Traceability for Vehicular Network Real-Time Messaging Based on Blockchain Technology

Chi-Sheng Shih*, Wei-Yu Hsieh, and Chia-Lung Kao
National Taiwan University, 10617 Taipei, Taiwan

Received: November 1, 2019; Accepted: December 7, 2019; Published: December 31, 2019

Abstract

Autonomous vehicles are capable of navigating without any intervention of human based on onboard sensor data and wayside sensor data. However, when an incident occurs, responsibility and liability must be indubitably decided based on those data received by vehicle in the past. Without those data or if the stored data is incorrect, one cannot reproduce the accident's scene correctly which will lead to incorrect result of accident forensic process. Therefore, we need a secure and real-time data logging system to record the data received by each vehicle. In this work, we propose a secure, real-time, and distributed data logging system with higher fault tolerance and scalability based on Blockchain technology, which enables autonomous vehicles log every data they received in real-time and ensure data integrity. In addition to Blockchain technology, digital signature and cryptography hash function are also included in our proposed system in order to detect if the stored data has been tampered. If the data is tampered, the system is able to detect the time period during which the data is tampered. With these properties, our system is capable of deploying in VANET and the stored data will be secure and reliable.

Keywords: Vehicular network, block-chain, traceability.

1 Introduction

Autonomous vehicles are capable of navigating without any human input by sensing its environment. In order to achieve self-driving, autonomous vehicles adopt not only various sensory technologies, such as camera, RADAR, and LIDAR, to sense its environment but also communication technology, e.g., DSRC [7], to receive information from other systems.

When autonomous vehicles are involved in the accident, how to determine the cause of accident and determine liability is one of the most important things. One approach is that if we can record all the data (both sensor and tele-matic data) received by autonomous vehicles all the time, the system can analyze those data to determine the cause. Therefore, it is desirable to have a secure and real-time data logging systems.

As mentioned in [1], a well designed logging system that can provide traceability of data needs to meet the following requirements:

- Make sure the recorded data is not tampered since perpetrator might want to tamper those recorded data to avoid liability,
- Make sure recorded data is not able to read by anyone except authorized party, and

*Corresponding author: Department of Computer Science and Information Engineering, National Taiwan University, 10617 Taipei, Taiwan, Tel: +886-2-3366-4927, email: cshih@csie.ntu.edu.tw

- Make sure the recorded data is not stored in plain-text.

Beside those requirements mentioned above, since our design is focus on vehicular network and recorded data is often used for forensic purpose, there are other challenges to tackle:

- **Scalability**: the designed system can be deployed and operated in vehicular network for large number of vehicles
- **Non-repudiation of recorded data**: the sender of each recorded data cannot dispute its authorship. This requirement is needed, since the sender of the malicious data may try to dispute its authorship to avoid liability.
- **Enhanced tamper-evident on the data before it is recorded**: if the data is tampered before recorded, we need to not only detect it but also determine when it's tampered. Therefore, knowing when the data is tampered can lead to different results in the forensic process.

IOTA, a distributed ledger whose data structure is based on Directed-Acyclic-Graph rather than traditional blockchain, has most of the features mentioned above. We can use IOTA as immutable data storage in our designed system. Moreover, IOTA has better scalability compare to traditional blockchain due to its parallel validation of consensus mechanism. Hence we choose IOTA as our data storage instead of other cryptocurrency such as Bit-coin [9] and Ethereum [4].

IOTA only provides a secure way to store data. Other requirements we stated above, including non-repudiation of recorded data and enhanced tamper-evident on the data before it is recorded, are not supported by IOTA. Therefore, we will draw support from cryptographic hash function and digital signature to meet these two requirements in our work.

In this work, we proposed a distributed traceability system in vehicular network based on IOTA. We adopt cryptographic hash function and digital signature to ensure our vehicular data is stored securely and can provide authenticated information to rebuild the accident scene. Also, our proposed method is able to handle huge amount of data in real-time. With this traceability system, each vehicle is provided a secure way to store every data it received during its journey. These stored data can be used in different purpose such as data analysis and traffic accident investigation.

The remainder of this paper is organized as follows. Section 2 presents the background and related work of this thesis. Section 3 describes the system architecture and define the problems and challenges we face in our work. The design and implementation detail will be elaborated in Section 4. We simulate our designed system and evaluate its traceability, data efficiency, and security in Section 5.

2 Background and Related Works

2.1 Background

We present the prerequisite knowledge of our work in this section. The cryptographic hash function and digital signature are used to tackle the challenges relative to security, which are enhanced tamper-evident and non-repudiation. On the other hand, IOTA and masked authenticated messaging are the core structure and protocol used in our designed system.

2.1.1 Cryptographic Hash Function

A cryptographic hash function is a trapdoor function which maps arbitrary-length input into fixed length output. The input of the hash function called *message* and the output of the hash function called *digest* or *checksum*. Cryptographic hash function can be used for integrity check by appending the digest to the message during transmission. The integrity check fails if the digest of received message is not equal to the received digest.

Suppose H is a secure hash function, it must have the following three main properties

1. Pre-image resistance: Given the output of hash function y, one cannot find an input x such that $H(x) = y$ within practical computation time
2. Second pre-image resistance: Given an input x, one cannot find another input x' such that $H(x) = H(x')$ where $x \neq x'$ within practical computation time
3. Collision resistance: One cannot find two non-identical inputs x and x' such that $H(x) = H(x')$ within practical computation time

These properties imply that a malicious adversary cannot replace or modify the message without changing its digest. Thus, if two messages have the same digest, we can assume that they are identical.

2.1.2 Digital Signature

Digital signature is a mathematical scheme for verifying the authenticity of digital messages or documents. A valid digital signature gives the receiver a very strong reason to believe that the message was created by a known sender (authenticity) and the message was not tampered (integrity).

It adopts asymmetric cryptography and hash function. Figure 1 shows the basic step for signing and verifying the signature. There are three steps for signing a message:

Step 1 : Applying hash function to the message to be signed and receiving digest of the message.

Step 2 : Using the private key to encrypt the message digest which is called signature.

Step 3 : Sending the message along with its signature to receiver.

When the receiver receives the message and its signature, one can verify it by following steps:

Step 1 : Computing the message digest of received message.

Step 2 : Using the public key of sender to decrypt the signature, the receiver will compute another digest.

Step 3 : Comparing these two digests computed in Step 1 and 2, verification fails if these two digests are not identical.

2.1.3 IOTA

IOTA is an open-source project created by German non-profit foundation located in Berlin, Germany, that aims to enable machine-to-machine (M2M) communication. It is a distributed ledger that the main data structure is a direct-acyclic-graph (DAG) called Tangle [12]. It can be used as an immutable data storage, which means that once the data is recorded and confirmed in IOTA they cannot be modified or deleted from the network.

Figure 2 shows the difference between IOTA Tangle and other block-chain technologies. The DAG structure in Tangle allows the transactions in IOTA be validated in parallel. Also, as claimed in IOTA foundation, each participant which wants to submit transaction has to participate in the consensus of the network by approving two past transactions on the network. This requirement makes IOTA zero transaction fee and removes miners from IOTA. Zero transaction fee and no miner in IOTA are the reasons that IOTA has better scalability compare to other block-chain technologies.

2.1.4 Masked Authenticated Messaging

Masked Authenticated Messaging (MAM) is a second layer data communication protocol which emits and accesses an encrypted data stream over the public ledger IOTA Tangle. Since every message that utilizes MAM protocol will be encrypted before attached to tangle. IOTA's consensus protocol adds integrity to these message streams. MAM uses merkle tree based signature scheme, which will be introduced later, to sign the digest of an encrypted message. Given these properties, MAM provides confidentiality, integrity, and privacy to the data stream.

MAM uses the message chain to store messages. Each message always points to the next message in the same message chain, similar to a single link list. Figure 3 shows the basic structure of MAM. `nextRoot` is the pointer points to the next message in this channel, and the other field will be explained in the following section. The whole MAM structure will be encrypted when one publishes the data to Tangle.

