BIFF: A Blockchain-based IoT Forensics Framework with Identity Privacy

Duc-Phong Le, Huasong Meng, Le Su, Sze Ling Yeo, Vrizlynn Thing
*Cyber Security & Intelligence, Institute for Infocomm Research (I^2R), A*STAR*
{ledp, menghs, lsu, slyeo, vriz}@i2r.a-star.edu.sg

***Abstract*—The ubiquitous deployment of Internet of Things (IoT) devices enhances connectivity and communication, and benefits almost every aspect of our lives from manufacturing to retail to smart homes. However, low levels of security protection in these devices due to their limited resources open opportunities for malicious users. An IoT forensics system collecting, processing, analyzing and reporting evidence of attack is required to mitigate the IoT security issues. Although such system has been studied over the past decade and solutions such as cloud-based IoT forensic were proposed, limitation still exist. In this paper, leveraging on the blockchain technology, we propose a permissioned blockchain-based IoT forensics framework to enhance the integrity, authenticity and non-repudiation properties for the collected evidence. We formally define the system architecture, provide framework details, and propose a cryptographic-based approach to mitigate identity privacy concern.**

***Index Terms*—IoT forensics, digital evidence, blockchain, identity privacy**

I. Introduction

With the rapid growth of different smart and lightweight devices, the Internet of Things (IoT) industry is booming, resulting in IoT devices permeating many aspects of our daily lives. Examples include devices in transport, smart home appliances and wearable devices to monitor our health or whereabouts. While this widespread use of IoT devices brings more convenience to our daily lives, their heterogeneous nature may be exploited by cyber criminals to launch malicious attacks. As such, it is important to have a common framework to obtain digital evidence from IoT devices to conduct investigations in the event of a cyber crime.

IoT forensics refers to the process of collecting, analyzing, storing and presenting digital evidence within IoT devices in a legally binding manner. In particular, the traceability, integrity and provenance of the evidence should be maintained. However, resource-constraint IoT devices may find it challenging to achieve these goals. For instance, the lack of memory may cause the data to be frequently altered or overwritten. In addition, some devices may only be connected locally and lack the capability to transfer the evidence to the investigator promptly. Finally, while devices such as computers or servers can be confiscated by the law enforcement, it may not be so straightforward to identify all the relevant IoT devices centered around a particular case for investigations.

One possible approach is to design a digital chain of custody via a group of trusted IoT devices, which allows the system to link the evidence and form a systematical flow of the events. While the digital witness (DW)-based chain of custody framework proposed in [1] allows IoT devices with constrained resource to systematically collect and transfer digital evidence to the investigator, it requires traceability and non-repudiability of the evidence being transferred to guard against malicious behaviors. The DW framework achieves these properties by ensuring that all transfers are transparent and delegating devices embedded with trust and integrity capabilities to perform the transfers. However, such transparency leads to privacy concerns for the devices involved in the chain.

In this paper, leverage on the emerging blockchain technology, we propose a new IoT forensic framework to record all the events in the life cycle of digital evidence instead of relying on the trust assumption on the devices, to ensure its integrity and traceability. Briefly speaking, blockchain, the technology underpinning Bitcoin [2], is a *trusted* and *distributed* ledger, in which users are able to record information, prove and transfer their ownership from one to another without a trusted third party intermediary. All these transactions are *fully traceable*, supervised by parties on the blockchain system and no single party has control over the data. While the widely connected IoT devices naturally forms a distributed network and the trust among the devices could typically be minimum, adopting the traditional blockchain system into an IoT forensic scenario is not straightforward. Challenges arises from the translation and re-design the intuitive financial transaction-based system into an event-based framework to allow the evidence to be transferred and recorded in a secure and authenticated way.

