order for them to connect to each other, connectors forward ESP tokensets from a partition to nodes of another partitions in their radio ranges. In a MANET partition, nodes that do not have any neighbor to propagate the tokenset are called *border nodes*. These nodes send the received ESP tokenset to connectors located in their radio range. There might be more than one border node in a partition.

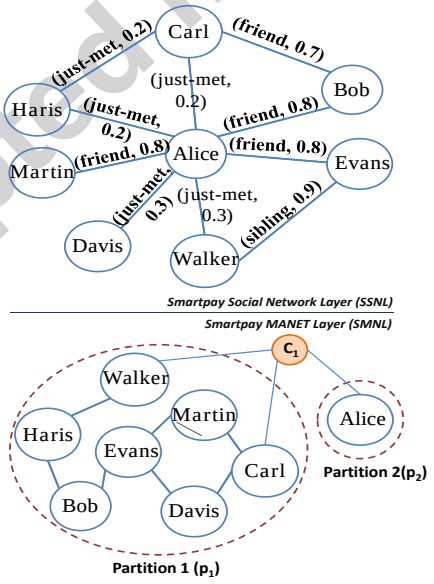



Figure 6: Social Network Layer and Manet Layer Cooperation

**Example 8.** Let us continue with Example 3 by assuming that the MANET is divided into two partitions, as in Figure 6. Suppose that Walker stays in partition $p_1$ and receives from Haris an ESP tokenset whose destination is Bob. However, Walker does not have any neighbor in $p_1$, so he forwards the ESP tokenset to connector $C_1$. In this example, Walker is a border node. When $C_1$ receives the ESP tokenset from Walker, it forwards the received ESP tokenset to nodes located in the radio range of $C_1$, that is, nodes in partition $p_2$. In this example, it is assumed that Alice is a node in $p_2$ staying in the radio range of $C_1$, so Alice is a border node as well. She can receive the ESP tokenset from $p_1$ through $C_1$. Alice continues to propagate the received ESP tokenset to her contacts. The ESP tokenset is then propagated until it arrives to Bob.

Now let us describe the operation done by intermediate nodes, encoded by Algorithm 1. Let us assume that the intermediate node $v_i$ receives an ESP tokenset, i.e., $TK_{(v_{payer},v_i)}$, from node $v_{i-1}$. Let $TK_{(v_i,v_{i+1})}$ be the ESP token aggregated with the information of the relationship between $v_i$ and $v_{i+1}$ by $v_i$. The result of Algorithm 1 is stored into a variable, namely $i\_resProc$. This variable can assume one of the values in the set $\{\_SENT\_PAYER, \_NO\_CONTACT, \_SUCCESS\_FORWARD, \_SUCCESS\_AGGR\_PROP, \_DUP\_TKSET, \_CHANGED\_CONTENT\}$, where *_SENT_PAYER* means that the ESP tokenset is sent back to the payer, *_NO_CONTACT* means that there does not exist any contact for $v_i$ to forward the ESP tokenset to, *_SUCCESS_FORWARD* means that $v_i$ forwards the ESP tokenset to its neighbors, *_SUCCESS_AGGR_PROP* means that $v_i$ aggregates the information on the relationship between itself and its contacts into the received ESP tokenset and forwards the aggregate tokenset to its contacts, *_DUP_TKSET* means that the tokenset was processed before, whereas, *_CHANGED_CONTENT* means that the tokenset content has been changed on the communication channel. The values *_SUCCESS_AUTH* and *_FAILED_AUTH* are used in the algorithm to show the result of tokenset authentication. *_SUCC_AUTH* shows that the tokenset is authenticated successfully, whereas *_FAILED_AUTH* means that the authentication fails. We also

use *_SUCCESS_DECRYPT* to describe the status where SecureKey decryption is successful.

We assume that each user on SSNL has the public keys of his/her contacts. $v_i$ first decrypts the SecureKey received tokenset with its private key (line 2) so as to determine if it is the intermediate node needing to aggregate the relationship information between itself and $v_{i-1}$. In case the decryption fails, $v_i$ is not the contact of the sender, $v_i$ needs to transfer the tokenset to its neighbors or connectors on SMNL (line 33). If the decryption works out (line 3), $v_i$ is the node needing to do aggregation. However, it also needs to verify the integrity of the received tokenset and that the tokenset is truly from its SSNL contact, by using the public key of its SSNL contact (line 5). However, $v_i$ does not know which contact sent the tokenset to it. Therefore to get the public key of its sending contact, $v_i$ retrieves the ID of the sending contact in the decryption. Assuming that, the found ID is of $v_{i-1}$, i.e., $ID(v_{i-1})$, it then looks for the public key of $v_{i-1}$ (line 4). Then $v_i$ authenticates the tokenset. If the authentication works out, it means that the integrity of the received tokenset is guaranteed and the tokenset content has been not changed. Hence, $v_i$ needs to check if the tokenset is duplicated based on the $SessionID$ in SecureKey (line 7). If the authentication or the duplication checking fail, the tokenset is dropped (lines 23, 28). If the duplication checking is successful, the next step is to evaluate the Validator, if $v_i$ is the destination of the tokenset (line 8). In particular, $v_i$ uses its private key to decrypt the Validator and then recovers the shuffled value in the decrypted Validator and sees if it makes sense or not. If the result contains the local information of $v_i$, it proves that $v_i$ is the destination of the tokenset, it then sends the tokenset back to the payer (line 9). Otherwise, $v_i$ obtains the contact list (line 12) by retrieving from its local storage, if it has contacts it does aggregation between itself and its contacts by calling function *aggregateTokenset()* (line 13) (see Function 2), then calls function *sendOnSMNL()* (line 14) (see Function 3) to propagate the tokenset. If it does not have any contact, the social path through it is dead-end, so it drops the tokenset (line 17).

**Algorithm 1** *processTokenset()*

**Input:** $TK_{(v_{payer},v_i)}$
**Output:** $i\_resProc$

Let $seckey$ be decryption of $TK_{(v_{payer},v_i)}.SecureKey$.
$resDec = decryptECC(TK_{(v_{payer},v_i)}.SecureKey, K_{v_i}, seckey)$;
**if** $(resDec == \_SUCCESS\_DECRYPT)$ **then**
  $K_{v_{i-1}} \cdot B = findContactPbKey(secKey.ID(v_{i-1}))$;
  $resAuthen = authenKey(TK_{(v_{payer},v_i)}, (K_{v_{i-1}} \cdot B))$;
  **if** $(resAuthen == \_SUCCESS\_AUTH)$ **then**
    **if** $isDuplicate(seckey.SessionID_{(v_{i-1},v_i)})$ **then**
      **if** $(isDestination(TK_{(v_{payer},v_i)}.Validator, K_{v_i}))$ **then**
        $send(TK_{(v_{payer},v_i)}, v_{payer})$;
        $i\_resProc = \_SENT\_PAYER$;
      **else**
        **if** $(hasContacts())$ **then**
          $TK_{(v_{payer},v_{i+1})} = aggregateTokenset(TK_{(v_{payer},v_i)})$;
          $i\_resProc = sendOnSMNL(TK_{(v_{payer},v_{i+1})})$;
          $return\ i\_resProc$;
        **else**
          $Drop(TK_{(v_{payer},v_i)})$;
          $i\_resProc = \_NO\_CONTACT$;
          $return\ i\_resProc$;
        **end if**
      **end if**
    **else**
      $Drop(TK_{(v_{payer},v_i)})$;
      $i\_resProc = \_DUP\_TKSET$;
      $return\ i\_resProc$;
    **end if**
  **else**
    $Drop(TK_{(v_{payer},v_i)})$;
    $i\_resProc = \_FAILED\_AUTH$;
    $return\ i\_resProc$;
  **end if**
**end if**
$i\_resProc = sendOnSMNL(TK_{(v_{payer},v_i)})$;
$return\ i\_resProc$;