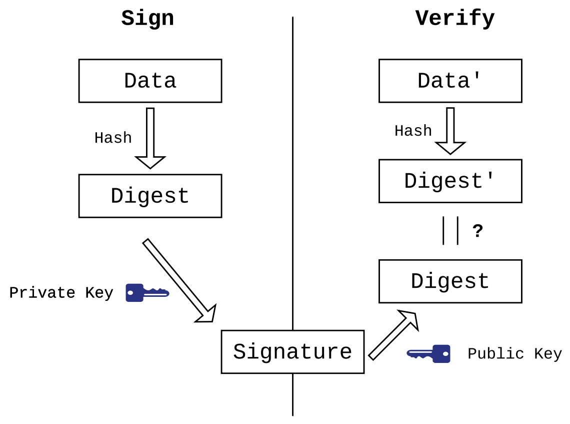


Figure 1: Sign and verify of digital signature

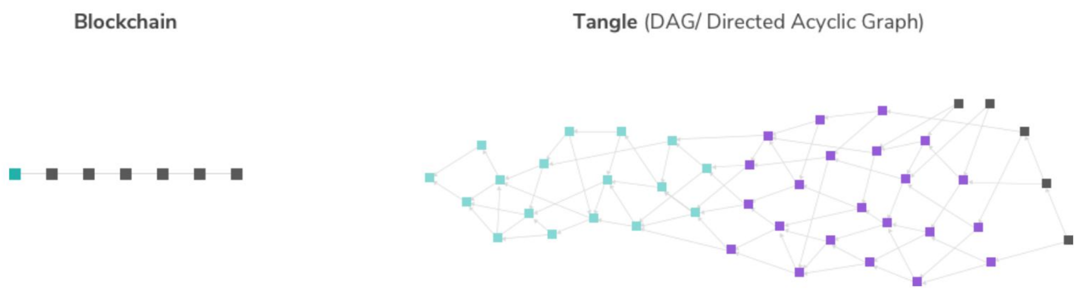


Figure 2: Data structure of Tangle and Block-chain

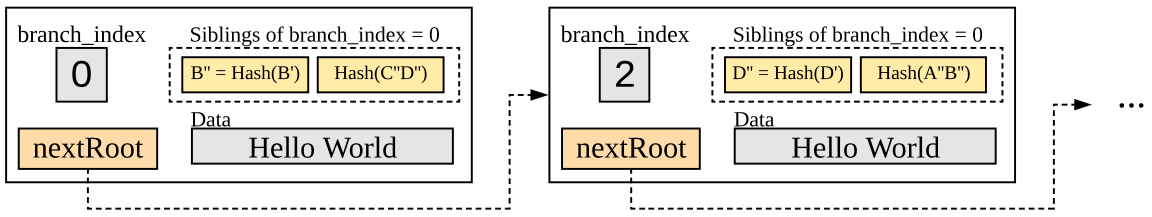


Figure 3: MAM structure

Merkle Tree Based Signature Scheme Merkle tree based signature scheme [3] is one of the hash based signature scheme which claims to be quantum resistance.

Figure 4 shows an example of merkle tree structure with four private keys, which are denoted as A, B, C, and D respectively. The private keys are generated by the index and seed, and are used to sign the message. The public keys are the hash of the private keys and are used to verify the signature. Merkle tree is a binary tree and its leaf nodes are the hash of the public keys. Each node except the leaf node in the merkle tree is the hash of the concatenation of its child nodes. For instance, the root is equal to Hash(Hash(A” || B”) || Hash(C’ || ’D”)) where Hash(A” || B”) and Hash(C” || D”) are the child nodes of root in Figure 4. Note that the symbol || stands for string concatenation.

When using this merkle tree based signature scheme to sign the message, the sender needs to choose a public-private key pair from the four key pairs shown in Figure 4. The chose private key is used to sign the message and the chose public key is used to verify the signature. The sender used the chose private key to sign the message and then send the message, the signature of the message, and the siblings of public key to the receiver. Siblings are the set of complementary hashes, by being combined with given public key, that can generate root of merkle tree. For example, the siblings of the public key A’ are B” and Hash(C” || D”) in Figure 4. One can compute A” by A’ then compute Hash(A” || B”) by A” and B”, and finally compute the root by Hash(A” || B”) and Hash(C” || D”). The reasons why the siblings of public key are needed is that the receiver can use the siblings to check the authenticity of sender’s public key.

2.2 Related Works

An event data recorder (EDR), also known as an automotive ”black box,” can be installed on the autonomous vehicles to record the information of the vehicle. Some EDRs will continuously record data until the vehicle crash, others are only activated only when crash-likes event (such as sudden change in wheel speed) occur and deactivated when that event is over [16].

Guo et al. [6] propose a block-chain-based event recording system for autonomous vehicles. The new mechanism called ”proof of event” is designed to achieve indisputable accident forensics by ensuring that event information is trust-able and verifiable. Figure 5 depicts an example scenario. The event record system is activated when the accident occurs, the ”accident” vehicle and the vehicles around it, which is called ”witness” vehicles, will form a ”community.” Every vehicle in the community will broadcast their event data with their own location and timestamps. All the broadcasted event data from both ”accident” and ”witness” vehicles will be verified and saved in new block, this process is called *proof-of-event* in their work.

In contrast with their event recording system which mainly focus on forensic purpose, our work

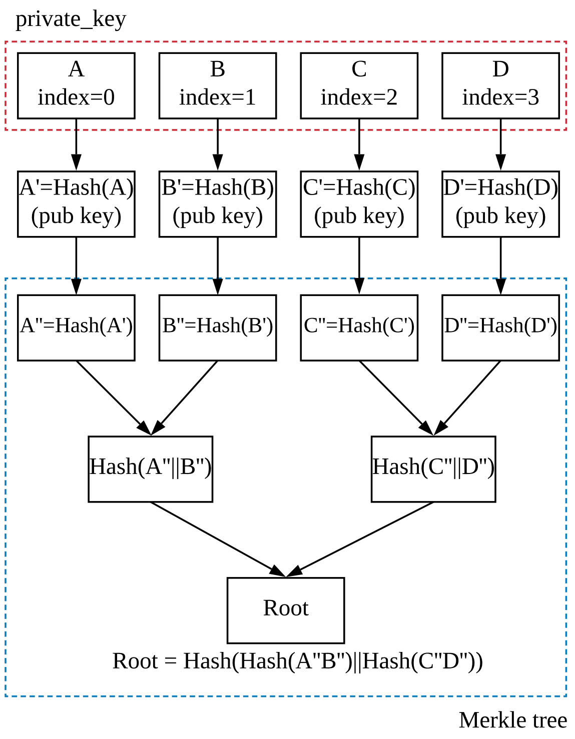


Figure 4: Merkle tree data structure

proposes a recording system such that the recorded data can be used in other scenario not just forensic process. This is because our designed system will record all the received data of a vehicle from its start-up to shutdown. For instance, the recorded data can be used for understanding the behaviour of autonomous vehicle, which will be helpful for tuning the car following model on autonomous vehicle.

3 System Architecture and Problem Definition

3.1 System Architecture

Our work focuses on data logging in vehicular network which requires real-time data storing and immutable data storage to ensure the recorded data is tamper-resistant. The distributed ledger IOTA is used in our work to play the role of immutable data storage. Also, we will utilize Masked Authenticated Messaging in IOTA to achieve traceability, confidentiality, and privacy of the recorded data.

Figure 6 shows the entire architecture of targeted use scenario. There will be multiple IOTA tangle networks in our designed system. Each Tangle network is held by the server which can be either on the cloud or on the local desktop computer. Each vehicle in VANET is a client in Tangle network and will publish the data it receives from other system in VANET as micro-transaction to Tangle. The vehicle will perform proof-of-work by itself if it wants to publish micro-transaction to tangle. Only authorized party, which denoted as traceability service in Figure 6, is able to fetch these recorded data from tangle for data analysis or forensic purpose. The vehicles can communicate with other system in VANET by DSRC[7], and every communication between Tangle network and other system rely on Masked Authenticated Messaging protocol in IOTA.