In this work, we propose a **b**lockchain-based **I**oT **f**orensics **f**ramework, or BIFF, to record the events of the entire life cycle of digital evidence in a transparent, traceable and identity privacy-preserving way. In particular, by deploying smart contracts, our framework provides an automated way for IoT devices to record a chain of custody for digital evidence during a cyber attack. In addition, our design accounts for different scenarios, including collaboration among the devices to report an attack. Further, by taking advantages of one-time signatures, our framework seeks to provide a practical method for both transparency and privacy of the devices simultaneously.

II. Related work

A. IoT forensics

Digital forensics is defined as a legally acceptable procedure to collect, examine, analyze and finally report the digital evidence [3]. The ubiquitously deployed IoT devices, although constantly been the target for attack, also helps to facilitate the digital forensics procedure, especially during evidence collection period by providing a rich set of personal and environmental data sources [4] and the high connectivity. However, as IoT devices are mostly configured with limited computational power and storage, the collected evidence need to be stored elsewhere apart from IoT devices for later analysis and reporting in an integrity-protected and authenticated way.

One approach for IoT forensics is to transfer the acquired digital evidence to a cloud platform [5]. However, this approach comes with a series of challenges [6]. As cloud computing systems usually span internationally, any cross border evidence transmission takes place within the IoT forensic procedure may result in jurisdictional difficulties [7]. The integrity of digital evidence could be questioned if the cloud computing system fails to provide adequate authentication and encryption schemes along with the IoT forensics [8]. Further, the traceability of digital evidence is another challenge that a cloud storage solution may not effectively offer.

Another group of approaches named as digital witness was first proposed in [1] and later enhanced in [9], [10]. Such a framework creates a digital chain of custody among the participants, which include but are not limited to devices, citizens and authorized custodians of evidence. It defines a number of security features and protocols to regulate the evidence transfer and protect the integrity of an evidence, and thereby overcomes the main challenge for IoT forensics of preserving evidence as mentioned above. With their own merit, the framework still suffers from few limitations. First, the framework could not detect should a participant behaves dishonestly, that is to properly transfer the evidence to the next participant. Second, as pointed out in [9], [10], there is a chance that the privacy of digital witnesses may be compromised under attack. Nieto et al. in [9] suggested few solutions to mitigate the above issues, however, they require a high computational power, and are not suitable for IoT devices.

B. Blockchain

Blockchain concept was introduced by Satoshi Nakamoto in [2] along with the creation of Bitcoin cryptocurrency. Over the last decade, this disruptive technology has swept across almost every industry and seen wide applications. Apart from the financial sector, other explored areas include data storage [11], sharing service [12], agriculture [13], and many more.

Blockchains could be categorized into *permissionless* (or *public*), or *permissioned* (sometimes called *private* or *consortium* blockchain) [14]. Bitcoin and cryptocurrencies such as Ether [15] are permissionless systems which are open to public to join the network and maintain the transactions. A permissioned blockchain, as opposed to the other variant, allows organizations to retain control over access rights of the system, while still enjoying the blockchain benefit such as record integrity, authenticity and non-repudiation. The system could also allow customized consensus protocols to improve the overall efficiency. Depending on the access control policy, permissioned blockchain typically only allows pre-determined participants to have permission to execute the consensus protocol and update the distributed ledger, while others could only submit transactions and view the history. The Hyperledger Fabric [16] is one such example.

Smart contracts, also called self-executing or digital contracts, was first introduced in [17] as a mean to embed contractual clauses into digital assets. This concept was formalized by the Ethereum project [15], whereby the smart contract is written as code, stored and replicated on the blockchain system, public and witnessed by all participants in the network, executes itself according to code terms once a trigger event such as time or strike price is hit.

III. Framework

In this section, we first define the roles and access rights in our system, followed by describing design specifics for the transaction, smart contracts, blocks and consensus.