**Function 2** *aggregateTokenset()*

**Input:** $TK_{(v_{payer},v_i)}$
**Output:** $TK_{(v_{payer},v_{i+1})}$
1: $TK_{(v_i,v_{i+1})}.SecureKey = encryptECC(concatenate(ID(v_{i+1}), K_{tempv_{payer}} \cdot B, generateSessionID(v_i, v_{i+1})), K_{v_{i+1}} \cdot B)$;
2: $TK_{(v_i,v_{i+1})}.SPTokenset = SmartPay.createTokenset(TK_{v_{payer},v_i}.SPTokenset, K_{tempv_{payer}} \cdot B)$;
3: $TK_{(v_i,v_{i+1})}.Validator = TK_{(v_{payer},v_i)}.Validator$;
4: $TK_{(v_{payer},v_{i+1})} = concatenate(TK_{(v_i,v_{i+1})}, signECDSA(TK_{(v_i,v_{i+1})}, K_{v_i}))$;
5: $return\ TK_{(v_{payer},v_{i+1})}$;

Additionally, function *aggregateTokenset()* receives an input, that is, a tokenset from $v_{i-1}$. It then returns an output, that is, a tokenset $TK_{(v_{payer},v_{i+1})}$ aggregated the relationship information on the edge connecting it and its contact, i.e., $v_{i+1}$. $v_i$ generates the session ID between it and $v_{i+1}$ then concatenates the ID of $v_{i+1}$, the temporary public key of the payer, and the session ID. After that, $v_i$ encrypts this concatenation by the public key of $v_{i+1}$ (line 1). Then, $v_i$ aggregates the relationship information between it and $v_{i+1}$, using the temporary public key of the payer, by calling function *SmartPay.createTokenset()* as in [1] (line 2). The Validator is reused from the received tokenset without being modified (line 3). After that, $v_i$ creates a signature of the aggregate tokenset and concatenate the new tokenset and its signature (line 4).

**Function 3** *sendOnSMNL()*

**Input:** $TK_{(v_{payer},v_j)}$
**Output:** $resSend$
1: $resSend = false$;
2: **if** $(hasNeighbor())$ **then**
3: $\quad resSend = send(TK_{(v_{payer},v_j)}, list\_manet\_neighbor)$;
4: **else**
5: $\quad resSend = send(TK_{(v_{payer},v_j)}, list\_manet\_connector)$;
6: **end if**
7: $return\ resSend$;

Function *sendOnSMNL()* receives a tokenset $TK_{(v_{payer},v_j)}$. $v_i$ verifies if it

has any neighbors (line 2) by the neighborhood discovery protocol.[15] In case the verification works out, $v_i$ obtains a list of neighbors (i.e., $list\_manet\_neighbor$), then transfers the tokenset to neighbors in $list\_manet\_neighbor$ (line 3). Otherwise, $hasNeighbor()$ returns a list of connectors (i.e., $list\_manet\_connector$) to $v_i$, then $v_i$ sends the tokenset to its connectors (line 5).

The following example clarifies how Algorithm 1 works.

**Example 9.** Let us continue with Example 3 and Figure 6. For the sake of simplicity, we do not consider any trust preference, so the relationship information is removed from the figures depicting the steps in the example. Consider the path on SSNL connecting Bob to Evans, that is, Evans→Alice→Bob, where Evans is the payer, and Bob is the payee. After Bob makes a request to Evans for an amount of money, Evans sends the initial ESP tokenset back to Bob. There is only one SMNL path from Evans to Bob, that is, Evans→Bob. According to Algorithm 1, the following steps are performed:

1. Bob sends the payment request and Validator to Evans (see Figure 7).

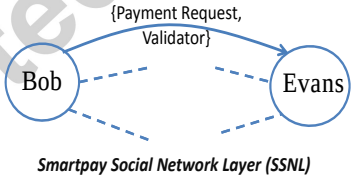



Figure 7: Step 1 of Example 9

2. Evans receives the request from Bob (see Figure 8), then it is:

[15] Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP) (RFC 6130) https://tools.ietf.org/html/rfc6130.

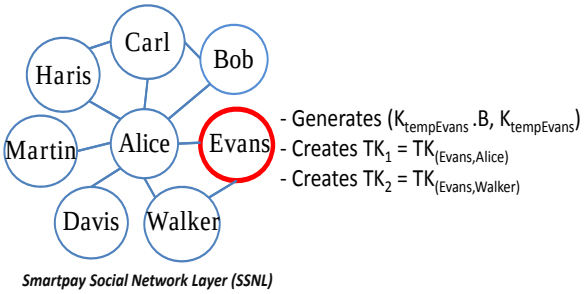



Figure 8: Step 2 of Example 9

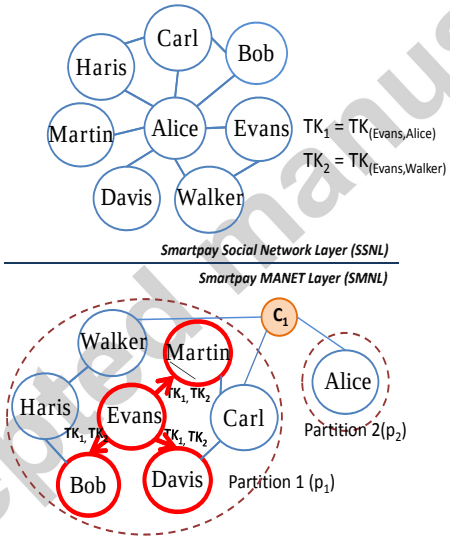



Figure 9: Step 3 of Example 9



- Evans generates a pair of temporary keys $(K_{tempEvans} \cdot B, K_{tempEvans})$ only used for this payment transaction.
- Evans initializes the ESP tokensets for his two directly connected contacts on SSNL, i.e., Alice and Walker, to obtain $TK_1 = TK_{(Evans,Alice)}$ and $TK_2 = TK_{(Evans,Walker)}$.

3. Evans then looks for the list of his neighbors to forward these two ESP

tokensets. Evans has three neighbors, that is, Martin, Bob, and Davis. Hence, he sends these two ESP tokensets to each of them.

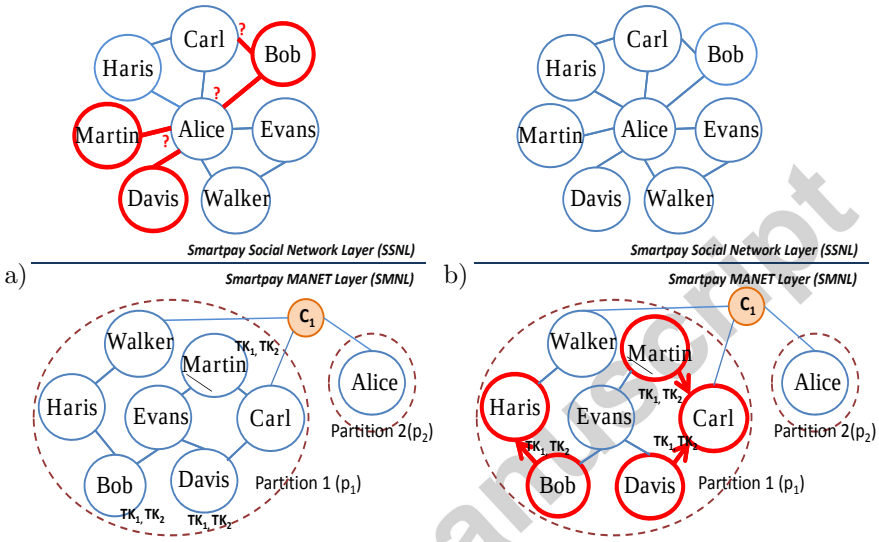



Figure 10: Step 4 of Example 9

4. Martin, Bob, and Davis receive two ESP tokensets. They start to verify if they are contacts of Evans in the social graph (line 5 of Algorithm 1). It results that they are not Evans' contacts (see step 4a in Figure 10-a). Bob, Martin, and Davis thus look for their neighbors (line 12), and forward the received ESP tokensets to them without changing the content (line 33). Bob sends two ESP tokensets to Haris while Martin and Davis send two ESP tokensets to Carl (see step 4b in Figure 10-b).