3.2 Problem Definition

The goal of this work is to design a distributed traceability system for exchanging data among autonomous vehicles. The recorded data in our designed system need to be stored securely in order to provide accurate information when it is needed. The major issue is to design and implement back-trace path, roaming, and secure data storing in order to reach the goal, which is defined as follow:

- **Back-trace Path**: the mechanism should support an efficient back trace path to the first data received by this autonomous vehicle after it starts up.
- **Roaming**: a vehicle might roam from one geographical region covered by one tangle network to another region. However, the message chains in different tangle networks are completely independent. mechanism should make the connection between two completely independent message chain when roaming occurs.
- **Secure Data Storing**: the mechanism should assure **non-repudiation** and **enhanced-tamper evident** on recorded data.

3.3 Challenges

We are using the IOTA Tangle network as our data storage, but publishing message to Tangle network require Proof-of-Work (PoW) which is a technique to deter denial of service attacks (DoS) and other service abuses such as spam on a network by requiring computation from the service requester. Therefore, publishing message to Tangle would experience certain delay due to the PoW mechanism in IOTA. In other words, this is very challenging for our designed system to achieve real-time.

Since we will deploy multiple Tangle networks in vehicular network (this will be described in Section 4), a vehicle might roam from one Tangle network to another Tangle networks. However, the message

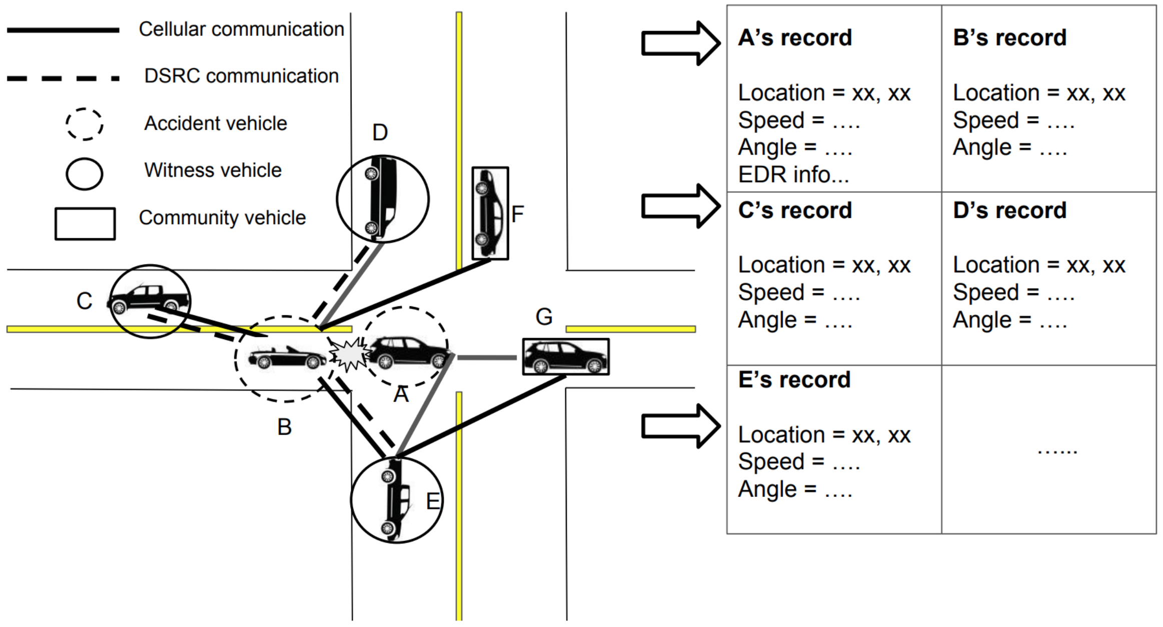


Figure 5: After an accident, both accident and witness vehicles generate and broadcast event data [6].

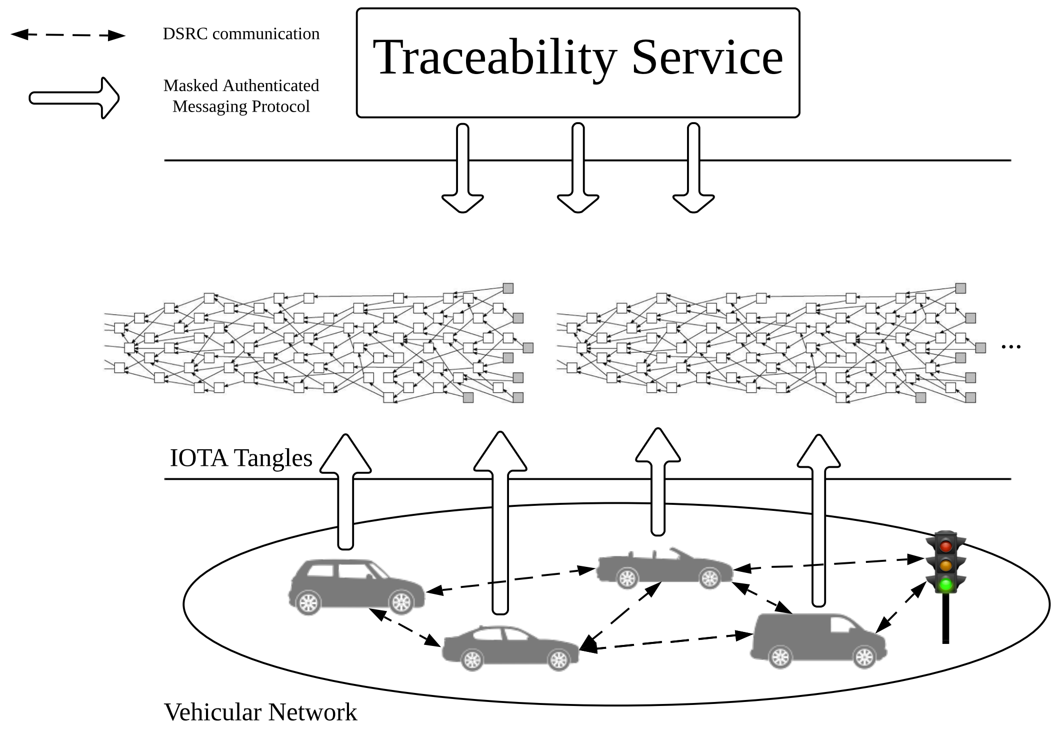


Figure 6: System architecture in our work

chains of MAM in the different Tangle network are completely independent. The challenge of handling roaming in Tangle network is to make a connection between two completely independent message chains.

Message chain with forward secrecy brings an issue for traceability system. Without the address of the first transaction in the message chain, one cannot trace back all the received data of a vehicle in the past. However, each vehicle will only has the latest message in the message chain in our designed system. Therefore, how to design a back trace path from the latest message to the first message in this message chain becomes a challenge due to the forward secrecy of MAM protocol.

4 Design and Implementation

This section presents the design and implementation of our vehicular data recording system based on IOTA MAM. We first present an overview of our system workflow, and give the assumption of our work. Section 4.2 describes how we deploy the tangle network in VANET. Section 4.3 presents the mechanism for roaming. Section 4.4 presents how to store the vehicular data securely in our work.

Figure 7 shows the entire workflow of our data recording system. Each vehicle will follow this workflow repeatedly from its start-up to shutdown. When an autonomous vehicle receives the data from other system in VANET, it will compute the keyed-hash digest of this data. The keyed-hash digest is the important basis for achieving enhanced tamper-evident on recorded data. After that, the message along with its digest will be put in the buffer. These are the blue blocks in Figure 7.

The first decision, denoted as *last message*, decides whether the vehicle is about to roam from one tangle network to another tangle network. If it is, we will generate head and link transaction for roaming, which are the red blocks in Figure 7. Otherwise, we will decide whether to attach the buffered data to tangle, which is the second decision block in Figure 7. This is because we will attach the data to tangle at certain frequency instead of attaching every data it received to tangle immediately.

If it is ready to attach the buffered data to tangle, the algorithm pack all the data stored in buffer into a single packed message. Then, the algorithm will attach this packed message to tangle. By packing the buffered data, we can shorten the latency of attaching to tangle. These are the green blocks in Figure 7. The detail of each block in Figure 7 will be described in the following sections.