A. Roles and Rights

For our blockchain-based digital chain-of-custody system, we first define the following entities in the framework:
Digital Witness (DW) is a device that is capable of identifying and collecting digital evidence, preserving it in a protected space, and sending it to other DWs or digital custodians (DC) who are authorized to participate in the proposed framework;
Digital Custodian (DC) is an entity designated by the law enforcement agency to help collecting the evidence submitted from DWs. It could be a physical entity, such as a police officer with warrant, or a virtual service appointed by the agency, responsible for a specific type of evidence collection.
Law Enforcement Agency (LEA) is a trusted entity that collects, analyzes, feedbacks, archives and disposes evidence gathered from DW and DC. The LEA discussed in our framework contains the following two core subsystems:

- a blockchain platform supported by a group of trusted and distributed servers, for recording all events occurred in the life cycle of the evidence, as well as the entities that are related to the evidence. Events to be recorded could include device registration, evidence submission, further evidence collection, archive and disposal, etc.;
- an evidence analysis platform, which is controlled by the LEA to perform analysis on the evidence submitted by DWs/DCs. We assume this platform, by incorporating advanced evidence analytical hardware/software, and based on the submitted evidence, will provide analytical results to DWs/DCs. The discussion of the detailed platform analytical capability is out of the scope of this work.

Each entity is identified through a cryptographic public key, obtained during registration with the LEA. We further define the entity access rights in the proposed blockchain system:
Read: all entities have the read access to the system, in other words, be able to view the "transaction" and "ledger" recorded in the blockchain system.
Write: the write access is divided into two sub-categories, namely "transaction write" and "ledger write". All the entities

have "transaction write" capabilities, that is to create transactions related to the activities centered around the evidence. For "ledger write", only the designated servers controlled by the LEA can perform block formation task, that is to aggregate the transactions and create new blocks. Such a design is due to the sensitivity and security of the entire use case.

Verify: the transactions created by the entities need to be verified before the designated servers could form immutable blocks. In our proposed system, we only allow DWs and LEA servers to have such a capability.

One may argue the need of using a blockchain in such a forensic application, as a centralized and trusted LEA exist. This seemingly contradicts to the original motivation of blockchain that should be deployed in a fully decentralized manner. However, apart from this decentralized property, another important essence of blockchain is the way of recording evidence in an integrity protected, authenticity guaranteed and non-repudiation way through hash chains and digital signatures. These properties reduce the trust among the participants and enhance the system reliability compare to a centralized approach. With such motivation, we propose a permissioned blockchain system for our IoT forensic framework.

The above listed entities and access rights are neither exhaustive nor restrictive. We only provide a minimum working skeleton for an evidence gathering platform. One could define more roles and rights that fit to the specific use case.

B. Design Details

The proposed system exists as a blockchain application within a distributed network. Every role we mentioned in the previous section is an individual peer node of the network. The procedure of evidence collection and transfer is achieved by inter-network collaboration based on a shared blockchain application among different entities. The essential component of a blockchain application is a distributed ledger, which is initiated and maintained by LEA (or a group of designated servers controlled by LEA). Below, we provide the design details of transaction, smart contract, blocks and consensus, which are essential elements constitute a blockchain system.

1) Transaction: as the atomic unit in the system, each transaction corresponds to an action performed by one or multiple participants. Its structure carries not only necessary information to describe the action (e.g. evidence submission), but the unique identifier to differentiate itself from others.

Fig. 1 defines the transaction format. To provide a detailed description of an action, the transaction type, time stamp, identity of a device, the payload and other supplementary data such as geographic location are included to the transaction. Moreover, the entity that submits a transaction is also required to calculate the hash digest of the transaction based on the body and payload, using a hash function such as SHA-256. This digest is then used as the unique identifier for this particular transaction. With the identifier, body and payload, the submitter signs the transaction using his private key (obtained during registration) and appends the digital signature to the end of the transaction, broadcasts it to the network, and waits for it to be verified and finally appends into the ledger.

Identifier	Hash digest of the transaction
Body	Transaction Type
	Timestamp
	Submitter ID
	Receiver ID (optional)
	Geographic position (optional)
Payload	Data payload (optional)
Digital Signature	Digital signature of the submitter

Fig. 1. Transaction format

In accordance to the life cycle of a digital evidence, we define five types of transaction:

Device Registration. All devices, regardless of their roles, need to register with the LEA before being granted with access to the system. To submit a registration transaction, the owner of the device needs to provide his personal identity information together with the unique identifier (UID) of the device.

