Published in Proceedings of VEHITS 2018 : 4th International Conference on Vehicle Technology and Intelligent Transport Systems, 16-18 March 2018, Funchal, Portugal, which should be cited to refer to this work. Website of the conference : http://www.vehits.org/?y=2018

HACIT: a privacy preserving and low cost solution for dynamic navigation and Forensics in VANET

Decoster Kevin[1] and Billard David[1]
[1]*University of Applied Sciences Western Switzerland in Geneva - HES-SO*
kevin.decoster@hesge.ch, david.billard@hesge.ch

Keywords: VANET, D2D, Distributed Systems, Ad-Hoc network, Privacy, Blockchain, Forensics

Abstract: The current architecture for VANET related services relies on a Client-Server approach and leads to numerous drawbacks, such as network congestion due to the bottleneck problem or, more importantly, data privacy concerns. Indeed, because of the network topology, traffic must go through nodes which limit the bandwidth and thus bound the overall network capacity. Finally, user data is collected and stored in servers, used by third party services. However, these providers are known to treat lightly user privacy by selling or using the data for their own purposes (Beresford and Stajano, 2004). By use of a decentralized and distributed communication protocol (*i.e.* D2D), one can overcome these problems by spreading the communication burden to all nodes in the mesh. By means of cryptographic techniques, we can ensure that the shared data is secured and controlled at the end-user side. This paper presents a study and proposes a proof of concept of a decentralized and distributed information system by means of a dynamic navigation system for VANET, using a low-cost solution such as Wifi or LTE-direct new 3GPPP protocol. This system preserves user privacy and is augmented with forensics capabilities.

1 Introduction

While many vendors offer IoT devices or propose cars with embedded networked devices, the notions of data privacy and security are often considered as a minor point, when the list of the device features is enumerated. However, law enforcement services, specialized services, and digital forensic experts witness a dramatic surge in criminal data leaks, espionage, social engineering exploits and abuse of industrial control systems (SCADA). The last semi-annual report from MELANI (Melani, 2006), the Swiss agency dealing with cybersecurity within the Federal Intelligence Service (NDB / SRC), stresses the importance of handling the security vulnerabilities. At the EU level, member states have consistently tried to implement legal frameworks to protect the processing of personal data and the free movement of such data within EU (Directive, 1995) and in April two year ago, a directive imposed a binding legal framework (EU, 2016). Therefore, all this constitutes sufficient warning to alert the security community and to trigger advanced research. The research presented in this paper concerns the protection of user privacy but also the necessity for law enforcement to gather evidence (”digital forensics”).

The proof of concept presented in this paper focuses on the collaboration of intelligent cars for determining the best driving route and avoiding traffic jams using only local information transmitted by the other cars, without using a central Internet service. By forbidding the use of centralized services like Google Maps or Tomtom Go Mobile, the user private data is kept at the user's premises and privacy is strengthened. For that purpose, we designed a decentralized and distributed communication scheme for Vehicular Ad-Hoc Network (VANET) with forensics capabilities. In order to (1) maintain data privacy at the user side and (2) avoid to rely on costly and embedded hardware such as embedded radar or computers in cars, we focus on a low-cost solution using mobile to mobile device communications (D2D) on Android device, and validate our model on numerical simulation aided with a VANET simulator such as GAMA (Grignard et al., 2013) or Veins (OMNET++ and SUMO) (Veins, 2011).

The modeled system can be decomposed into three layers:

1. The *communication layer*, which ensures Confidentiality, Integrity, and Availability (CIA) using symmetric and asymmetric cryptography (end to end). Furthermore, we must address the problem

of access control (Identification, Authentication, and Authorization) in a strong dynamic environment and anonymity (*e.g.* by the use of temporal Id's);

2. The *message dissemination* layer: once the secure communication channel is established, the vehicles (nodes) must send information about the traffic, current attributes, etc., while addressing the problem of network overload;
3. The *application layer*, for the processing of the gathered data. The two applications foreseen in our testbed are (1) the rerouting using an off-line partially stored map (*i.e.* OpenstreetMap API) and (2) the forensics capability using a blockchain method.

Furthermore, confidentiality between nodes is ensured by asymmetric and symmetric cryptography while authentication and forgery-proof logging will be ensured by blockchain technology.