5. After receiving the ESP tokensets, Carl and Haris verify if they are contacts of Evans by decrypting the SecureKey (line 2 of Algorithm 1) (see step 5a in Figure 11-a). The function fails, therefore, Carl and Haris forward the ESP tokensets to their neighbors (see step 5b in Figure 11-b). Haris' neighbor is Walker, thus, Haris transfers the ESP tokensets to Walker. Carl does not have neighbors, so he sends the ESP tokensets to connectors in his radio range. Here we assume that Carl's connector is

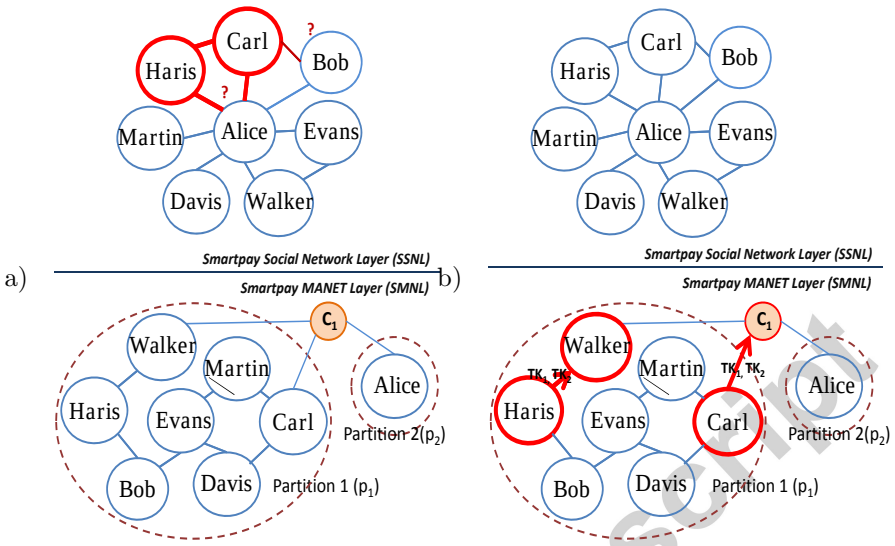



Figure 11: Step 5 of Example 9

$C_1$.

6. Walker receives two ESP tokensets from Haris. He checks if he is Evans' contact (line 2 of Algorithm 1) and the decryption succeeds (see step 6a in Figure 12-a). Then he authenticates if the tokenset content has been changed (line 5 of Algorithm 1). Assume that the tokenset has been changed on the communication channel between Walker and Haris. Hence, Walker checks if this tokenset from Evans was processed before (line 7 of Algorithm 1). Assume that the tokenset has been already processed. So, Walker continues to verify if he is the payee of Evans (line 8) and the check fails. Therefore, Walker continues to search his contacts (line 12) and it retrieves Alice. Walker aggregates the relationship information between him and Alice to gain $TK_3 = TK_{(Evans,Alice)}$ (line 13). Then, Walker needs to propagate the tokenset (line 14), he looks for his neighbors, he finds out that he does not have neighbors. So, he forwards to $C_1$ the ESP tokensets, including the updated ESP tokenset $TK_3 = TK_{(Evans,Alice)}$ and the remaining tokenset $TK_1$ from Haris (see step 6b in Figure 12-b).

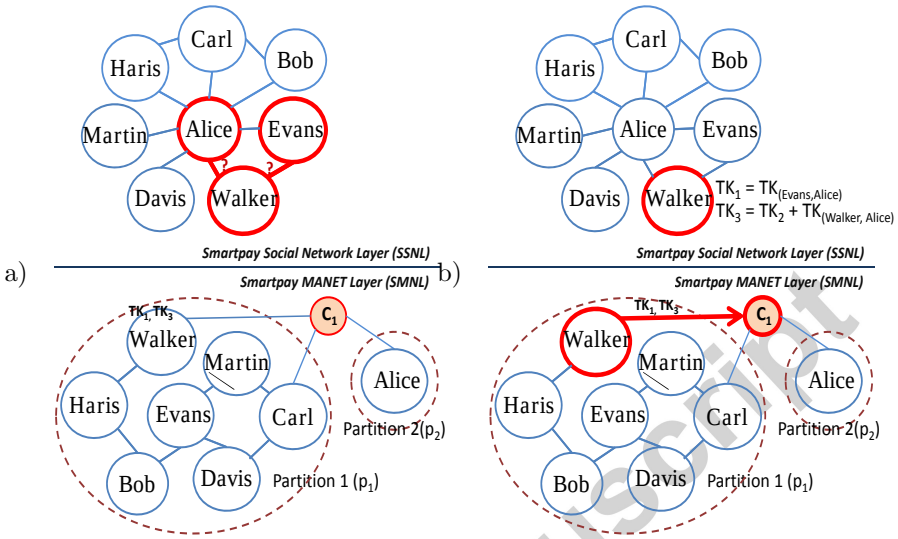



Figure 12: Step 6 of Example 9

7. Connector $C_1$ sends the received ESP tokensets from Carl and Walker to partition $p_2$. In $p_2$, there is only one node, that is, Alice (see step 7 in Figure 13).

8. Two ESP tokensets from Walker and one from Carl are for Alice. Another one is not for Alice, that is, $TK_2 = TK_{(Evans,Walker)}$ from Carl. So, Alice should forward $TK_2$ to her neighbors without modifying its content, but Alice does not have any neighbor. So, she forwards $TK_2$ to $C_1$.

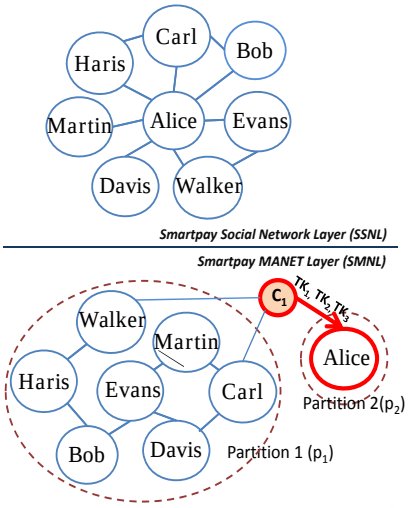



Figure 13: Step 7 of Example 9

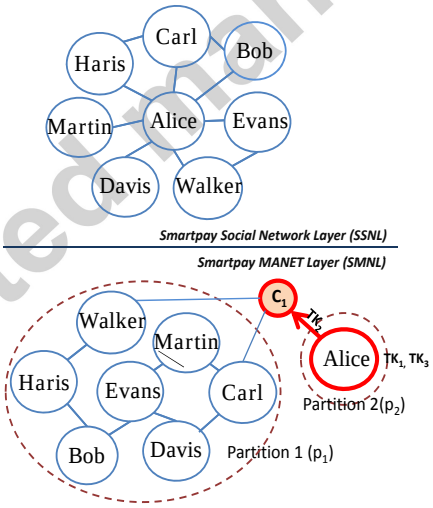



Figure 14: Step 8 of Example 9

9. With the remaining tokensets, Alice searches the list of her contacts (line 12) and finds out six contacts that have to update their contents, that is, Bob, Carl, Davis, Haris, Martin, and Evans. She updates the received ESP tokensets (line 13), and obtains $TK_4 = TK_{(Evans,Bob)} = TK_1 +$