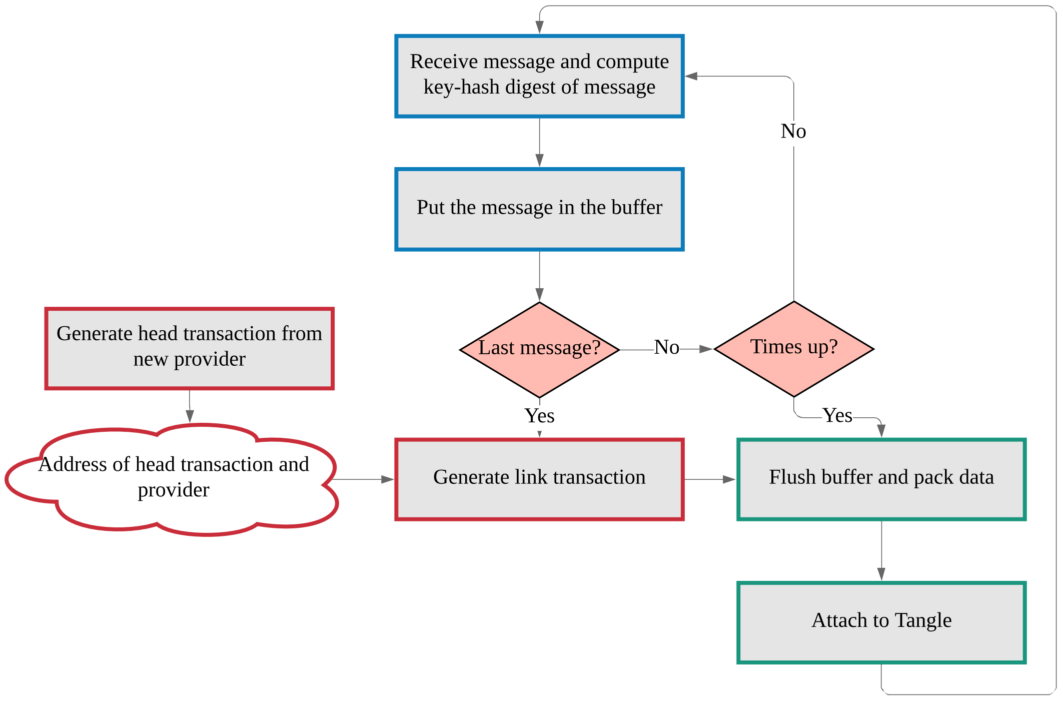

Receive message and compute key-hash digest of message
No
Put the message in the buffer
Generate head transaction from new provider
Last message?
No
Times up?
Yes
Yes
Address of head transaction and provider
Generate link transaction
Flush buffer and pack data
Attach to Tangle

Figure 7: Workflow of our designed system

There are two assumptions in our work. We assume public key infrastructure (PKI) is well-deployed in the vehicular network, which can securely manage and deploy certificate of each autonomous vehicle. We also assume the malicious vehicles have limited capacity to compromise more than thirty three percent of vehicles in vehicular networks, which will not let an attacker to launch the 34% attack [15]. Note that the 34% attack refers to a scenario in which an attacker is able to contribute over one third of the network's total processing power, allowing it to produce conflicting transactions

4.1 Deploying Network and Packing Data

IOTA Tangle plays the role of immutable data storage and each vehicle will attach all the data it received as micro-transaction to the tangle. However, the latency of attaching to tangle depends heavily on the geographic location of tangle network. By default, one can use the public network provided by IOTA foundation, but this will lead to significant delay of attaching to tangle. On the other hand, one can deploy the private tangle network to store data.

Figure 8 shows the latency of attaching messages to tangle network between using public and private tangle network with different payload sizes. The latency of attaching data to public and private tangle takes 4.5 and 0.5 seconds in average, which are tested under the network with the bandwidth of 160

Mbps. Using private tangle network can significantly shorten the latency and increase its scalability.

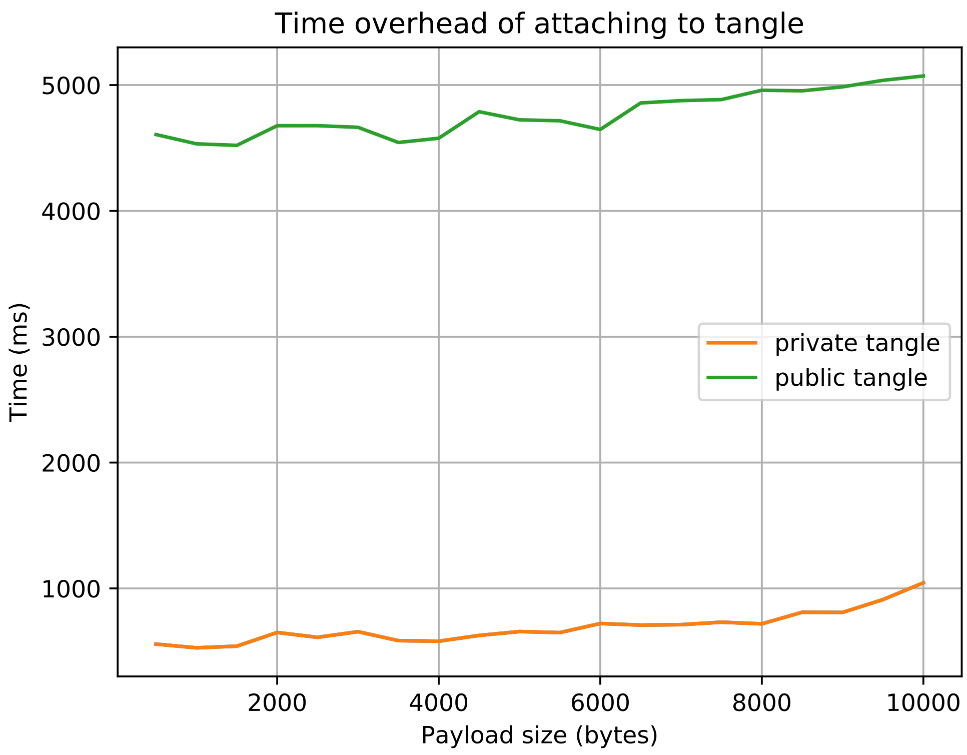


Figure 8: Latency between public and private tangle

Our system builds a distributed tangle network to store the messages in VANET. The distributed tangle network consists of multiple private tangle networks, each of which covers a non-overlapped geographical region. For example, one private tangle can be responsible for storing the data for the vehicles in one state or city in a country. Each vehicle in VANET will attach the received message to its associated private network. However, if a vehicle wants to attach every message it received to tangle immediately, 0.5 seconds is still too long. For example, the frequency of basic safety message (BSM) [7] in VANET is 10 Hz. Hence, even if each vehicle uses its nearest private tangle, it is still not able to attach every message it received to tangle in real-time.

We solve this issue by making each vehicle buffers the data it received and then packs these buffered data into a single packed message. The vehicle will attach this packed message to tangle network. This procedure will repeat every 2 seconds. The reason of packing data is that from Figure 8 we know that the payload size has negligible impact to the latency of attaching a message to tangle too much. With this method, the autonomous vehicle in our traceability system can now handle up to 10KB of data every second even if the latency of attaching to tangle is about 0.5 seconds.

Distributed private tangle networks can shorten the latency for attaching message to the network. However, the tangle networks are independent. When the vehicle moves one geographical region to another one, the messages will be stored on independent networks and lost their traceability. We will present the mechanism to maintain the traceability during roaming.

4.2 Traceability During Roaming

The messages published by one vehicle during roaming will be stored on different private tangle networks. However, the message chains of MAM in the different tangle networks are completely independent and do not have the mechanism to link to different networks. The proposed mechanism creates micro-transaction during roaming to link the message chains on different networks.

Suppose a vehicle is now roaming from tangle network N_A to tangle network N_B. We will handle the roaming issue by creating a **link transaction** in N_A and a **head transaction** in N_B, which will be explained in the following subsections.