The technology enabling communication is a framework called GRCBox (Tornell et al., 2015), installed on a Raspberry Pi device, which allows ad-hoc networking on Android phones. This paper leaves aside the study of the new proposed technology ProSe (Prasad et al., 2014), an LTE based technology which directly compete with the Dedicated Short Range Communication (DSRC) algorithm used for VANET networking. It was proposed in Release 12 of the 3GPP specification for device to device communication using licensed spectrum. While the deployment is still in progress, we foresee that this technology can enhance our proposed algorithm. More generally, the communication medium capability is a key research issue in our work.

The rest of the paper is organized as follow. Section 2 expresses the current state of the art regarding the proposed functionalities. Section 3 presents the chosen system models and answers the problem of communication, message dissemination and dynamic rerouting. Section 4 describes the forensics implementation and section 5 concludes the paper.

2 Related work

The first layer of our system is the creation of an ad-hoc network between in-range nodes (vehicles). The targeted application (*i.e.* VANET communication) leads to highly dynamic and changing network, meaning a node is expected to connect and disconnect to a particular network fast. Two cars moving in opposite directions at $50km/h$, with a WIFI range of 100m leads to a gross calculus of a maximum transmission duration of around 7s. This means that, in order to connect two nodes, one cannot take longer than a dozen seconds to perform all connexion stages. As a consequence, we cannot use the Wifi-Direct option, available on stock Android smartphone device, as explained in (Funai et al., 2015). Although too slow and power consumer, a solution for rooted device, the Serval project (Serval project, 2016), provides a full mesh network without external hardware. From the authors' knowledge, only the GRCBox framework enables a free and open-source solution for ad-hoc network deployment for Android devices by using an external embedded hardware such as a Raspberry Pi. However,they are recent crowd-sourced solution such as GoTenna (GoTenna, 2016) that propose a proprietary dedicated hardware for mesh networking based radio, which is also under consideration.

Regarding decentralized and distributed device to device communication, we use the solution proposed in (Kaul et al., 2017), focusing on efficient and dynamic message dissemination in ITS, to draw our communication model, and avoid the broadcast storm problem while efficiently delivering traffic information messages. More specifically to our rerouting application, the paper (Leontiadis et al., 2011) proposes a dynamic routing application. However, they don't address the limitations of communication within VANET and they assume the use of external infrastructures. As a matter of fact, the more recent paper (Garip et al., 2015) proposes a suboptimal offline rerouting solution while addressing the communication problems that might arise in VANET (*i.e.* request and response traffic information approach). Nonetheless, the security and forensic are not fully addressed.

To the best of our knowledge, although the security in VANET is a well researched field ((Raya and Hubaux, 2007)), no paper fully addresses the forensics concern. Often, they assume a third party TA, for real identity recovery in a secure and anonymous network. However, trusting an external TA would break the proposed project requirements. This is where forensics logging comes into play: the whereabouts and exchanged communication are logged into the own user's system while ensuring unforgeability, via blockchain. For instance, (Sharma et al., 2017) and (Leiding et al., 2016) use blockchain in VANET. However, they use it for monetary applications such as an automatic smart contract for insurance or tolling. The most used blockchain network enabling smart contract alongside cryptocurrency is Ethereum (see (Wood, 2014)). However, without the need for a cyptocurrency support (and thus proof of work through mining), our blockchain can achieve consensus via randomness and thus avoid-

ing the computational burden of mining. Therefore, our application will be designed on top of a permissioned blockchain framework called Hyperledger fabric (linux Foundation, 2016).

3 System models

In this section, we present the model design of our decentralized system.