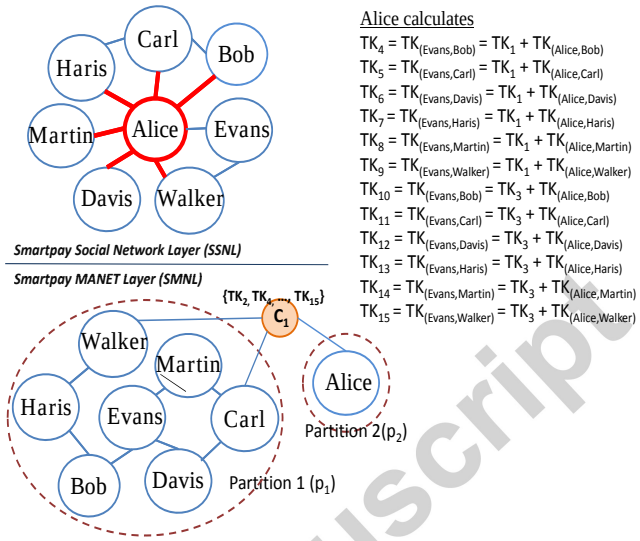



Figure 15: Step 9 of Example 9

$TK_{(Alice,Bob)}$, $TK_5 = TK_{(Evans,Carl)} = TK_1 + TK_{(Alice,Carl)}$, $TK_6 = TK_{(Evans,Davis)} = TK_1 + TK_{(Alice,Davis)}$, $TK_7 = TK_{(Evans,Haris)} = TK_1 + TK_{(Alice,Haris)}$, $TK_8 = TK_{(Evans,Martin)} = TK_1 + TK_{(Alice,Martin)}$, $TK_9 = TK_{(Evans,Walker)} = TK_1 + TK_{(Alice,Walker)}$, $TK_{10} = TK_{(Evans,Bob)} = TK_3 + TK_{(Alice,Bob)}$, $TK_{11} = TK_{(Evans,Carl)} = TK_3 + TK_{(Alice,Carl)}$, $TK_{12} = TK_{(Evans,Davis)} = TK_3 + TK_{(Alice,Davis)}$, $TK_{13} = TK_{(Evans,Haris)} = TK_3 + TK_{(Alice,Haris)}$, $TK_{14} = TK_{(Evans,Martin)} = TK_3 + TK_{(Alice,Martin)}$, $TK_{15} = TK_{(Evans,Walker)} = TK_3 + TK_{(Alice,Walker)}$. Alice wants to send them to her neighbors (line 14), so searches for them, but she does not have any neighbor. Therefore, she forwards all the tokensets to connector $C_1$ which is in her radio range (see step 9 in Figure 15).

10. $C_1$ receives the ESP tokensets from Alice, and sends all of them to Walker and Carl who are border nodes of partition $p_1$. Let us first consider Walker. Carl will repeat similar steps to process the received ESP tokensets (see step 10 in Figure 16).

11. Walker receives the ESP tokensets from $C_1$. He decrypts the SecureKey

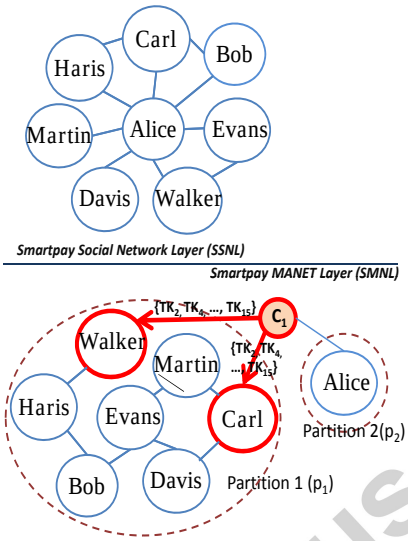



Figure 16: Step 10 of Example 9

of all received tokensets to check if there is any tokenset for him to update (lines 2, 5 of Algorithm 1). But no tokenset is for him. Then, he forwards all of them to Haris who is his neighbor (line 33) (see step 11a in Figure 17-a). Among the received ESP tokensets from Walker, there are tokensets for Haris to update, that is, $TK_7$ and $TK_{13}$. Haris forwards the other tokensets to Bob as Bob is his neighbor (see step 11b in Figure 17-b).

12. Bob decrypts SecureKey of the received tokensets (line 2 of Algorithm 1), there is one tokenset for him. He authenticates if the tokenset has been changed on the communication channel (line 5 of Algorithm 1), then he also checks if the tokenset has been processed (line 7 of Algorithm 1). Then, he validates if he is the destination of the received ESP tokensets (line 8). He is the destination of two ESP tokensets $TK_4$ and $TK_{10}$. (see step 12 in Figure 18).

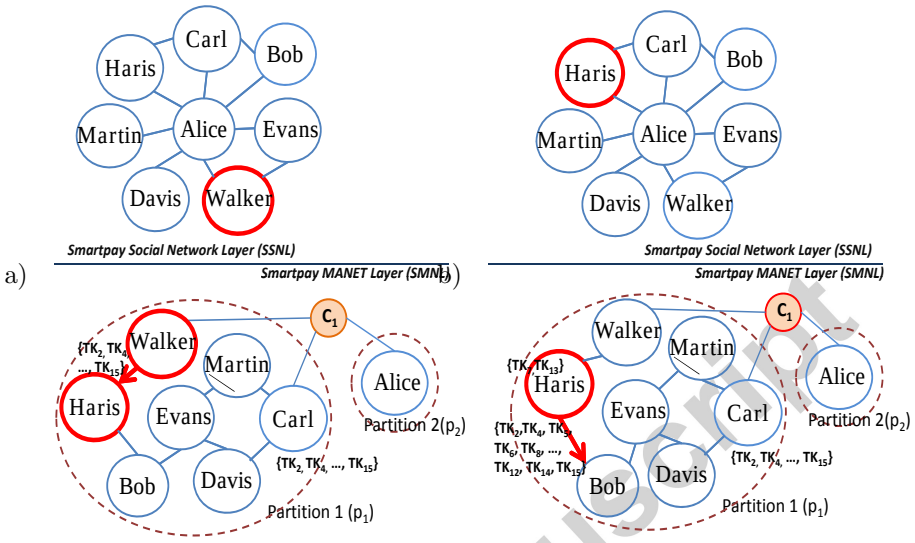



Figure 17: Step 11 of Example 9

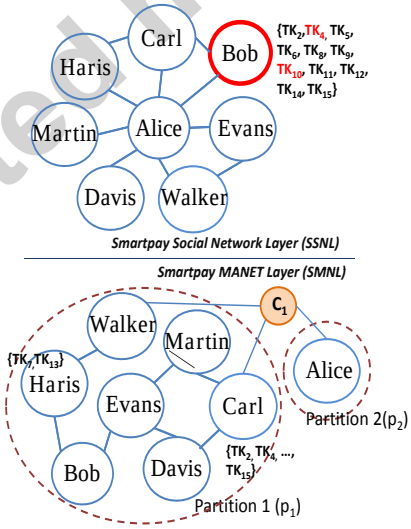



Figure 18: Step 12 of Example 9

13. Bob sends ESP tokensets to Evans (line 9) (see step 13 in Figure 19).

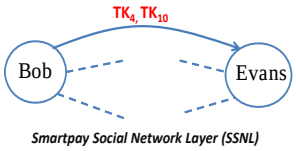



Figure 19: Step 13 of Example 9

## 7. Condition-driven flooding

As presented in Section 4, SmartPay exploits $k$-anonymity [33] to reduce the number of distributed tokensets. Relationship information privacy as well as the number of propagated tokensets depend on the value of $k$. Selecting a reasonable value of $k$ to make the network consumption effective is not easy. In particular, in case $k$ is large, network performance is similar to the one of broadcasting techniques, but privacy is preserved more strongly. In case $k$ is small, the bandwidth consumption is low but privacy cannot be preserved effectively. Therefore, in order to preserve privacy, and, at the same time, improve network performance, here we propose an alternative approach exploiting secure comparison techniques.

Let us first recall that the goal of finding a social path connecting the payee and the payer is to verify if the relationship between the payee and the payer satisfies the specified trust preferences. Let us consider a trust condition $tc = (rt, d_{max}, t_{min})$ in a trust preference of the payer. Finding a valid social path connecting the payer and the payee requires to evaluate whether the relationship type of the path is equal to $rt$, the aggregated trust value is greater than or equal to $t_{min}$, and the depth of the path is less than or equal to $d_{max}$. As such, if an intermediate node is able to detect that one of these conditions is not satisfied, it is able to terminate the ESP tokenset propagation. Thus, the number of propagated ESP tokensets is significantly reduced. For this purpose, there is the need of a way that enables intermediate nodes to privately verify trust conditions. In [1], since the trust condition verification is enforced at the

payer side, only the payer can read trust conditions. In this paper, we assume that each intermediate node receives trust conditions encrypted with the payer's temporary public key, so they cannot learn plain texts of trust conditions. With respect to the protocol explained in Section 6, we add an additional step. The intermediate node, say $v_i$, has to evaluate the updated ESP tokensets based on the encrypted trust conditions by exploiting the proposed secure comparison approach. Based on the results, $v_i$, propagates only the updated ESP tokensets that verify the trust conditions.