Case Creation. This type of transaction is submitted by the digital witness with the primary evidence (e.g. the trace of an attack to the device) embedded into the payload field. The recipient of this transaction could be a digital custodian, the evidence analysis platform, or the LEA.

Additional Evidence. When the primary evidence is assessed to be insufficient for the investigation, the corresponding receiving party of the Case Creation transaction could require the victim for additional evidence. Subsequently the victim could create a new and submit to the ledger on behalf of cooperative digital witnesses (Co-DWs).

Case Archiving. A case archiving transaction is expected to be submitted by LEA to indicate the completion of investigation and closure of the case.

Evidence Disposal. An evidence disposal transaction is performed by LEA too, to guarantee that the evidence containing privacy data have been safely disposed.

2) Smart Contracts: widely been used by blockchain platforms such as Ethereum and Hyperledger, smart contract is similar to a computer program code with a *unique identifier*. By providing with an input, it executes the logic pre-defined, and produces an output.The unique identifier allows the system entities to locate, call and run the smart contract correctly.

Each transaction could be associated with one or more smart contracts. In other words, by taking the transaction payload as input and obtaining an output, this output could be further used as an input to trigger another smart contract. To illustrate the concept, we provide the pseudocodes of the smart contract associated with two of the transactions defined above.

Algorithm 1 describes a high-level routine of the smart contract for the case creation transaction. The smart contract first retrieves the identity of the submitter along with the evidence data, then initiates a case instance with the information

Algorithm 1 *contract* CASE CREATION

```
$submitter \leftarrow transaction.author$
$evidence \leftarrow transaction.payload$
$case \leftarrow$ CREATECASE($submitter$)
$result \leftarrow$ ANALYSIS($case, evidence$)
if result == SUFFICIENT then
    ACKNOWLEDGE($submitter$, $case$)
else
    ASKFORADDITIONALEVIDENCE($submitter$, $case$)
end if
```

Algorithm 2 *contract* ADDITIONAL EVIDENCE

```
$submitter \leftarrow transaction.author$
$evidence \leftarrow transaction.payload$
if evidence is NULL then
    ASSIGNDIGITALCUSTODIAN($case$)
else
    $case \leftarrow transaction.case$
    $result \leftarrow$ ANALYSIS($case, evidence$)
    if result == SUFFICIENT then
        ACKNOWLEDGE($submitter$)
    else
        ASKFORADDITIONALEVIDENCE($submitter$)
    end if
end if
```

obtained from the submitted transaction, after that invokes the analysis service through the interface provided by the evidence analysis platform. Based on the analysis outcome, the smart contract accordingly replies the submitter. Algorithm 2 depicts the logical procedure of smart contract triggered by an additional evidence transaction. The smart contract repeats sending additional evidence to the analysis service until it is informed that all the submitted evidence by now are sufficient for the current case, or the submitter has no more evidence to provide. In the latter case, the smart contract will look for other witnesses and request relevant evidence from them. Due to space limitation, the subroutine defined in these 2 algorithm are not presented in this paper, and the details of case archiving and evidence disposal, are omitted in the diagram.

With the transaction and smart contracts defined, Fig. 2 provides the sequence diagram from attack till the end of evidence analysis. The involved entities are listed at the top, where the diagram divides the different scenarios into subcategories. By submitting the evidence to the blockchain platform (step 3) after the victim has been attacked (step 1) and evidence gathered (step 2), the system will trigger the evidence analysis process (steps 3.1 to 3.3) and feedback to the victim. Subsequently, based on the analytical results, the system may either close the case (sufficient evidence, step 4.1), or ask the victim to provide additional information, either from himself (step 4.2.1), or other Co-DWs (step 4.3.1-4.3.6). Due to space limitation, case archiving and evidence disposal are omitted.