3.1 High dynamic Ad-Hoc network

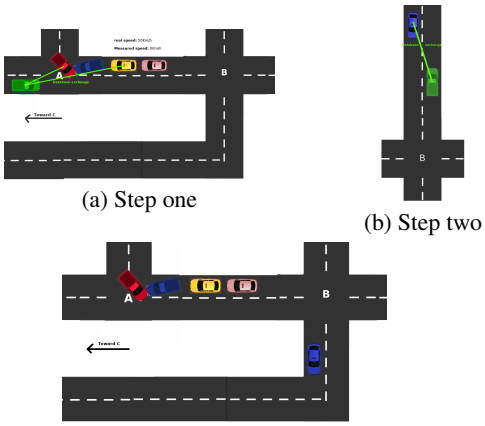
The GRCBox framework installed on a Raspberry Pi 3 model B, with 2 Wifi interfaces, enables ad-hoc networking. This architecture allows to communicate with other cars or with Internet via outer wireless interfaces and to the application via the inner interface remotely (Wifi). The framework handles the messages received and forwards them to every outer Wifi interfaces of nodes connected to the ad-hoc network. The authors of (Hadiwardoyo et al., 2017) uses the GRCBox to enable communication between smartphones in order to test a simple application and asses the performance of the framework. Interestingly, this system can deliver message at $10HZ$, up to 80 meters and within a delay from 100ms to 900ms. Thus, it drives us to conclude that this system is really suitable for our practical implementation.

Finally, on the application layer, we propose an end-to-end secure communication channel and authentication, using the asymmetric cryptography and the blockchain (see Section 4).

3.2 Scalable message dissemination

Each vehicle stores a weighted graph G, containing for each road segment a weight representing the time needed to travel this segment and used for offline shortest path computation (we are using the well-known Dijkstra (Dijkstra, 1959) algorithm). Alongside this graph, it stores also a database DB (as an Hashmap) containing information measured as the vehicle goes through roads or resulting from the union of other nodes databases. Each entry is of the form $< RoadId, AvgSpeed, Timestamp >$. $RoadId$ is a unique key, meaning that if a new entry is available for this $RoadId$, the system will simply keep the latest update. $AvgSpeed$ is the measured speed on the road segment with id $RoadId$. Finally, $Timestamp$ stores the time at which the speed sample was measured.

Whenever a vehicle joins a network, it sends its own DB to every node in the joined network. As a reply, the receiving nodes send all entries in its database whose road ID differs from DB or are simply more recent.



(a) Step one

(b) Step two

(c) Step three

Figure 1: An example of data dissemination

With this approach, we obtain a simple but efficient 1-hop broadcasting scheme (or also called 1-persistent broadcast), which avoids the "broadcast storm" problem in highly dense traffic. An example of the dissemination technique for our delay resistant application can be found in figure 1. In subfigure 1a, we observe a car accident at crossroad A, while a green car comes in the opposite direction and proceeds to the database exchange with the stopped cars. In subfigure 1b, the green car exchanges its database (with entry at road AB: $AvgSpeed = 0$) with the blue car. Finally, in subfigure 1c, the blue car updates its route and reroutes its course, and thus, it avoids the traffic jam in crossroad A.

3.3 Dynamic Navigation rerouting

The rerouting algorithm proposed in this paper uses a weighting schemes where weights are stored locally (as a graph) and later used with an optimal shortest path algorithm.

- *Weighting scheme*. We recall that the weights of the roadmap graph are simply computed as the time needed to travel the road segment. In other words, given the road segment between edges i and j, the weight $w_{i,j}$ is computed as $w_{i,j} = \frac{l_{i,j}}{v_{i,j}}$, with l and v being the road length and average speed respectively (information available on the road graph, alongside other parameters such as the number of lines, type of road $etc.$). Given a new estimated average speed $\hat{v}_{i,j}$ measured at time t, and thus, the corresponding weight $\hat{w}_{i,j}$, we can compute the new weight as in Equation 1, with T_2 being a time threshold after which the data is considered unreliable.

$$w_n^{i,j} = \alpha \cdot w_{n-1}^{i,j} + (1-\alpha) \cdot \hat{w}^{i,j} \quad with\, \alpha = \frac{t}{T_2}. \quad (1)$$

- *Procedure*. Proceed to a simple navigation rerouting, based on newly received traffic information from other nodes, which can be found in Algorithm 1 and summarized as follow:

  After receiving a message containing the gathered road informations db_r from another node database, where each entry contains a structure of the form $< RoadID, avgSpeed, Timestamp >$, we can distinguish three steps: The road segment weight update, the navigation path rerouting computation and the database update.