**Example 10.** Let us continue with Example 3. Assume that an intermediate node $v_i$ receives an ESP tokenset along with the encryption of trust condition $tc = (Friend, 3, 0.6)$. After receiving the request from Bob, Evans initializes two ESP tokensets for his contacts, Alice and Walker, where the trust values are 0.8 and 0.9, respectively. These trust values and the depths (i.e., 1, as they are direct friends) satisfy $tc$. However, the type of the relationship in the initial ESP tokenset is *Friend* between Alice and Evans, but is *Sibling* between Evans and Walker. Hence, Evans only sends the ESP tokenset to Alice. The number of forwarded ESP tokensets is reduced by half. In turn, Alice has six contacts, that is, Bob, Carl, Haris, Martin, Davis, and Walker. She creates six duplicates of the received ESP tokenset, then updates them with the relationship information of the edges between herself and these contacts. After doing the secure comparison between the encrypted trust conditions and each of the updated ESP tokensets, it results that, among these six contacts, Alice chooses Bob and Martin to propagate the updated ESP tokensets since the relationship information in the ESP tokensets for both Bob and Martin are $(Friend, 2, 0.64)$. Here, the number of forwarded ESP tokensets is also reduced. In the four other updated ESP tokensets, the type of the relationship is not *Friend*. Therefore, Alice sends the updated ESP tokensets only to Bob and Martin.

In order to do a comparison between the encrypted trust conditions and the encrypted values in the ESP tokenset, we adopt the secure comparison scheme proposed by F. Kerschbaum et al. in [35]. The authors proposed a general

framework for a secure comparison between $x_0$ and $x_1$. All participants involved in this scheme do not know the real values of $x_0$ and $x_1$. They just receive the encrypted values, denoted as $E(x_0)$ and $E(x_1)$. This framework exploits homomorphic encryption, and the authors used $RSA$ (1024-bit key length) for the experiment. However, $RSA$ consumes a lot of memory, CPU and time on encrypting and decrypting, so it is not suitable for the mobile environment. To cope with this issue, we adopt the original scheme in [35], and apply the homomorphic binary ECC algorithm into it.

Secure comparison is done on trust, depth, and type of the relationship. Among these three components, trust and depth are numbers, whereas, the type needs to be converted into a number in the range $[1, 35]$. This range is chosen according to the quantity of relationship types defined in the widely used FOAF vocabulary [27]. The basic idea is that, for each intermediate node, instead of forwarding a received ESP tokenset to several contacts, this node applies the secure comparison scheme in [35] to compare the encrypted trust, depth, type of the relationship in the received tokenset with the respective encrypted threshold in the trust condition. Based on the result, the node can decide to which contacts the ESP tokenset should be forwarded.

Let us denote trust, depth, type in a trust condition as $t_{min}$, $d_{max}$, $rt$, respectively. These are encrypted with the payer's temporary public key (i.e., $(k_{v_{payer}} \cdot B)$), and denoted respectively as $E_{v_{payer}}(t_{min})$, $E_{v_{payer}}(d_{max})$, and $E_{v_{payer}}(rt)$. The encryptions are sent from the payer to its contacts, say $v_i$. Let $T_{(v_{payer},v_i)}$ be the ESP tokenset containing the encryption of trust, depth, and type which are collected in the social path from the payer to $v_i$. Let $T_{(v_{payer},v_i)}.trust$, $T_{(v_{payer},v_i)}.depth$, and $T_{(v_{payer},v_i)}.rt$, $E_{v_{payer}}(T_{(v_{payer},v_i)}.trust)$, $E_{v_{payer}}(T_{(v_{payer},v_i)}.depth)$, $E_{v_{payer}}(T_{(v_{payer},v_i)}.rt)$ be respectively trust, depth and type tokenset in $T_{(v_{payer},v_i)}$ and their corresponding encryptions with the payer's public key. Let us first consider secure comparison on trust. Secure comparison on depth and relationship type are managed in a similar way. Let us assume that $v_i$ sends $E_{v_{payer}}(T_{(v_{payer},v_i)}.trust)$ to $v_{i+1}$. Node $v_{i+1}$ aggregates $T_{(v_i,v_{i+1})}.trust$ (i.e., the trust value of the edge between $v_i$ and $v_{i+1}$ into

$E_{v_{payer}}(T_{(v_{payer},v_i)}.trust)$ and obtains $E_{v_{payer}}(T_{(v_{payer},v_{i+1})}.trust)$. Then, it applies the scheme in [35] to decide if this updated ESP tokenset can be propagated to its contacts by comparing two encryptions of the trust threshold and the aggregate trust in the tokenset. To do this comparison, $v_{i+1}$ randomizes two large numbers $r$, $r'$ in $\mathbb{N}$, and calculates the value of $E_{v_{payer}}(c)$ with the following equation, where $E(c)$ is an encryption for the computation at the next round as described in [35].

$$\begin{aligned} E_{v_{payer}}(c) &= r \cdot (E_{v_{payer}}(t_{min}) - E_{v_{payer}}(T_{(v_{payer},v_j)}.trust)) \\ &\quad - E_{v_{payer}}(r') \\ &= (r \cdot r_0 \cdot B - r \cdot r_1 \cdot B - r'' \cdot B, (r \cdot (t_{min} \\ &\quad - T_{(v_{payer},v_j)}.trust) - r') + (r \cdot r_0 - r \cdot r_1 - r'') \\ &\quad \cdot k_{v_{payer}} \cdot B) \\ &= E_{v_{payer}}(r \cdot (t_{min} - T_{(v_{payer},v_j)}.trust) - r') \end{aligned} \tag{1}$$

where $r_0$, $r_1$, $r''$ are the randoms generated while the encryptions are done.

The basic idea of Formula 1 is to compute $d = (t_{min} - T_{(v_{payer},v_{i+1})}.trust)$. $d$ is used for comparing $t_{min}$ and $T_{(v_{payer},v_{i+1})}.trust$. If $d < 0$, it implies that $t_{min} < T_{(v_{payer},v_{i+1})}.trust$; otherwise, $t_{min} \geq T_{(v_{payer},v_{i+1})}.trust$. However, to hide $d$, $d$ is multiplied with a random number $r$ to obtain a multiplication $m = d.r$. Then, to prevent the factoring of the result [36], $r'$ is added to $m$ and we obtain $c = d.r + r'$. Here, we can replace the addition with the subtraction, and we have $c = d.r - r'$. Actually $r$, $r'$ are used for obfuscating the value of $d$. However, the goal is to do a secure comparison between $t_{min}$ and $T_{(v_{payer},v_{i+1})}.trust$. Therefore, $c$ must be encrypted, and we obtain $E(c)$ as above.

After computing $E_{v_{payer}}(c)$, $v_{i+1}$ sends $(a_1, a_2, a_3) = (E_{v_{i+1}}(0), E_{v_{i+1}}(1), E_{v_{payer}}(c))$ back to $v_i$, where $E_{v_{i+1}}(0)$, $E_{v_{i+1}}(1)$ are encryptions of values '0' and '1' with $v_{i+1}$'s public key. Then, $v_i$ also randomizes two large numbers $r_i$, $r'_i$, and flips

a coin $b \in \{0, 1\}$, then re-calculates $(a_1, a_2, a_3)$:

$$\begin{aligned} a_1 &= a_{1+b} + E_{v_i}(0); \\ a_2 &= a_{2-b} + E_{v_i}(0); \\ a_3 &= (-1)^{b \cdot r_i}(a_3) + (-1)^{(1-b)} \cdot r'_i \cdot E_{v_i}(1); \end{aligned} \tag{2}$$

$b$ is known only to $v_i$. $v_i$ sends $(a_1, a_2, a_3)$ back to $v_{i+1}$. Then, $v_i$ and $v_{i+1}$ collaboratively checks $a_3$. At $v_i$ side, if $a_3 < 0$, we have the boolean expression $[t_{min} \leq T_{(v_{payer}, v_{i+1})}.trust] = 1 - b$; otherwise, we have the boolean expression $[t_{min} \leq T_{(v_{payer}, v_{i+1})}.trust] = b$. With the value of $b$, $v_i$ can learn the result of the comparison. Notice that the boolean expression is 0 when it is false, and it is 1 when it is true.

This secure comparison makes it possible to reduce a large number of messages going through the network, as shown in the performance results reported in Section 8.2.

