Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 4029591, 14 pages
https://doi.org/10.1155/2018/4029591

Research Article

A Blockchain-Based Contractual Routing Protocol for the Internet of Things Using Smart Contracts

Gholamreza Ramezan and Cyril Leung

Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, Canada

Correspondence should be addressed to Gholamreza Ramezan; gramezan@ece.ubc.ca

Received 27 July 2018; Accepted 10 October 2018; Published 1 November 2018

Guest Editor: Jiageng Chen

In this paper, we propose a novel blockchain-based contractual routing (BCR) protocol for a network of untrusted IoT devices. In contrast to conventional secure routing protocols in which a central authority (CA) is required to facilitate the identification and authentication of each device, the BCR protocol operates in a distributed manner with no CA. The BCR protocol utilizes smart contracts to discover a route to a destination or data gateway within heterogeneous IoT networks. Any intermediary device can guarantee a route from a source IoT device to a destination device or gateway. We compare the performance of BCR with that of the *Ad-hoc* On-Demand Distance Vector (AODV) routing protocol in a network of 14 devices. The results show that the routing overhead of the BCR protocol is 5 times lower compared to AODV at the cost of a slightly lower packet delivery ratio. BCR is fairly resistant to both Blackhole and Greyhole attacks. The results show that the BCR protocol enables distributed routing in heterogeneous IoT networks.

1. Introduction

Recent progress in wireless communications and mobile computing has enabled a large variety of devices to connect to the Internet, forming the Internet of Things (IoT) [1, 2]. The IoT is a heterogeneous network of various types of devices from different vendors which collect, transfer, process, and analyze data and take appropriate actions [3, 4]. The IoT faces numerous challenges due to the need to integrate a large number of dissimilar objects.

Routing, which establishes a communication path from a source IoT device to a destination node, for example, a gateway, is one such challenge. A variety of routing protocols for IoT networks have been studied [5–9]. In [5], a routing protocol for low-power and lossy networks (RPL) was proposed. The RPL protocol is a promising routing protocol that is used in the large-scale BC Hydro smart meter project in British Columbia, Canada [10]. Providing secure communication and preventing attackers from interfering with the routing process are major concerns in this network.

The utilization of cryptographic algorithms is the first approach in securing routing protocols. However, in the design of most existing routing protocols, such as Secure *Ad-hoc* On-Demand Distance Vector (SAODV) [11], Ariadne [12], Optimized Link State Routing (OLSR), and optimal and secure routing (OSR) [13], the availability of a central authority (CA) to distribute the secret keys between network nodes is assumed [14–16]. The major problem is that the large number of IoT vendors cannot simply agree on a centralized management system. This is due to the trust issue between IoT vendors and the high cost of implementing trust management infrastructures such as the Public Key Infrastructure (PKI).

The second approach is the reputation-based method that measures the degree to which a network node contributes to the routing process [17, 18]. In [17], a reward mechanism is proposed to incentivize nodes to participate in the routing process. Each network node is selected based on its reputation in the routing process. The reputation information is derived either from observing the behaviour of its neighbors or from trusted external advisors in the network. In both cases, the accuracy of the reputation system can be affected either because of the limited network view of a network node based solely on viewing its neighbors, or from its attackers'

falsification of reputation information coming from external trusted systems [19].

The lack of trust in a central management system and the need for a publicly verifiable reputation system lead us to leverage public ledger techniques, such as blockchain, to design routing protocols for the IoT.

In this paper, we introduce a decentralized blockchain-based contractual routing (BCR) protocol. The BCR protocol enables IoT devices from diverse vendors to trust one another and cooperate during data communication. Using this approach, the devices in a delay-tolerant IoT network can find routes to a gateway or destination device in a decentralized manner. The main contributions of the paper are as follows:

(i) We propose contractual routing as a blockchain-based routing protocol for the IoT. A public ledger system is used to decentralize the BCR protocol.

(ii) We provide a proof of concept of the BCR protocol using the Ethereum blockchain and consider the following four performance metrics: Packet Delivery Ratio (PDR), Throughput (TP), Routing Overhead (RO), and Route Acquisition Latency (RAL).

(iii) We compare the performance of BCR with that of *Ad-hoc* On-Demand Distance Vector (AODV) which is a commonly used routing protocol [20]. Our results show that the BCR has a slightly lower PDR but a much lower routing overhead.

(iv) We study the performance of BCR under Blackhole and Greyhole attacks by malicious devices which do not necessarily follow the smart contract rules.