  1. For each entry in db_r, we update the entry of the weighted graph G corresponding to the same road segment (*i.e.* with same *RoadId*) using Equation 1. Then, we verify (1) if the difference of speed is bigger than a certain threshold (*i.e.* T_1) (2) if it has not expired regarding a time threshold T_2 and (3) if the road segment path list R contains this road segment and we store the boolean result in a global variable.
  2. If the variable containing the result of the previous step tests is true, we must recompute the shortest path (using Djisktra) given the new updated graph G, the current position and the destination (*i.e.* *R.end*).
  3. Finally, we merge the received set db_r to the database *DB* containing the traffic information sent by other nodes and recorded by the user, keeping only the latest database entry.

  The full algorithm is shown in Figure 1.

4 Privacy and forensics

Section 3 presents a simple but complete communication model in order to connect nodes in communication range and exchange packet frames. Once the traffic information is filtered and stored in each node, it is processed to update road weights in the roadmap graph used in another run of Dijkstra algorithm to find the new shortest path, if different. However, the innovative approach in this work is to tackle the problem of a complete decentralized and secure system while enabling forensics capabilities. We can divide the problem into (1) the forensics system and (2) the security between nodes.

Algorithm 1 Navigation rerouting algorithm upon newly received traffic information

Require: db_r,*R*, *G*.

1: db_r is a list containing a structure of $[RoadID, AvgSpeed, Timestamp]$, R and G are the road path list and the road weighted graph respectively

2: **function** PATH REROUTING(db_r,*R*, *G*)
3: $isModify \leftarrow False$
4: **for** d in db_r **do**
5: $t \leftarrow d.Timestamp$
6: $w \leftarrow G[d.RoadID].Weight$
7: $v \leftarrow d.AvgSpeed$ ▷ New speed
8: $n \leftarrow$ current time
9: $G[d.RoadID] \leftarrow$ Updated as Eq 1
10: $isModify \leftarrow ||w-v|| \leq T_1$ & $||t-n|| \leq T_2$ & $R.contains(d.RoadId)$
11: **if** isModify **then**
12: $R \leftarrow DJISKTRA(G, position, R.end)$
13: $DB \leftarrow DB \cup db_r$ ▷ Keep latest information

4.1 Forgery-proof logging with Blockchain

Our original contribution to the VANET community is to propose a decentralized VANET communication network without relying on any external third party. However, the need to provide evidence might arise, such as in car accident or police investigation (*eg*: a previous car behavior), in order to establish responsibilities. Due to the problem specification, such as low bandwidth capacity and computation on a mobile device, we propose that the user stores in its local storage *blocks* containing logged data and chained with hash, similar to a blockchain. The hashing of data guarantee a one way encryption function, meaning it guarantees that the data is untouched as long as the hash submitted is immutable. Obviously, we don't want the user to modify this data considering forensics fraud. Therefore, we propose the following scheme (see Figure 2):

- A user initially submits a smart contract (also known as chaincode) similar to Figure 3, which contains a public hash, the owner public key and a public function executable only by the owner of the Smart Contract, to the network consensus. This step is used as registration. In other words, once the smart contract code is compiled, the new user needs to submit the contract to the other peers for consensus. Upon validation, the user received a confirmation alongside the hash of his smart