4.2.1 Head Transaction

A head transaction is used to indicate the start of the message chain in MAM and provide a back-trace path to the message chain in another network. The data structure of head transaction is shown below. `provider` field is the address of previous network N_A. `head` field is a pointer that points to the head transaction of another network, which provides a back trace path. When a vehicle switches to a new tangle network, the head transaction will be created immediately. After head transaction is created, this vehicle can start to attach the data it receives to this new tangle network.

```
struct {
  char* provider; // The address of last network
  char* head;     // The address of head transaction
                  // in previous network
} headTransaction;
```

4.2.2 Link Transaction

A link transaction is used to connect two message chains of MAM in different tangle networks. The data structure of link transaction is shown below. When a vehicle is ready to roam from N_A to N_B, link transaction will be created to connect these two independent message chains of MAM in N_A and N_B. `provider` field is the address of the new provider N_B. `TX` field is a pointer that points to the head transaction in N_B which is the essential part in this transaction to maintain the traceability when roaming.

```
struct {
  char* provider; // The address of new network
  char* TX;       // The address of head transaction
                  // in previous network
} linkTransaction;
```

The process of roaming is explained as follows. The vehicle first generates head transaction HT in N_B and then records the address of HT and N_B. Now, it can generate link transaction LT in N_A and then insert the address of N_B and HT in the content of LT. The pseudo code of roaming process is shown in Algorithm 1.

Figure 9 shows an example of message chain of MAM on the tangle after roaming. Head transaction and link transaction play important roles in roaming process. They connect two independent message chains and maintain the traceability of vehicular data when roaming.

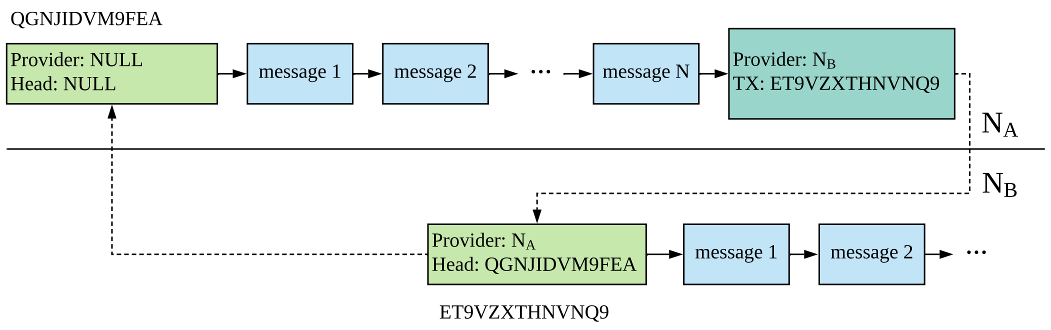


Figure 9: An example of message chain after roaming

Algorithm 1 Roaming Transaction

```
Input: Address of old provider $P_{old}$, new provider $P_{new}$, the state of message chain $S$, and address of
    head tx in old network $A$
Output: Update the state of message chain $S$ and link two message chain in different network
 1: $L \leftarrow \{\}$
 2: $H \leftarrow \{\}$
 3: $H$[provider] $\leftarrow P_{old}$
 4: $H$[tx] $\leftarrow A$
 5: $S_{new} \leftarrow$ init($P_{new}$)    // initialize the state of new provider
 6: $D \leftarrow$ attachToTangle(JSON.stringify(H))    // attach head tx to new network
 7: $L$[provider] $\leftarrow P_{new}$
 8: $L$[tx] $\leftarrow D.root$
 9: setProvider($P_{old}$)    // use old provider
10: attachToTangle(JSON.stringify(L))    // attach link tx to old network
11: setProvider($P_{new}$)
12: $S \leftarrow S_{new}$
```

4.3 Secure Data Storing

For secure data storing, we need to provide **non-repudiation** and **enhanced-tamper evident** on recorded data to ensure the data stored on the network is authenticated. Non-repudiation refers to that the sender of each recorded data cannot dispute its authorship. This property is needed since the sender of the malicious data may try to dispute its authorship to avoid liability. On the other hand, enhanced-tamper evident implies that if the recorded data is tampered, the system needs to detect it and determines when it is tampered. This property is needed because the data in VANET before it is recorded could be tampered at two phases:

- **Transmission Phase**: Since the data is transmitted in an insecure communication channel, the data could be tampered by any malicious party during transmission.

- **Receiving Phase**: The receiver might be a malicious vehicle and tried to tamper the data it received. After the data is tampered, this malicious vehicle can attach this tampered data on the Tangle and try to avoid liability.

Therefore, knowing when the data is tampered can lead to different results in the forensic process.

We describe how we provide enhanced tamper-evident on vehicular data before it is recorded and non-repudiation on recorded data by using cryptographic function in this section. The purpose of doing this is to make sure the recorded data can provide useful and accurate information when it is used in the forensic process and data analysis.

4.3.1 Non-repudiation

We will apply digital signature scheme to provide non-repudiation on recorded data. We employ elliptic curve cryptography [8] in our digital signature scheme. Elliptic curve cryptography requires smaller key size to provide the same level of security afforded by RSA [13], which is shown in Table 1. We choose SHA-224 [11] as our hash function in our digital signature scheme, which is the truncated variants of SHA-256. Gueron et al. [5] shows that using truncated SHA-256 variant like SHA-224 gives a significant performance advantage over SHA-256 on 64-bit platforms, which is due to the doubled input block size. The method we designed to achieve non-repudiation on recorded data is described in the following.

RSA key size (bits)	Elliptic Curve key size (bits)
1024	160
2048	224
3072	256
7680	384
15360	521

Table 1: Key size different with same security level between RSA and ECC

Each vehicle needs to maintain a table that records all the public key it had before. The reason why this table is needed will be explained later. The sender of each message needs to sign the message and then sends the message, the signature of the message, and the key ID to the receiver. The key ID is used to tell the verifier which public key it should use to verify this signature, since each vehicle might have a different public-private key pair from the certificate authority. Therefore, the message chain of receiver on the tangle might contain messages that are signed by the different private key of the sender. Without the public key table mentioned above and the key ID in sender payload, the verifier will not be able to verify the signature.

4.3.2 Enhanced Tamper-Evident

We present the solution of providing enhanced tamper-evident on the data before it is recorded in this section. The receiver extracts the message part right after it receives the payload from sender (the other parts of the sender payload are signature and key ID, which are mentioned in the last section). After that, it will compute the keyed-hash digest of this message and then add this digest to the payload it received.

Figure 10 is an example of how keyed-hash digest is generated. Note that the symbol || stands for string concatenation. The digest is generated and added to the payload before the message is sent to the application layer. The key used in the keyed-hash function is the secret value which is not known by any application. After the digest is added to payload, this new payload will be attached to tangle and sent to the application layer.

We are using BLAKE2 [2] as our hash function to generate keyed-hash digest, which is as fast as SHA-1 and as secure as SHA-3. Table 2 shows the time overhead of generating digest between BLAKE2 and SHA family. The biggest time overhead relative to BLAKE2 is 194.32 %, which is almost 3 times slower than BLAKE2. Even with SHA-1 which is the fastest hash function in SHA family still exists 41.23 % difference between BLAKE2. Another reason to choose BLAKE2 over the well-known hash function SHA-1 is that the desire hash function here needs to be **fast** and **secure**. SHA-1 is fast, but not secure anymore since the collision of SHA-1 has already been found [14].

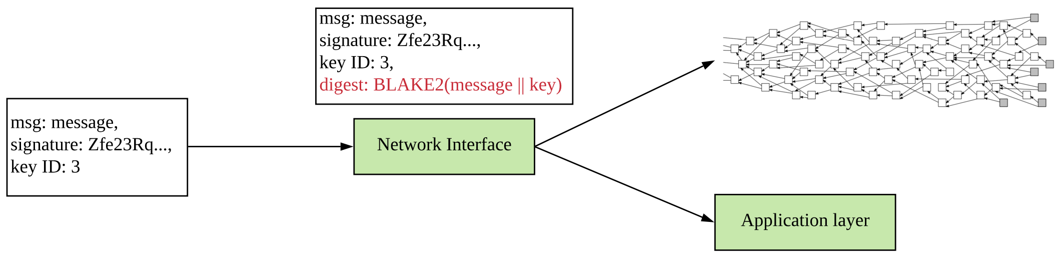


Figure 10: An illustration of computing keyed-hash digest

Hash Algorithm	Time Overhead (ms)	Time Overhead Relative to BLAKE2 (%)
SHA-1	0.00572	41.23
SHA-2	0.00978	141.48
SHA-3	0.01192	194.32
BLAKE2	0.00405	-

Table 2: Comparison between SHA family and BLAKE2

Keyed-hash digest can determine whether the message is ever tampered. Since the application does not know the key used in the keyed-hash function, it cannot tamper the message and regenerate the correct keyed-hash digest. With the keyed-hash digest and the digital signature mentioned above, the requirements of enhanced tamper-evident is met.