3) Block: The blockchain ledger is composed of a chain of data structures called "block". Starting from the very first "genesis" block generated at the moment of the ledger being initiated, all the subsequent blocks are created by linking to the previous one on the ledger. Block is a data structure bearing a number of verified transactions performed during a specific time frame. In BIFF, the formation of blocks is executed by the miners, a set of pre-defined nodes, by collecting and verifying all newly submitted transactions in the network.

Fig. 3 illustrates the key components in a block. Each block contains block identifier, body, payload, and the digital signature field. The block identifier is the hash output of the concatenation of the body and payload, and is used for the next block generation to chain the blocks together. The body field contains information such as timestamp, miner ID (the identifier of the entity who creates this block), the previous block identifier (to chain the blocks), and proof of consensus (explained below). The payload includes all the transactions collected for the past epoch. The digital signature is created based on the concatenation of the first three fields.

Again, the above defined transaction format, types, smart contracts and block formats are just minimum skeleton of the system. One could enrich the definition whenever deem fit.

4) Consensus: One of the greatest challenges in the distributed system is to ensure the entire community to have a unified view of the current network state. Consensus protocol, as its name suggests, provides such a mechanism. Commonly known consensus such as "proof-of-work" (PoW), whereby the entities in the system need to perform computational hard tasks to create a block, or "proof-of-stake" (PoS) which the block creation is based on the total "stake" one holds, are typically employed in permissionless blockchains such as Bitcoin and Ethereum. Both these mechanisms assume no-trust environment among the system participants, with the trade-off of heavy computation required for each block creation.

In BIFF, we utilize a different consensus protocol called Byzantine Fault Tolerance (BFT) [18], which is typically used in a permissioned blockchain. For each pre-defined epoch, the system selects one "leader" from the designated entities (e.g. servers under LEA's control). This leader then collects the unconfirmed transactions, forms a block, and includes its ID into the miner ID field. This particular block is then broadcast to the entire network and verified by the community. Once the number of successful verification passes a pre-defined threshold, this particular block is considered as valid and written into the immutable ledger. The "proof-of-consensus" to be included into the blocks are the digital signatures generated by the entities who have successfully verified the blocks.

BFT is a well-studied protocol and many variants exist. Due to space limitation we will not further elaborate the details. By selecting different variants, the algrithm execution procedure could slightly differ and the consensus threshold level could be adjusted based on the security level required. For a very basic BFT scheme where the threshold is set to be 2/3, an attacker still needs to compromise at least 1/3 of the entities to launch a successful attack for modifying the ledger, which is almost impossible consider the number of entities in the system.

IV. DISCUSSION

In this section, we discuss the features provided by the proposed system, followed by proposing a modified Merkle signature that resolves user identity privacy issue.

A. System Features

A DW signs on the digital evidence that he intends to report before sending it to the DC. The signature along with the DW's