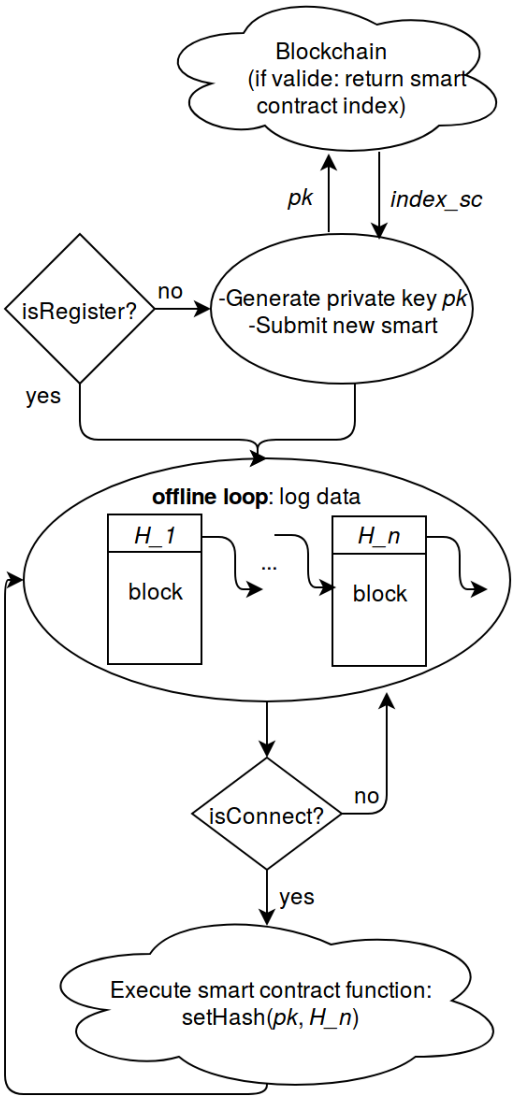


Figure 2: Blockchain pipeline

```
contract Client_contract {
  bytes[] hashes = new bytes()[N];
  uint[] validation = new uint()[N];
  address public pk = msg.sender;

  fct sethash(bytes[] sHash, address[] oPk) {
                require(msg.sender == this.pk);
        require(sHash.length== oPk.length);
        for(uint i =0; i< sHash.length; i++){
                #no validation for this hash
                if(oPK[i] == null){
                this.hashes.push(h);
                        this.validation.push(0);
                }
            else{
                byte[] h = SHA256(sHash[i]);
                byte[] h_p= decrypt(decrypt(sHash[
                     i], oPk[i]), pk);
                if(h==h_p) push(sHash[i])
            }
        }
        }
    fct push(bytes a){
        if(hashes.contains(a)){
                uint i = hashes.indexof(a);
            hashes[i] = a;
            validations[i] += 1;
        }
        else{
                hashes.push(a);
            validations.push(1);
        }
    }
}
```

Figure 3: Proposed initial Ethereum smart contract

contract (*i.e.* contract index). An example of an initiating smart contract is shown in 3.

- While the user uses the system, it logs a block of data (*i.e.* an offline loop) and computes the hash of the newest block (*i.e.*: blockchain).
- When user connect to network, it sends the latest hash alongside its database (road informations). The receiving node returns its database as well as the signed hash.
- When a user is connected to the Internet through Wi-Fi or cellular network, it executes the *setHash* function given its accumulated hash (signed or not).
- Every time a node is connected to the network, it downloads the new blockchain version (*i.e.* up-to-date hash from all nodes).

This scheme allows the user to verify, using the hash available in his or her smart contract, that the data he/she owns is valid and not modified. Moreover, encountered nodes sign the latest hash which add a layer of proof. A legal entity can then check the ledger to see which submitted hash were endorsed by other peers.

Without the need for cryptocurrency (*i.e.* monetary scheme), the proof-of-work is useless here. Thus, the Byzantine fault tolerance consensus algorithm can be achieved by simply selecting a peer ”leader” randomly, which will gather all transactions (smart contract execution) into a new block. As for most blockchain technologies, this block is then broadcast eventually to all peers with the gossip protocol. Therefore, such a consensus algorithm is considered computationally easy and suitable for mobile application.

4.2 Inter-node security

Furthermore, thanks to the scheme, a receiver node can retrieve the public key of the sender, given the sender’s smart contract index (*i.e.* smart contract hash

address, considers unique among users and serving to identify), in the blockchain. An example of the exchange can be seen in figure 4. In other words, we guarantee that the data remains confidential within our service users (*i.e.* valid registered users) while being able to authenticate data and avoid malicious injection, by means of asymmetric encryption. Indeed, a node A maintains an up-to-date and encrypted version of its database C, containing gathered road traffic informations, using his private key K_a. This encrypted data is computed at a certain interval considering a trade-off between power consumption and the novelty of the data. Once it has joined a new network, it broadcasts its encrypted data C to all other nodes alongside its blockchain identity H_sc (smart contract address). Upon message reception, the receiving node uses the unencrypted A's id H_sc to retrieve A's public key PK_a into the permissioned ledger that it has already access to. If no address matches or no public key available, A is not registered to the network. Otherwise, we use PK_a to decrypt C. A challenge is finally tested to assess the validity of the data.

By means of blockchain functionalities, we propose a simple way of encryption and authentication. Of course, it implies that the user needs to have an up-to-date ledger in order to acknowledge newly subscribed users and retrieve their public key, and that the data is secured and validated by the community.