Figure 11 shows the process of determining when the message is tampered. Signature verification determines if the data fetched from the tangle network is correct. If the signature verification fails, the algorithm can use `keyed-hash` field in the fetched data to determine when this message is tampered. If the regenerated key-hash digest is different from the `keyed-hash` field in the fetched data, it implies that this fetched data is tampered by the application of the receiver. Otherwise, this fetched data is tampered during the transmission.

4.4 Summary

Figure 12 shows the data flow of our designed system which contains all the components we described in this chapter. Figure 13 shows an example of the message chain on the tangle network. Each vehicle will pack the data it received into a packed message and attach this packed message to tangle every 2 seconds. Each message chain on tangle network is composed of all the data which is received by a single vehicle. Traceability is provided if we have any address of the transaction in this message chain.

5 Performance Evaluation

We present the simulation of our designed traceability system and evaluate its scalability, traceability, data efficiency, and security in this chapter.

5.1 Simulation Environment

The experiment environment simulates the VANET, which consists of vehicles and multiple private tangle networks.

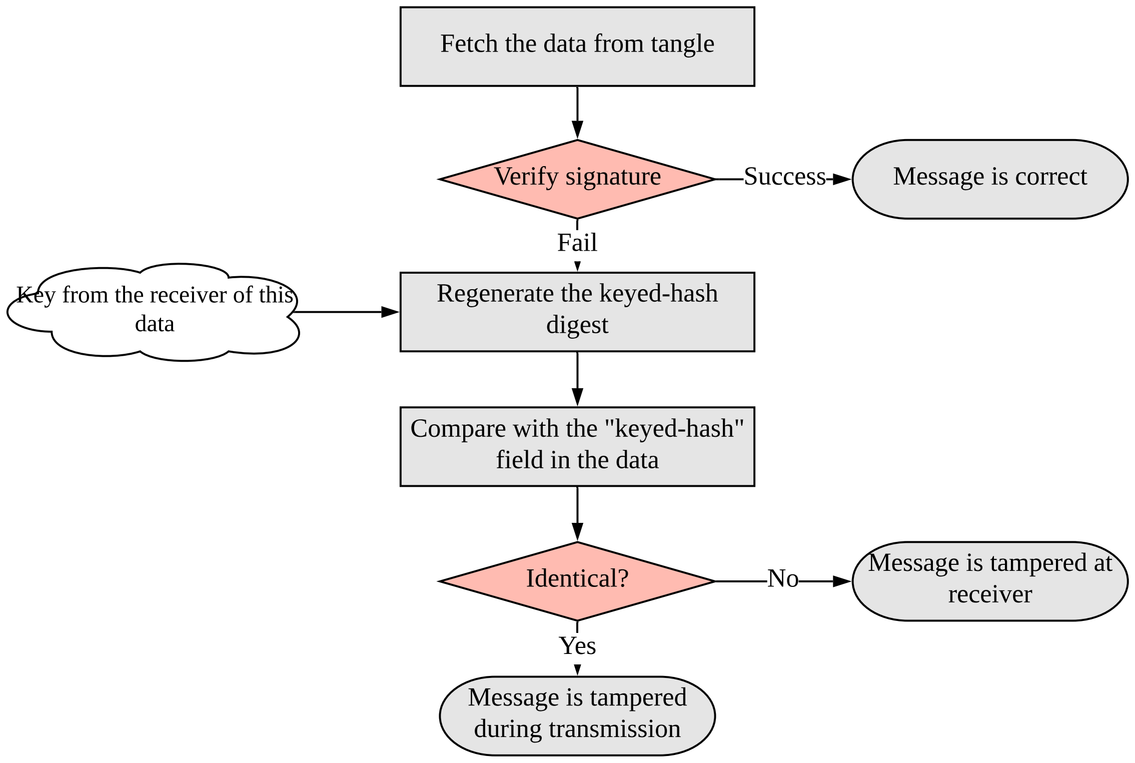

Fetch the data from tangle
Verify signature
Success
Message is correct
Fail
Key from the receiver of this data
Regenerate the keyed-hash digest
Compare with the "keyed-hash" field in the data
Identical?
No
Message is tampered at receiver
Yes
Message is tampered during transmission

Figure 11: Process for detecting when the message is tampered

The on-board computing device is represented by a nVidia Jetsen TX2 [10]. Two desktop computers represent the computing devices to store private tangle networks and clients of tangle networks. The hardware specifications of these three devices are shown in Table 3. TX2 represents the autonomous vehicle in our simulation, and both PCs will run an IOTA tangle network respectively.

Device	**Hardware**
Jetsen TX2	Quad-Core ARM Cortex-A57
PC1	Intel Core i9-9820X
PC2	Intel Core i7-6700K

Table 3: Hardware of devices used in simulation

We first set the frequency of attaching to tangle at 1 Hz and simulate 7 scenarios. In each scenario, we generate different amounts of data per second to TX2, which ranges from 4KB to 10KB. Then, it will pack these data and attach to the private tangle networks. Each simulation will last for 5 minutes. We will observe the CPU and memory utilization on TX2 to see if it can handle large volume of data in real-time. Next, we will set the frequency of attaching to tangle to 0.5 Hz and simulate the same scenarios mentioned above to observe the utilization of CPU and memory.

5.2 Simulation Results

Attaching to tangle at 1 Hz Figure 14 shows the CPU and memory utilization on TX2 when attaching to tangle at 1 Hz. The memory utilization in TX2 increase slowly in this setup no matter how much data it had received. The cause of this phenomenon is that the system cannot process all the data stored in the buffer in real-time due to the latency of attaching to tangle. This implies that with attaching to tangle every 1 second, the system is not able to process huge amount of data in real-time

Attaching to tangle at 0.5 Hz Figure 15 shows the CPU and memory utilization on TX2 with the frequency of attaching to tangle set to 0.5 Hz. With the lower frequency of attaching to tangle, the memory utilization in TX2 stays in a stable state. This implies that with this setup, our traceability system is able to process huge amount of data in real-time.

5.3 Scalability Evaluation

We will evaluate the scalability of our designed system by observing the latency of attaching to tangle with different transaction rate. If Tangle network does have better scalability as claimed by IOTA, the latency of attaching to tangle should decrease when the transaction rate increase.

We use PC1 to store the tangle network, and PC2 simulates large number of clients to access the tangle network run in PC1. The hardware specification of both PC1 and PC2 are shown in Table 3. The client will generate different amount of transactions at every second, which will range from 1 to 100 transactions per second. We will observe the latency of attaching message to tangle in client to evaluate the scalability of tangle network run in PC1.