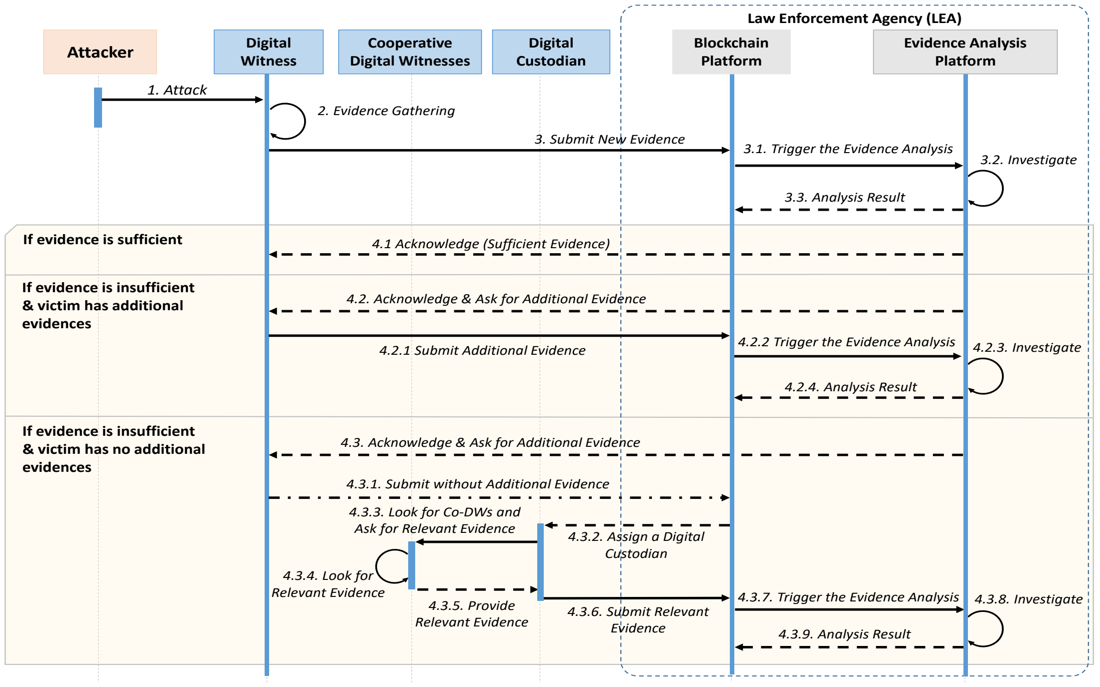
Law Enforcement Agency (LEA)
Attacker
Digital Witness
Cooperative Digital Witnesses
Digital Custodian
Blockchain Platform
Evidence Analysis Platform
1. Attack
2. Evidence Gathering
3. Submit New Evidence
3.1. Trigger the Evidence Analysis
3.2. Investigate
3.3. Analysis Result
If evidence is sufficient
4.1 Acknowledge (Sufficient Evidence)
If evidence is insufficient & victim has additional evidences
4.2. Acknowledge & Ask for Additional Evidence
4.2.1 Submit Additional Evidence
4.2.2 Trigger the Evidence Analysis
4.2.3. Investigate
4.2.4. Analysis Result
If evidence is insufficient & victim has no additional evidences
4.3. Acknowledge & Ask for Additional Evidence
4.3.1. Submit without Additional Evidence
4.3.3. Look for Co-DWs and Ask for Relevant Evidence
4.3.2. Assign a Digital Custodian
4.3.4. Look for Relevant Evidence
4.3.5. Provide Relevant Evidence
4.3.6. Submit Relevant Evidence
4.3.7. Trigger the Evidence Analysis
4.3.8. Investigate
4.3.9. Analysis Result

Fig. 2. Sequence diagram for evidence collection

Identifier	Hash digest of the block
Body	Timestamp
	Miner ID
	Hash digest of the previous block
	Proof of consensus
Payload	List of enclosed transactions
Digital Signature	Digital signature of the miner

Fig. 3. Block format

public key determines the unique device's owner. This not only offers the ownership for evidence, but also discourages false reporting. Apart from providing the authenticity of the evidence, the signature also ensures the integrity: if a malicious user tries to modify an evidence, as long as the private signing key is secure, the attacker could not generate a valid signature based on the modified content, and thus such an act could be detected by the entire community.

Once the transactions containing the digital evidence are recorded into the blockchain, their integrity is further protected through hash values of each block by chaining the blocks together. Moreover, the use of consensus mechanism such as BFT guarantees that any new creation of block has to reach a majority agreement before being recorded into the distributed ledger. This further increases the difficulty for an attacker should he wishes to sabotage the system: he has to compromise more designated nodes in the system which are involved in the consensus protocol, compare to the traditional way of just a few (possibly centralized) servers.

By utilizing smart contracts, all transactions will be performed automatically and recorded on the blockchain. Transparency is guaranteed as all these transactions could be publicly read and further verified by all participants of the system.