4.3 Malicious node

The proposed model allows for an user to prove the data provenance and its integrity based on validation from other encountered users. However, a malicious node could alter its data while still submitting a valid hash. Thus, we propose the following extensions:

- White-box cryptography (Brecht, 2012): While this field is still under study, it proposes a way to hide private key into the source code. In our case, it could secure the submitted hash, providing a way for any nodes to verify that it has been signed by our application (and not submitted *a posteriori*).

- 50% overlaps: Each block of data will overlaps each other, to help check data consistency at the forensics level.

However, the incentive to submit as much valid and correct hashes as possible in order to protect an user legally will eventually protect against malicious behaviour.

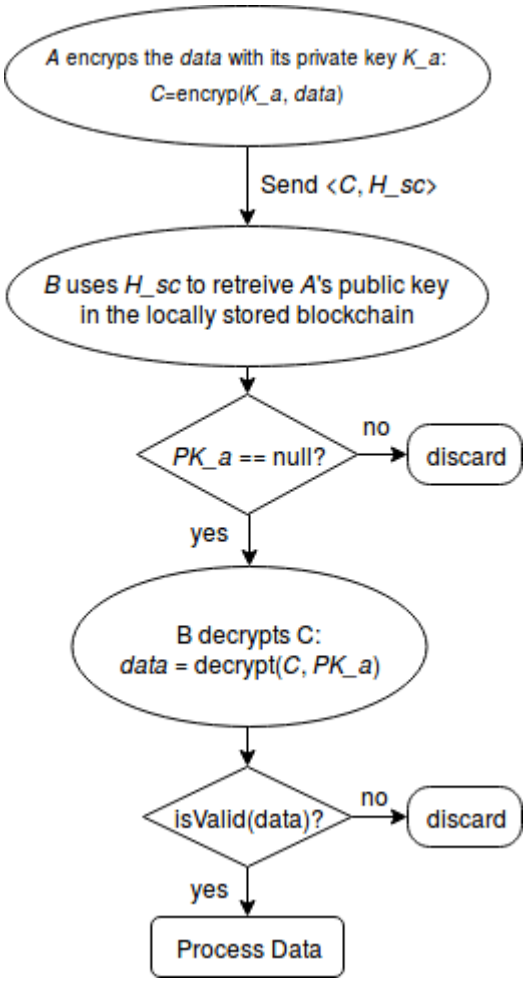


Figure 4: Authentication and privacy pipeline

5 Conclusion

Using well known techniques, we propose a real-time low-cost system able to connect vehicles in a network and exchange traffic information in order to dynamically reroute user to spare overall time. With a simple raspberry pi and GRCbox framework, we enable multi interface short-range communications through an ad-Hoc network. We opt for a simple 1-hop broadcasting technique in order to share each node database, containing the accumulated traffic knowledge. Once new information is received about road segment on the navigational path, one can use our simple but yet efficient re-routing algorithm.

The dynamic routing application in this project is the test-bed for our innovative contribution: we propose a complete and secure decentralized system that preserves user privacy, since no computation is made at a central server that could gather private data. In addition, the system enables forensics logging within

a network made exclusively of mobile device: by use of data hash stored in a lightweight distributed blockchain through Smart Contract, we allow each user to be able to prove its own data integrity. Furthermore, this blockchain is used to retrieve nodes public keys and thus, authenticate and decrypt packets.

5.1 Discussion on mobile computing

Given the nature of our proposed solution, we foresee several problems and points that must be addressed:

- Given the communication window between nodes, the data exchanged must be as concise as possible. To that end, we can prune the data to be shared to include only latest gathered road information.
- The computation of the navigation path might be a challenge for mobile devices. However, nowadays smartphones can handle heavy computational burden and finding the shortest path is a well known and optimized algorithm.
- We encrypt the data exchange between nodes with asymmetric encryption. Such encryption is however computationally expensive.
- The frequency of smart contract execution and ledger update will directly affect the efficiency of the proposed solution but also the power and computational requirements.