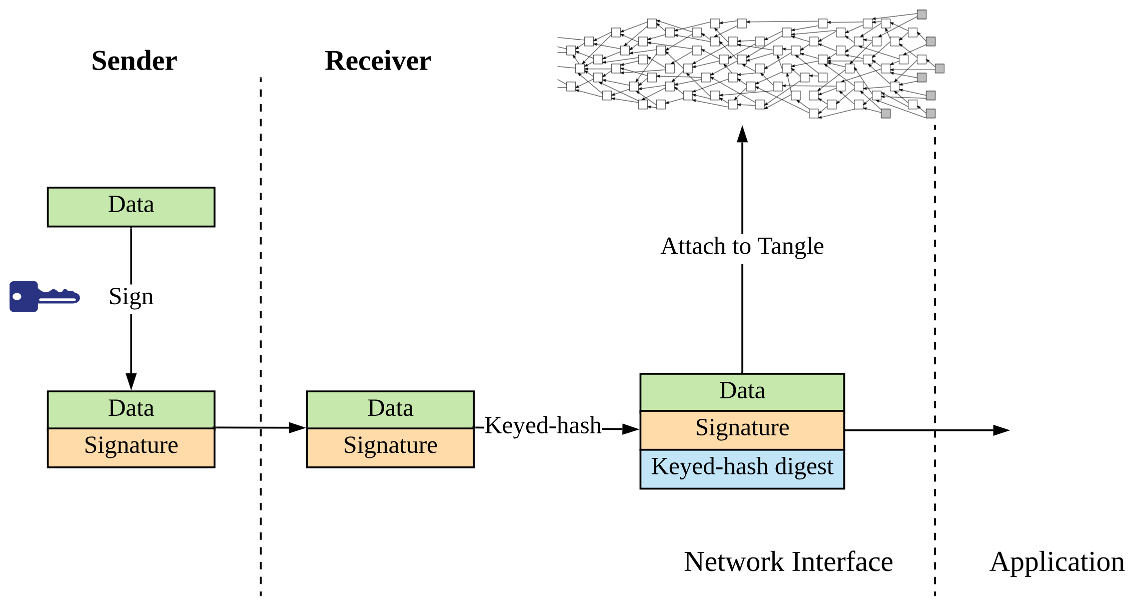


Figure 12: Data flow of our designed system

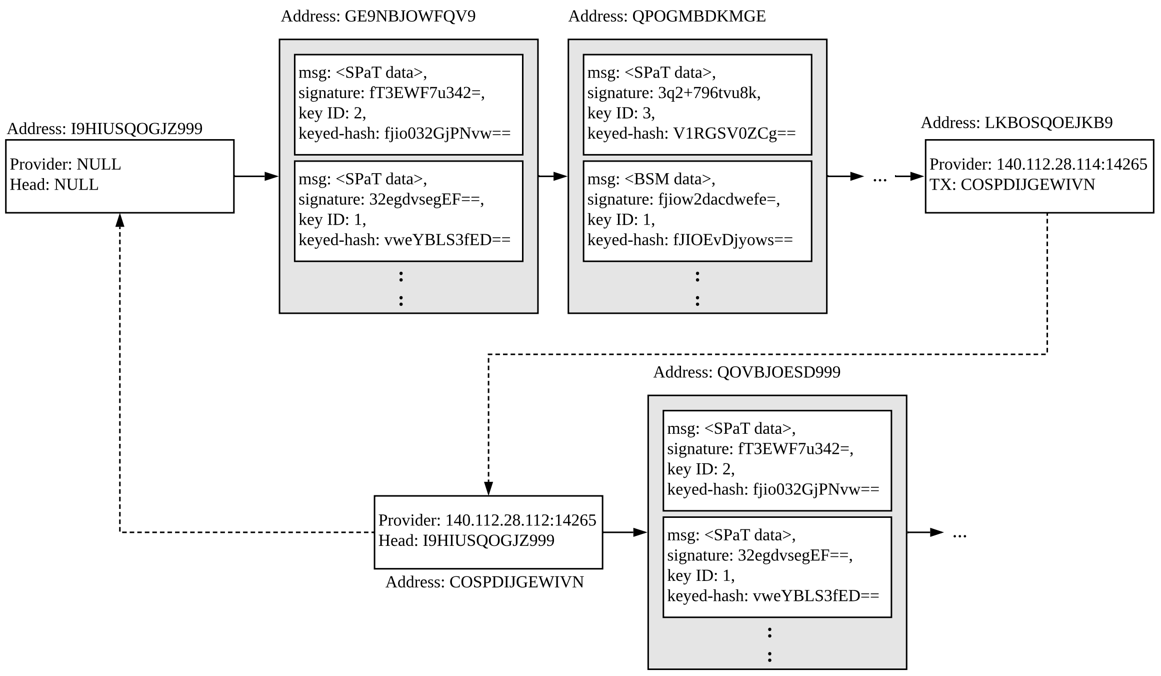


Figure 13: An illustration of message chain on tangle network

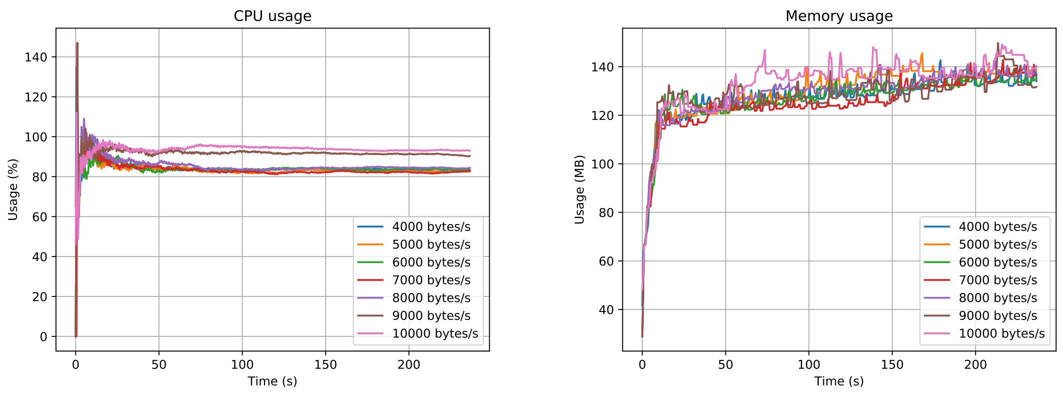


Figure 14: Simulation result when attach to tangle is set to 1 Hz

Figure 16 shows the result of our evaluation. The result indicates that the latency of attaching to tangle will slightly decrease when the transaction rate increase. By observing the trend of curve in Figure 16, the capacity of the tangle network will not decrease if the transaction rate keep increasing. This imply that our designed system has a good scalability, which is an important requirement if our system needs to be deployed in VANET.