## 8. Experiments

This section presents the performance of ESP protocol and the flooding optimization.

### *8.1. ESP Performance*

In order to prove the efficiency of ESP, we measure the time an ESP tokenset spends reaching the payee from the payer in several network configurations. Network configurations are defined based on three parameters: the SMNL and SSNL topologies, the wireless network bandwidth, and the adopted encryption algorithms (i.e., their key sizes).

Regarding the first parameter, we set up several SMNL and SSNL topologies, by varying the number of contacts/nodes from 4 to 17. These results in 16 different network configurations, which are summarized in Table 1, where network 1 has the shortest path, including 3 SSNL contacts, and a 8 SMNL hop distance between every 2 SSNL contacts; whereas, network 16 has the longest relationship

path, including 9 SSNL contacts, and a 14 SMNL node distance between every 2 SSNL contacts. With respect to the second parameter, we simulate different wireless network bandwidths. In general, MANET nodes transmit data through different wireless standards. We deploy experiments assuming nodes exploit the popular wireless standard IEEE 802.11a/b/g, and their respective bit rates of 54/11/24Mbps. The third parameter that might impact the performance is the payload size of ESP tokenset. Since we are using encryption schemes, the payload might vary based on the key sizes of the adopted algorithms. In particular, we deploy the UDP protocol for exchanging the tokensets between two mobile devices, because its speed and small payload fit well MANET. The UDP payload size varies according to the key sizes of ECC and ECDSA algorithms. In this experiment, we select four ECC key sizes, that is, 163, 283, 409, 571 bits, and two ECDSA key sizes, that is, 384, 521 bits.

Table 1: Network configurations

| **Network** | **Number of SSNL contacts** | **Number of SMNL nodes between two SSNL contacts** | **Total number of SMNL nodes** |
|---|---|---|---|
| N1 | 3 | 8 | 17 |
| N2 | 5 | 8 | 33 |
| N3 | 7 | 8 | 49 |
| N4 | 9 | 8 | 65 |
| N5 | 3 | 10 | 21 |
| N6 | 5 | 10 | 41 |
| N7 | 7 | 10 | 61 |
| N8 | 9 | 10 | 81 |
| N9 | 3 | 12 | 25 |
| N10 | 5 | 12 | 49 |
| N11 | 7 | 12 | 73 |
| N12 | 9 | 12 | 97 |
| N13 | 3 | 14 | 29 |
| N14 | 5 | 14 | 57 |
| N15 | 7 | 14 | 85 |
| N16 | 9 | 14 | 113 |

Table 2 shows the UDP payload size by using different combination of ECC and ECDSA key sizes. The smallest UDP payload size is 446 bytes, using 163 bit ECC key size and 384 bit ECDSA key size, whereas, the greatest UDP payload size is 1298 bytes, using 571 bit ECC key size and 521 bit ECDSA key size.

These payload sizes do not exceed the limit of a UDP packet size in MANET, i.e., 1460 bytes. Hence, the ESP tokensets are not segmented into more than one packet.

Table 2: UDP payload size with different ECC and ECDSA key sizes

| **ECC key size (Bit)** | **ECDSA key size (Bit)** | **UDP payload size (Byte)** |
|---|---|---|
| 163 | 384 | 446 |
| 283 | 384 | 686 |
| 409 | 384 | 942 |
| 571 | 384 | 1263 |
| 163 | 521 | 482 |
| 283 | 521 | 722 |
| 409 | 521 | 978 |
| 571 | 521 | 1298 |

We used OmneT++[16] and INET framework[17] to set up the different network configurations. OMNeT++ is a component-based C++ simulation library and framework for building network simulators for a variety of application domains, such as sensor networks, wireless ad-hoc networks, photonic networks, etc. Whereas, the INET Framework is an open-source communication network simulation package for the OMNeT++ simulation environment. The INET Framework contains models for several Internet protocols, such as UDP, TCP, SCTP, IP, IPv6, Ethernet, PPP, IEEE 802.11, MPLS, OSPF. By using these tools, we conduct experiments on physical resources including Duo Core CPU 4GHz, 4GB RAM, 64-bit Windows 7.

We measure the delay (ms) spent on transmitting an ESP tokenset from the payer to the payee according to different parameters values. Particularly, the transmission delay includes the time of aggregating an ESP tokenset at each SSNL contact and the time of propagating ESP between SMNL nodes. The experimental results are reported in Figure 20. According to Figure 20, we can see that at the same bit rate the transmission delays in cases of 384 bit ECDSA are higher than the ones of 521 bit ECDSA, and the transmission delays with

[16] http://www.omnetpp.org/
[17] http://inet.omnetpp.org/

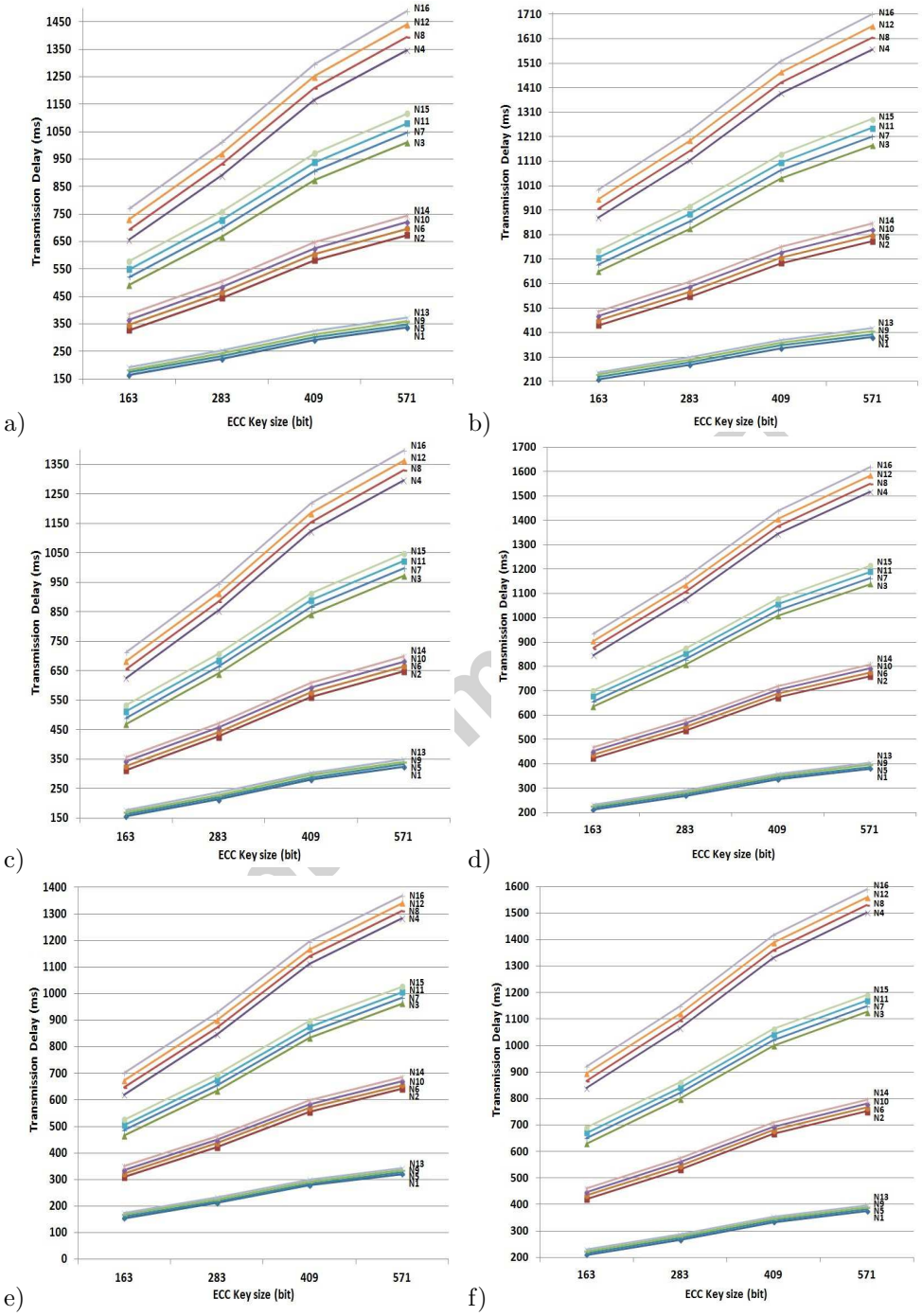



Figure 20: Transmission delay of an ESP tokenset vs key size of ECC and ECDSA vs bit rate standard: a) 11Mbps bitrate and 384bit ECDSA; b) 11Mbps bit rate and 521bit ECDSA; c) 24Mbps bitrate and 384bit ECDSA; d) 24Mbps bit rate and 521bit ECDSA; e) 54Mbps bitrate and 384bit ECDSA; f) 54Mbps bit rate and 521bit ECDSA.