B. *Identity Privacy*

The proposed framework requires each digital witness to be bound to the identity of its owner. While this prevents people from reporting fake evidence, it may potentially affect the user privacy. Although each entity in the system is identified through a cryptographic public key and achieves certain degree of anonymity, such a static representation may lose its benefit if the link between the public key and true identity is leaked. This may discourage people from joining the system.

To overcome such a privacy limitation, we incorporate a modified Merkle signature scheme into our system. Briefly, such a scheme first generates 2^n intermediary private/public key pairs (X_i, Y_i) using a one-time signature scheme. By employing a hash function H, the system computes $h_i = H(Y_i)$ for $1 \leq i \leq 2^n$. With all the h_i values, a hash (or Merkle) tree is built by taking the h_i values as leaf nodes, and recursively hashing two adjacent nodes to form a binary tree. To be more specific, let $a_{k,\ell}$ denote the node in the tree with height k (counting from the bottom) and position ℓ (from left). The bottom of the tree is filled with $h_i = a_{0,i}$ as the leaves. For each inner node, one further calculates the hash of the concatenation of its two children. For example, $a_{1,0} = H(a_{0,0}||a_{0,1})$ and $a_{2,0} = H(a_{1,0}||a_{1,1})$. The root of the tree, $a_{n,0}$, is denoted as the public key *pub* of the Merkle signature scheme. For each leaf node $a_{0,i}$, the internal nodes that are adjacent to the path from $a_{0,i}$ to the root form the authentication path $auth_i$ of $a_{0,i}$. To verify a signature sig_i, the signer needs to provide both Y_i (to verify sig_i) and $auth_i$ (to reconstruct and verify pub). Readers may refer to [19] for more details of Merkle signature.

Adopting a Merkle signature straightforwardly still leaks the identity privacy, as by using the authentication path and intermediary signing public key Y_i will allow the attacker to deduce the public key pub. One approach is to let only the LEA control the authentication path and pub, but hide them from the public. However this disables the signature verification capability of the participants, and contradicts to the fundamental intuition of a blockchain system. To overcome this limitation, we further employ a digital certificate mechanism into the Merkle signature. Such modification protects the entity privacy, while enabling the system participants to verify the signature associated with each transaction. We describe our modified protocol as follows:
1. During device registration, same as the normal Merkle signature, the LEA generates 2^l private/public key pairs (X_i, Y_i) for each device. With Y_i as intermediary public keys, LEA builds a Merkle tree, and obtains the public key pub.
2. For each Y_i, LEA further generates a digital certificate $cert_i$ by signing Y_i using its own private key. This is to provide authenticity for each Y_i it has generated.
3. LEA only sends the 2^l pairs of $(X_i, Y_i, cert_i)$ to the DW. The Merkle root pub and the authentication paths $auth_i$ corresponding to Y_i are kept secret at the LEA.
4. For each transaction, DW uses a different pair of (X_i, Y_i) to generate signature sig_i, then $(sig_i, Y_i, cert_i)$ is broadcast into the blockchain network together with the transaction.
5. The system participants could verify the signature using Y_i, and further verify the authenticity of Y_i using $cert_i$ and the LEA public key. Should the LEA need to further identify the true public key of the user, it queries its database to get back $auth_i$ and pub, and then verifies if Y_i belongs to the tree. If so, it maps pub to the corresponding identity and associates the reported evidence to the unique user.

A digital witness uses his private/public keys (X_i, Y_i) for transaction signature generation in a non-repeating way. As no information (i.e., authentication path) maps to the Merkle root pub, even if a malicious user has captured some signatures and the associated public keys Y_i, he will not be able to derive the Merkle root and hence the identity of the digital witness. To further prevent the attacker from collecting all the public keys and thus deducing the Merkle root (should he is a persistent attack), the system could require the digital witness not to use all Y_i. By simply left one public key unused, the attacker, even collecting all other Y_i, could not reconstruct the Merkle root and learn the DW identity. Finally, after running out of key pairs, DW could request the LEA for generating a new set of key pairs and replacing the Merkle root. As this procedure occurs at a very low frequency, the computational and communication overhead could be ignored.