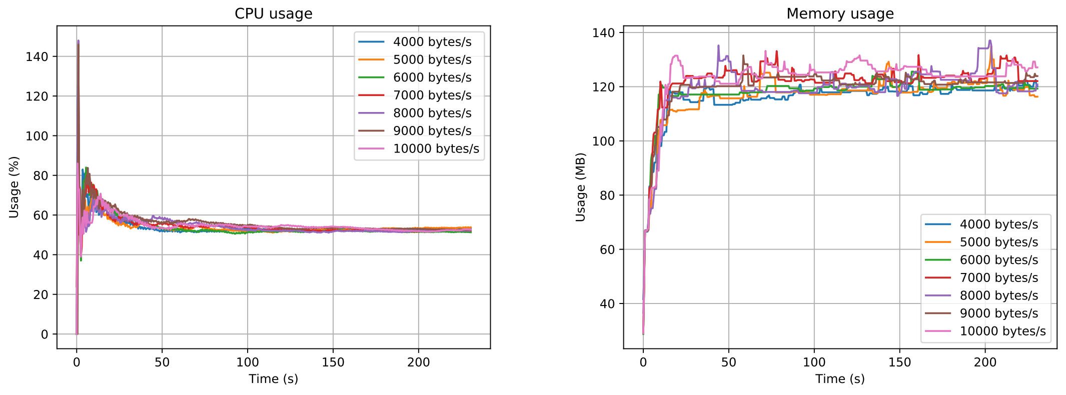


Figure 15: Simulation result when attach to tangle is set to 0.5 Hz

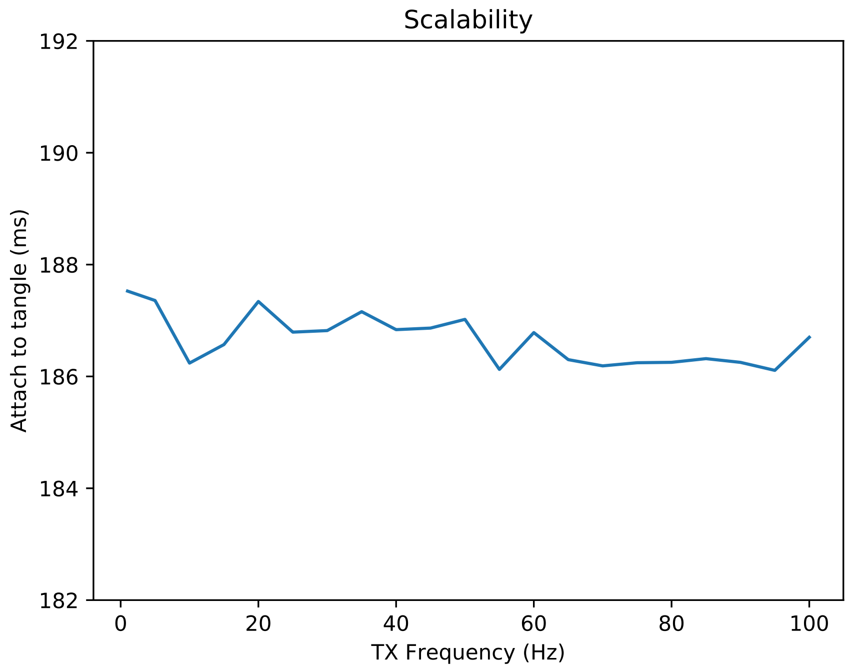


Figure 16: Scalability of our designed system

5.4 Data Efficiency

Before we discuss the data efficiency of each transaction in our work, we need to introduce the structure of MAM bundle first. Bundle of MAM has roughly two sections, which are **signature section** and **MAM section**. The data of each section is stored in the field called `signatureMessageFragment` (`sMF`). The encrypted MAM structure mentioned in Section 2.1.4 will be stored in `sMF` field of MAM section. The signature of the encrypted MAM structure will be stored in `sMF` field of signature section. Each `sMF` field has the maximum size of 2187 trytes which is about 1300 bytes. Message with size greater than 2187 trytes will be carried by multiple MAM sections.

In our proposed system, every vehicular data will be added with some meta-data such as `signature`, `key ID`, and `keyed-hash`. The vehicular data along with the meta-data will be called processed data in the following section. Multiple processed data will be packed and attach to tangle. Now, `sMF` field of MAM section is composed of processed data, `index`, `siblings`, and `nextRoot`. We now compute the data efficiency of `sMF` field of MAM section.

Table 4 shows every field that will appear in `sMF` field of MAM section and its size. The size of field `msg` *m* depends on the size of vehicular data we want to record.

Field	**Size (bytes)**
`msg`	*m*
`signature`	96
`key ID`	4
`keyed-hash`	88
`index`	4
`siblings`	49
`nextRoot`	49

Table 4: Field and size contain in `sMF` of MAM section

The size of meta-data in `sMF` field of MAM section is $96+4+88+4+49+49 = 290$, then the data efficiency of `sMF` is

$$\frac{1300-290}{1300} = 77.69\%$$

6 Summary

In this work, we designed a real-time and secure distributed traceability system which can be deployed in vehicular networks. The system is based on the IOTA and utilize its communication protocol MAM to store the vehicular data. In order to make our system achieve real-time, we solve the issue of huge latency of attaching to tangle by deploying our own private tangle network in VANET. We also solve the issue of losing traceability when roaming happen by creating link transaction in old network and head transaction in the new network. Furthermore, we add digital signature and keyed-hash function to ensure our vehicular data stored on the tangle can provide accurate and useful information when one needed.

Acknowledgment

This research was supported in part by the Ministry of Science and Technology of Taiwan (MOST 106-2218-E-009 -030), and MediaTech Inc (Contract Number: MTKC-2018-0167).

References

[1] R. Accorsi, “Log data as digital evidence: What secure logging protocols have to offer?” in *Proc. of the 33rd Annual IEEE International Computer Software and Applications Conference (COMSPAC’09), Seattle, Washington, USA*. IEEE, July 2009, pp. 398–403.

[2] J.-P. Aumasson, S. Neves, Z. Wilcox-O’Hearn, and C. Winnerlein, “Blake2: Simpler, smaller, fast as md5,” in *Proc. of the 11th International Conference on Applied Cryptography and Network Security (ACNS’13), Banff, Alberta, Canada*, ser. Lecture Notes in Computer Science, vol. 7954. Springer, Berlin, Heidelberg, 2013, pp. 119–135.

[3] G. Becker, “Merkle signature schemes, merkle trees and their cryptanalysis,” https://www.emsec.ruhr-uni-bochum.de/media/crypto/attachments/files/2011/04/becker_1.pdf [Online; accessed on December 15, 2019], 2011.

[4] C. Chinchilla, “A next-generation smart contract and decentralized application platform,” https://github.com/ethereum/wiki/wiki/White-Paper [Online; accessed on December 15, 2019], 2019.

[5] S. Gueron, S. Johnson, and J. Walker, “Sha-512/256,” in *Proc. of the 8th International Conference on Information Technology : New Generations (ITNG’11), Las Vegas, Nevada, USA*. CPS, April 2011, pp. 354–358.

[6] H. Guo, E. Meamari, and C.-C. Shen, “Blockchain-inspired event recording system for autonomous vehicles,” in *Proc. of the 1st IEEE International Conference on Hot Information-Centric Networking (HotICN’18), Shenzhen, China*. IEEE, August 2018, pp. 218–222.

[7] J. Kenney, “Dedicated short-range communications (dsrc) standards in the united states,” *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162–1182, August 2011.

[8] N. Koblitz, “Elliptic curve cryptosystems,” *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, January 1987.

[9] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” https://bitcoin.org/bitcoin.pdf [Online; accessed on December 15, 2019], 2008.

[10] nVidia Inc., “nVidia jetson TX2 module,” https://developer.nvidia.com/embedded/jetson-tx2 [Online; accessed on December 15, 2019], 2019.

[11] N. I. of Standards and Technology, “Secure hash standard,” https://csrc.nist.gov/publications/detail/fips/180/2/archive/2002-08-01 [Online; accessed on December 15, 2019], 2002.

[12] S. Popov, “The tangle,” https://iota.org/IOTA_Whitepaper.pdf [Online; accessed on December 15, 2019], 2018.

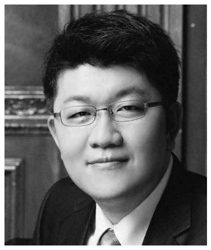
[13] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, February 1978.

[14] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov, “The first collision for full sha-1,” in *Proc. of the 37th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO’17), Santa Barbara, California, USA*, ser. Lecture Notes in Computer Science, vol. 10401. Springer, Cham, 2017, pp. 570–596.

[15] S. Vogt, “Is a double spending attack possible with IOTA?” http://www.tangleblog.com/2017/07/10/is-double-spending-possible-with-iota/ [Online; accessed on December 15, 2019], 2017.

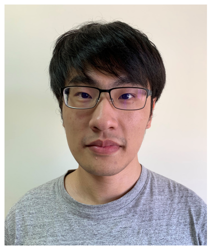
[16] WikiPedia, “Event data recorder,” https://en.wikipedia.org/wiki/Event_data_recorder [Online; accessed on December 15, 2019], 2019.

Author Biography



Chi-Sheng Shih Dr. Chi-Sheng Shih has joined the Department of Computer Science and Information Engineering at National Taiwan University in Feb. 2004. He currently serves as an Professor and Director of Graduate Institute of Networking and Multimedia. He received the B.S. in Engineering Science and M.S. in Computer Science from National Cheng Kung University in 1993 and 1995, respectively. In 2003, he received his Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign. His main research interests include embedded systems, hardware/software codesign, real-time systems, and database systems. Specifically, his main research interests focus on real-time operating systems, real-time scheduling theory, embedded software, and software/hardware co-design for system-on-a-chip. His research results won several awards including the Best Paper Award, 2011 ACM Research in Applied Computation Symposium (RACS 2011), the Best Paper Award, IEEE

RTCSA 2005, the Best Student Paper Award, IEEE RTSS 2004, and Best Paper Award, IEEE International Symposium of Service and Cloud Computing (SC2), 2016. He also serve the steering committee for several international conferences including IEEE Cyber-Physical Systems, Network and Applications (CPSNA) and editoral board of international journals such as Journal of Service-Oriented-Computing Architectures.



Wei-Yu Hsieh He received B.S (2017) in Mathematic and M.S (2019) in Computer Science from National Taiwan University. His main field of interest is computer security.

Chia-Lung Kao He received his M.S. degree in Electronics Engineering from National Taiwan University in 2013. After receiving his master degree, he joined TSMC as DBA. His research interests include multicore computing.