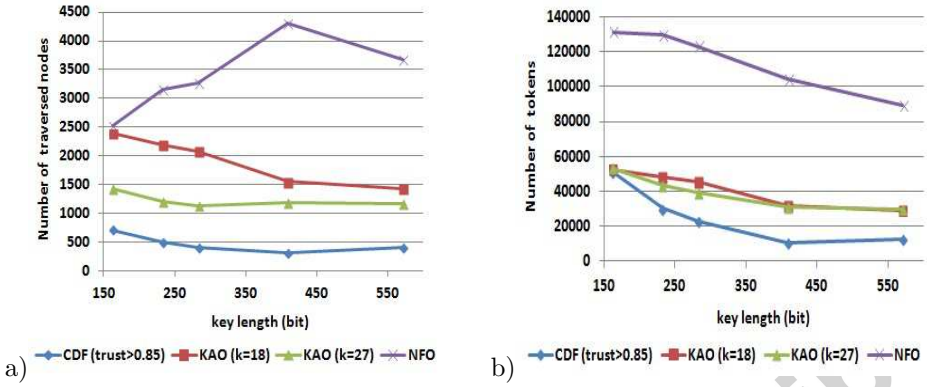



Figure 21: A comparison between NFO, KAO, and CDF
a) Number of traversed nodes per second; b) Number of flooding tokens per second

increasing ECC key sizes are rising. In the worst case (i.e., 11Mbps bitrate, 521 bit ECDSA and 571 bit ECC), the transmission delay for an ESP tokenset to move through 113 SMNL nodes and to aggregate relationship information among 8 SSNL contacts is 1710,4 ms (approximately 1.7s). This transmission delay is reasonable with MANET performance requirement. Therefore, this result proves that the proposed ESP has an effective performance.

### *8.2. Condition-driven Flooding Optimization*

In order to prove the efficiency of the proposed flooding optimization method, we measure the number of generated tokensets propagated through the network and the number of nodes traversed by tokensets, by exploiting the solution proposed in this paper, i.e., the *C*onditional *D*riven *F*looding (CDF), the solution proposed in [1], i.e., the *K*-*A*nonymity based *O*ptimization (KAO), and the one without any optimization, i.e., the *N*o *F*looding *O*ptimization (NFO). The experiment is performed on the trust value. If the trust threshold is very small, CDF is similar to the broadcast technique. If the trust threshold is very high, the tokensets are easily dropped at each intermediate node. Hence, we choose 85% as a threshold in CDF. This threshold is neither very high nor very small. With KAO, to set a good trade-off we have selected k = 18 and k = 27, that is, half and 3/4 of available relationship types in [27]. In this experiment, we

use the Epinion dataset[18]. This is a who-trust-whom online social network of a general consumer review for the site www.Epinions.com. The dataset includes 75.879 nodes and 508.837 edges. We modified the relationship information including a relationship type and a trust value on each edge. Relationship types are uniformly and randomly picked up from [27] based on the ontological FOAF vocabulary specification which provides a set of 35 relationship types. A trust value on an edge $e$ is also randomly generated uniformly from a range [0, 1].

Figure 21 shows that CDF flooding has the best performance. In the case of the largest ECC key size (i.e., 512 bits), the number of tokensets flooding the network with KAO ($k \in \{18, 27\}$) is approximately 2.25 times higher than with CDF, whereas the one with NFO is approximately 7 times higher than with CDF; the number of traversed nodes with KAO (k = 18) is approximately 3.48 times higher than with CDF, and the number of traversed nodes with KAO (k = 27) is approximately 2.85 times higher than with CDF; whereas, the one with NFO is approximately 8.9 times higher than with CDF.

## 9. Security Analysis

Since each ESP tokenset component is encrypted, the tokenset can be considered robust to external eavesdropping. Indeed, the adversaries able to thwart the system are more likely to be playing the roles of participants of the protocol, that is, payer, payee, and intermediate nodes. Based on their behaviors, we classify adversaries into two types, that is, honest-but-curious, and malicious. Honest-but-curious adversaries are intermediate nodes correctly enforcing the protocol by at the same time also looking for extra information. In contrast, malicious nodes might try to modify the ESP token so as to illegitimately retrieve information (e.g., identity, or relationship information). In this section, we discuss how our proposal can resist to both attacks.

[18] https://snap.stanford.edu/data/soc-Epinions1.html

### 9.1. Honest-but-curious nodes

According to the honest-but-curious model, it is expected that nodes comply with the proposed protocol and algorithms, by trying to infer additional private information. In this section, we show how ESP is enough robust to avoid the inference of payee's and payer's identities, and relationship information.

**Scenario 1.** *Inferring payee's identity.* Payee's identity is stored into the Validator component, which is encrypted with the payee's public key. This makes hard for an adversary to learn its plain-text value. However, an intermediate node might try to infer the payee's identity even without decrypting the Validator component. Indeed, since the node has the list of its neighbors identities (i.e., IP addresses, MAC addresses), and the respective public keys, it might try to encrypt neighbor's identity with the corresponding public key and compare the result with Validator content. However, we have to note that this attack is hard to be carried out since Validator's encryption is created based on a large random number generated and held by the payee (see Section 2.2), and the timestamp, generated based on the payee's local system and appended to the pre-encrypted Validator. Moreover, to improve the robustness of the system, we adopt the shuffle algorithm in [34] algorithm so as to obtain a permutation of validator content, computed based on a set of random numbers generated and held by the payee. The adversaries cannot have the above mentioned local parameters, hence, they cannot infer the information of the payee's.

**Scenario 2.** *Inferring payer's identity.* We recall that in SmartPay [1], in order to elaborate the received tokensets, intermediate nodes have to know the public key of the payer, as such they are aware of the payer identity. To avoid this exposure, in ESP, each payer generates a different temporary public key for every payment transaction instead of using its real public key. Thus, any intermediate node is not able to re-identify the payer by the used public key. According to the proposed protocols, the receiving SMNL node can infer information about the sender (e.g., MAC address, IP address). However, since it cannot see any other information as all other components in the ESP tokenset are encrypted,

the receiving SMNL node cannot understand if the sender is indeed the payer and thus the payer indetity.

**Scenario 3.** *Inferring relationship information.* Since SPTokenset content is encrypted by the temporary public key of the payer, an intermediate node cannot access the relationship information, as it does not hold the corresponding private key. However, since the node owns the temporary public key of payers, he might try to encrypt a pre-defined set of values for trust and relationship types and then compare the results with encrypted values in SPTokenset. However, let us recall that each node in ESP, including the payer, needs to locally generate one random number for aggregating a trust/type into SPTokenset using the homomorphic ECC encryption (see Section 2.2). Random numbers are different at every node, which makes encryptions of the same value different. Hence, even if a node uses the temporary public key of the payer for a statistical analysis, it cannot infer trust/type of SPTokenset as well.

### *9.2. Malicious nodes*

Under this attack model, malicious nodes might try to modify the ESP tokenset so as to illegitimately retrieve information (e.g., identity, relationship information). In what follows, we analyze the scenarios of these malicious attacks and discuss how the proposed protocol resists on them.

**Scenario 4.** *Anti-replay attack.* Let us consider the SSNL layer and assume that $v_1^S$ sends a tokenset to $v_2^S$ and that this tokenset passes through node $m$ in the SMNL layer. Let us assume that malicious node $m$ tries to impersonate $v_1^S$ by reusing a copy of the tokenset that has previously passed through $m$. In SecureKey, a parameter, called sessionID, is inserted and used for determining the processed tokensets. When $v_2^S$ receives the same tokenset, it decrypts SecurKey's encryption with its private key and checks sessionID. In this case, $v_2^S$ verifies that this tokenset has been already processed, then it drops the tokenset, avoiding thus the replay attack.