V. Conclusion

In this work, we propose a blockchain-based IoT forensic evidence gathering framework. We define the workflow for the entire process from evidence gathering, transmission, to analysis and eventual archiving and disposal. Leveraging on smart contract, we craft different transaction types that are suitable for this forensic application. To further address the identity privacy issue, we utilize a modified Merkle signature scheme to hide the identity of the evidence submitter from the public. One possible future work is to implement the framework into a IoT testbed that contains a heterogeneous group of devices, to test the framework reliability and benchmark the performance.

Acknowledgment

This material is based on research work supported by the Singapore National Research Foundation under NCR Award No. NRF2014NCR-NCR001-034.

References

[1] A. Nieto, R. Roman, and J. López, "Digital Witness: Safeguarding Digital Evidence by Using Secure Architectures in Personal Devices," *IEEE Network*, vol. 30, no. 6, pp. 34–41, 2016.
[2] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," Tech. Rep., 2008.
[3] K. Kent, S. Chevalier, T. Grance, and H. Dang, "Guide to Integrating Forensic Techniques into Incident Response," *NIST Special Publication*, vol. 10, pp. 800–86, 2006.
[4] R. Hegarty, D. J. Lamb, and A. Attwood, "Digital Evidence Challenges in the Internet of Things," in *The Ninth International Workshop on Digital Forensics and Incident Analysis*, 2014.
[5] S. Zawoad, A. K. Dutta, and R. Hasan, "SecLaaS: secure logging-as-a-service for cloud forensics," in *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*. ACM, 2013, pp. 219–230.
[6] M. Taylor, J. Haggerty, D. Gresty, and R. Hegarty, "Digital Evidence in Cloud Computing Systems," *Computer Law & Security Review*, vol. 26, no. 3, pp. 304–308, 2010.
[7] E. Miranda Lopez, S. Y. Moon, and J. H. Park, "Scenario-Based Digital Forensics Challenges in Cloud Computing," *Symmetry*, vol. 8, no. 10, p. 107, 2016.
[8] ICO, "Personal Information Online Code of Practice," 2010, https://ico.org.uk/media/for-organisations/documents/1591/personal_information_online_cop.pdf.
[9] A. Nieto, R. Rios, and J. López, "Digital Witness and Privacy in IoT: Anonymous Witnessing Approach," in *IEEE Trustcom/BigDataSE/ICESS*, 2017, pp. 642–649.
[10] ——, "IoT-Forensics Meets Privacy: Towards Cooperative Digital Investigations," in *Sensors*, 2018.
[11] H. Kopp, D. Mödinger, F. Hauck, F. Kargl, and C. Bösch, "Design of a Privacy-Preserving Decentralized File Storage with Financial Incentives," in *Euro S&P Workshop*, 2017, pp. 14–22.
[12] Q. Xia, E. B. Sifah, K. O. Asamoah, J. Gao, X. Du, and M. Guizani, "MeDShare: Trust-Less Medical Data Sharing Among Cloud Service Providers via Blockchain," *IEEE Access*, vol. 5, 2017.
[13] F. Tian, "An Agri-food Supply Chain Traceability System for China based on RFID & Blockchain Technology," in *13th International Conference on Service Systems and Service Management (ICSSSM)*, 2016.
[14] V. Grewal-Carr and S. Marshall, "Blockchain: Enigma. Paradox. Opportunity," Deloitte, UK, Tech. Rep., 2016.
[15] G. Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," Ethereum & ETHCore, Tech. Rep., 2016.
[16] "Hyperledger Fabric," https://hyperledger-fabric.readthedocs.io.
[17] N. Szabo, "The Idea of Smart Contracts," http://szabo.best.vwh.net/smart_contracts_idea.html, 1997.
[18] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," in *OSDI*, vol. 99, 1999, pp. 173–186.
[19] R. Merkle, "Secrecy, Authentication and Public Key Systems: a Certified Digital Signature," Ph.D. dissertation, 1979.