5.2 Future work

The solution that we propose is yet to be developed. The work will question different sensitive subjects specific to VANET limitations. For instance, we foresee that the size of the database to be shared over a really small amount of time might be a challenge, or that the computing power available on mobile device might not be sufficient to run Dijkstra or asymmetric cryptography. The latest project break-in involves a complete different approach. Indeed, the hyperledger Fabric functionalities allow to store key/value pairs alongside the ledger of transactions as permissioned blockchain. A user would then directly submit a new transaction containing a new measured speed for a certain road id as transaction. Such scheme enable forensics through transactions crawling and dynamic rerouting through database query. However, the scalability of the system for mobile application is yet unknown.

REFERENCES

Beresford, A. R. and Stajano, F. (2004). Mix zones: User privacy in location-aware services. In *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, pages 127–131. IEEE.

Brecht, W. (2012). White-box cryptography: hiding keys in software. *NAGRA Kudelski Group*.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

Directive, E. (1995). 95/46/ec of the european parliament and of the council of 24 october 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. *Official Journal of the EC*, 23(6).

EU (2016). Directive (EU) 2016/680 of the European Parliament and of the Council of 27 April 2016. `http://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:32016L0680`.

Funai, C., Tapparello, C., and Heinzelman, W. (2015). Supporting multi-hop device-to-device networks through wifi direct multi-group networking. *arXiv preprint arXiv:1601.00028*.

Garip, M. T., Gursoy, M. E., Reiher, P., and Gerla, M. (2015). Scalable reactive vehicle-to-vehicle congestion avoidance mechanism. In *Consumer Communications and Networking Conference (CCNC), 2015 12th Annual IEEE*, pages 943–948. IEEE.

GoTenna (2016). GoTenna. `www.gotenna.com`. [Online; accessed 21-November-2017].

Grignard, A., Taillandier, P., Gaudou, B., Vo, D. A., Huynh, N. Q., and Drogoul, A. (2013). Gama 1.6: Advancing the art of complex agent-based modeling and simulation. In *International Conference on Principles and Practice of Multi-Agent Systems*, pages 117–131. Springer.

Hadiwardoyo, S. A., Patra, S., Calafate, C. T., Cano, J.-C., and Manzoni, P. (2017). An android its driving safety application based on vehicle-to-vehicle (v2v) communications. In *Computer Communication and Networks (ICCCN), 2017 26th International Conference on*, pages 1–6. IEEE.

Kaul, A., Obraczka, K., Santos, M., Rothenberg, C., and Turletti, T. (2017). Dynamically distributed network control for message dissemination in its. In *IEEE/ACM DS-RT 2017-21st International Symposium on Distributed Simulation and Real Time Applications*.

Leiding, B., Memarmoshrefi, P., and Hogrefe, D. (2016). Self-managed and blockchain-based vehicular ad-hoc networks. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, pages 137–140. ACM.

Leontiadis, I., Marfia, G., Mack, D., Pau, G., Mascolo, C., and Gerla, M. (2011). On the effectiveness of an opportunistic traffic management system for vehicular networks. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1537–1548.

linux Foundation, T. (2016). HyperLedger Fabric. https://hyperledger-fabric.readthedocs.io/en/release/. [Online; accessed 21-November-2017].

Melani (2006). MELANI, Reporting and Analysis Centre for Information Assurance. https://www.melani.admin.ch/melani/en/home.html. [Online; accessed 21-November-2017].

Prasad, A., Kunz, A., Velev, G., Samdanis, K., and Song, J. (2014). Energy-efficient d2d discovery for proximity services in 3gpp lte-advanced networks: Prose discovery mechanisms. *IEEE vehicular technology magazine*, 9(4):40–50.

Raya, M. and Hubaux, J.-P. (2007). Securing vehicular ad hoc networks. *Journal of computer security*, 15(1):39–68.

Serval project (2016). Serval project. http://www.servalproject.org/. [Online; accessed 21-November-2017].

Sharma, P. K., Moon, S. Y., and Park, J. H. (2017). Block-vn: A distributed blockchain based vehicular network architecture in smart city. *Journal of information processing systems*, 13(1):184–195.

Tornell, S. M., Patra, S., Calafate, C. T., Cano, J.-C., and Manzoni, P. (2015). Grcbox: extending smartphone connectivity in vehicular networks. *International Journal of Distributed Sensor Networks*, 2015.

Veins (2011). VEINs simulator. veins.car2x.org/. [Online; accessed 21-November-2017].

Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151.