**Scenario 5.** *Altering tokenset content.* According to this scenario, we assume that a malicious node wishes to modify the tokenset content, by replacing one of the elements of the tokenset (i.e., SecureKey, SPTokenset, Validator), and/or by simply inserting/deleting some bits, so as to invalidate the tokenset. To detect the corrupted tokensets, we impose that the ESP is digitally signed by the node processing the ESP tokenset. Since a malicious node does not hold the private key of ESP signature, it cannot generate a new one for the modified ESP tokenset. Thus, the node receiving this ESP tokenset can detect that its content has been maliciously modified, by validating the original digital signature.

## 10. Conclusions

In this paper, we proposed a privacy-preserving path discovery protocol, namely ESP, in support of trust preference enforcement in decentralized mobile payment systems exploiting MANET. The ESP protocol is able to protect the relationship information in terms of the type, depth, and trust of the discovered paths. We propose some optimization strategies to decrease the number of tokensets sent over MANET to improve the network performance. We prove the efficiency of the system by experimental results. Future work includes handling offline nodes, and deploying a full ESP prototype into a real setting.

## References

[1] B. Carminati, E. Ferrari, N. H. Tran, Enforcing trust preferences in mobile person-to-person payments, in: Social Computing (SocialCom), 2013 International Conference on, IEEE, 2013, pp. 429–434.

[2] V. Katiyar, K. Dutta, S. Gupta, Article:a survey on elliptic curve cryptography for pervasive computing environment, International Journal of Computer Applications 11 (10) (2010) 41–46.

[3] K. I. Lakhtaria, K. I. Lakhtaria, Technological Advancements and Applications in Mobile Ad-Hoc Networks: Research Trends, 1st Edition, IGI Global, Hershey, PA, USA, 2012.

[4] G. R. S., New trends in mobile ad hoc network, International Journal of Ethics in Engineering and Management Education (IJEEE) 1 (4).

[5] J. Hoebeke, I. Moerman, B. Dhoedt, P. Demeester, An overview of mobile ad hoc networks: Applications and challenges, Journal-Communications Network 3 (3) (2004) 60–66.

[6] N. Kotilainen, M. Weber, M. Vapa, J. Vuori, Mobile chedar-a peer-to-peer middleware for mobile devices, in: Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops. Third IEEE International Conference on, IEEE, 2005, pp. 86–90.

[7] E. Sarigöl, O. Riva, P. Stuedi, G. Alonso, Enabling social networking in ad hoc networks of mobile phones, Proceedings of the VLDB Endowment 2 (2) (2009) 1634–1637.

[8] A.-K. Pietiläinen, E. Oliver, J. LeBrun, G. Varghese, C. Diot, Mobiclique: middleware for mobile social networking, in: Proceedings of the 2nd ACM workshop on Online social networks, ACM, 2009, pp. 49–54.

[9] M. Mehdi, A. Nguyen, C. Noel, Whats up 2.0: P2P Spontaneous Social Networking, in: Proceedings of the IEEE International Conference on Pervasive Computing and Communications, 2009, pp. 1–2.

[10] P. Goyal, V. Parmar, R. Rishi, Manet: Vulnerabilities, challenges, attacks, application, IJCEM International Journal of Computational Engineering & Management 11 (2011) (2011) 32–37.

[11] D. Johnson, A. Menezes, S. Vanstone, The elliptic curve digital signature algorithm (ecdsa), International Journal of Information Security 1 (1) (2001) 36–63.

[12] C. K. Toh, Ad Hoc Mobile Wireless Networks: Protocols and Systems, Prentice Hall, 2001.

[13] E. Ferro, F. Potorti, Bluetooth and wi-fi wireless protocols: a survey and a comparison, Wireless Communications, IEEE 12 (1) (2005) 12–26.

[14] B. Xiao, J. Cao, Z. Shao, Q. Zhuge, E. H.-M. Sha, Analysis and algorithms design for the partition of large-scale adaptive mobile wireless networks, Computer communications 30 (8) (2007) 1899–1912.

[15] G. Dini, M. Pelagatti, I. M. Savino, An algorithm for reconnecting wireless sensor network partitions, in: Wireless Sensor Networks, Springer, 2008, pp. 253–267.

[16] R. L. Rivest, L. Adleman, M. L. Dertouzos, On data banks and privacy homomorphisms, Foundations of Secure Computation, Academia Press (1978) 169–179.

[17] S. Nakamoto, Bitcoin: A peertopeer electronic cash system, http://bitcoin.org/bitcoin.pdf (2009).

[18] I. Miers, C. Garman, M. Green, A. D. Rubin, Zerocoin: Anonymous distributed e-cash from bitcoin, in: Security and Privacy (SP), 2013 IEEE Symposium on, IEEE, 2013, pp. 397–411.

[19] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, S. Capkun, Evaluating user privacy in bitcoin, in: 17th International Conference,Financial Cryptography and Data Security, Vol. 7859, 2013, pp. 34–51.

[20] A. Pfitzmann, M. Köhntopp, Anonymity, unobservability, and pseudonymitya proposal for terminology, in: Designing privacy enhancing technologies, Springer, 2001, pp. 1–9.

[21] K. Herrmann, M. A. Jaeger, Payflux – secure electronic payment in mobile ad hoc networks, in: Information and Communications Security, Springer, 2004, pp. 66–78.

[22] J. T. Isaac, S. Zeadally, J. S. Cámara, A lightweight secure mobile payment protocol for vehicular ad-hoc networks (vanets), Electronic Commerce Research 12 (1) (2012) 97–123.

[23] W. Li, Q. Wen, Q. Su, Z. Jin, An efficient and secure mobile payment protocol for restricted connectivity scenarios in vehicular ad hoc network, Computer Communications 35 (2) (2012) 188–195.

[24] T. B. Ramkumar, T. Kalaikumaran, S. Karthik, Trust based secure payment scheme for multi-hop wireless networks, International Journal of Advanced and Innovative Research 3 (5) (2014) 173–177.

[25] A. Michalas, V. A. Olesh, N. Komninos, N. R. Prasad, Privacy Preserving Trust Establishment Scheme for Mobile Ad-hoc Networks, in: IEEE International Conference on Communications, 2011, pp. 752–757.

[26] J. Golbeck, Personalizing applications through integration of inferred trust values in semantic web-based social networks, in: Semantic Network Analysis Workshop co-located with ISWC'05, 2005.

[27] I. Davis, V. J. E., A vocabulary for describing relationships between people, http://vocab.org/relationship/ (2013).

[28] P. Mika, A. Gangemi, Descriptions of social relations, http://www.w3.org/2001/sw/Europe/events/foaf-galway/papers/fp/descriptions_of_social_relations/.

[29] G. Liu, Y. Wang, M. A. Orgun, Trust transitivity in complex social networks., in: AAAI, Vol. 11, 2011, pp. 1222–1229.

[30] B. King, A point compression method for elliptic curves defined over $gf(2^n)$, in: Work. Theory and Practice in Public Key Cryptography, 2004.

[31] B. King, Mapping an Arbitrary Message to an Elliptic Curve when Defined over $GF(2^n)$, Int. Journal of Network Security 8 (2).

[32] P. Eagle, S. Galbraith, J. Ong, Point compression for koblitz elliptic curves, Advances in Mathematics of Communication 5 (1).

[33] L. Sweeney, k-anonymity: A model for protecting privacy, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10 (05) (2002) 557–570.

[34] R. Durstenfeld, Algorithm 235: Random permutation, Commun. ACM 7 (7).

[35] F. Kerschbaum, D. Biswas, S. de Hoogh, Performance comparison of secure comparison protocols, in: Database and Expert Systems Application, 2009. DEXA'09. 20th International Workshop on, IEEE, 2009, pp. 133–136.

[36] E. Kiltz, G. Leander, J. Malone-Lee, Secure computation of the mean and related statistics, in: Theory of Cryptography, Springer, 2005, pp. 283–302.