The remainder of this paper is structured as follows. In Section 2, we review related works. Section 3 presents the system model. In Section 4, we discuss the attack model. In Section 5, we describe the proposed routing protocol. In Section 6, we compare the performances of the BCR and AODV protocols. Finally, the main conclusions are discussed in Section 7.

2. Related Works

Financial incentive models have been introduced in various works [17, 18, 21–24]. For example, *Ad-hoc* VCG [17] provides a game-theoretical setting for routing within mobile *ad-hoc* networks in which a node accepts a payment for forwarding data packets from other agents provided the payment exceeds its cost. The system provides the incentive for users to cooperate. In [18], Sprite is proposed as a model to reward each participant node when routing data packets. However, the approach still requires that nodes access a central system, such as a bank, to send a proof message which shows a data packet is delivered. The proof message includes digital signatures and node identities, so as to receive rewards from the bank. This method is vulnerable, as attackers can forge a proof message to be sent to a central management system to generate rewards. The Onion Router proposed in [23] is based on [24], a blockchain-based reward mechanism for anonymous routing. This routing needs a centralized network since it requires that nodes be assigned to their specific relay nodes, after which only these nodes will receive the data. The authors of [22] introduce the idea of monetizing routing protocols based on public ledger techniques, whereby reputation is traded as an asset. In contrast, we propose a communications network model and describe an implementation of our proposed decentralized BCR protocol. Furthermore, we analyze the performance of the proposed protocol.

3. System Model

In this section, we describe a model to implement the proposed BCR protocol. The system consists of a multihop IoT network $\mathcal{I}_{\mathcal{S},\mathcal{I},\mathcal{D}}$ which cooperates with blockchain network $\mathcal{B}^{\mathcal{P},\mathcal{Q},\mathcal{G}}_{\mathcal{BP},\mathcal{BG}}$, as shown in Figure 1.

3.1. Multihop IoT Network. The IoT network $\mathcal{I}_{\mathcal{S},\mathcal{I},\mathcal{D}}$ consists of a set of Source devices $\mathcal{S}$, a set of intermediary devices $\mathcal{I}$, and a set of Destination devices and Data gateways $\mathcal{D}$. There is no central management for registration, authentication, or device authorization. The source device aims to send data to a destination device or a data gateway.

(i) ***Source devices*** $\mathcal{S}$: A Source device originates a request access to send data to a destination, or a data gateway. The gateway allows the source device access to the Internet to periodically update firmware or upload data to its vendors' cloud.

(ii) ***Intermediary devices*** $\mathcal{I}$: The devices with no direct connection to a destination or data gateway use other devices to relay their traffic. An IoT device that relays source device data traffic to a gateway or destination is referred to as an intermediary device.

(iii) ***Destination devices*** or ***Data gateways*** $\mathcal{D}$: Data gateways provide source devices access to larger networks, or the Internet. Data gateways can be access points within Wi-Fi networks, base stations in Multihop Cellular Networks (MCN) [25], or sink nodes in Wireless Sensor Networks (WSN) [6].

3.2. Blockchain Network. The system includes a blockchain network denoted by $\mathcal{B}^{\mathcal{P},\mathcal{Q},\mathcal{G}}_{\mathcal{BP},\mathcal{BG}}$ with the following parameters, components, and capabilities:

(1) Parameters: The blockchain has the following parameters [26]:

(i) **Common Prefix property $\mathcal{P}$ with the parameter $\mathcal{K} \in \mathbb{N}$:** Suppose the honest blockchain nodes, $\mathcal{BN}_1$ and $\mathcal{BN}_2$, maintain chains $\mathcal{C}_1$ and $\mathcal{C}_2$; then $\mathcal{C}_1^{-\mathcal{K}}$ would be a prefix of $\mathcal{C}_2$ and $\mathcal{C}_2^{-\mathcal{K}}$ a prefix of $\mathcal{C}_1$, where $\mathcal{C}^{-\mathcal{K}}$ is Chain $\mathcal{C}$ minus its last $\mathcal{K}$ blocks. We would call $\mathcal{K}$ the depth parameter.

(ii) **Chain Quality property $\mathcal{Q}$ with parameters $\mathcal{L} \in \mathbb{N}$ and $\mu \in (0, 1]$,** where $\mathcal{L}$ is the length of the blockchain owned by an honest node and $1 - \mu$ is the ratio of the greatest chain that can be created by an adversary. μ is called the chain quality coefficient.

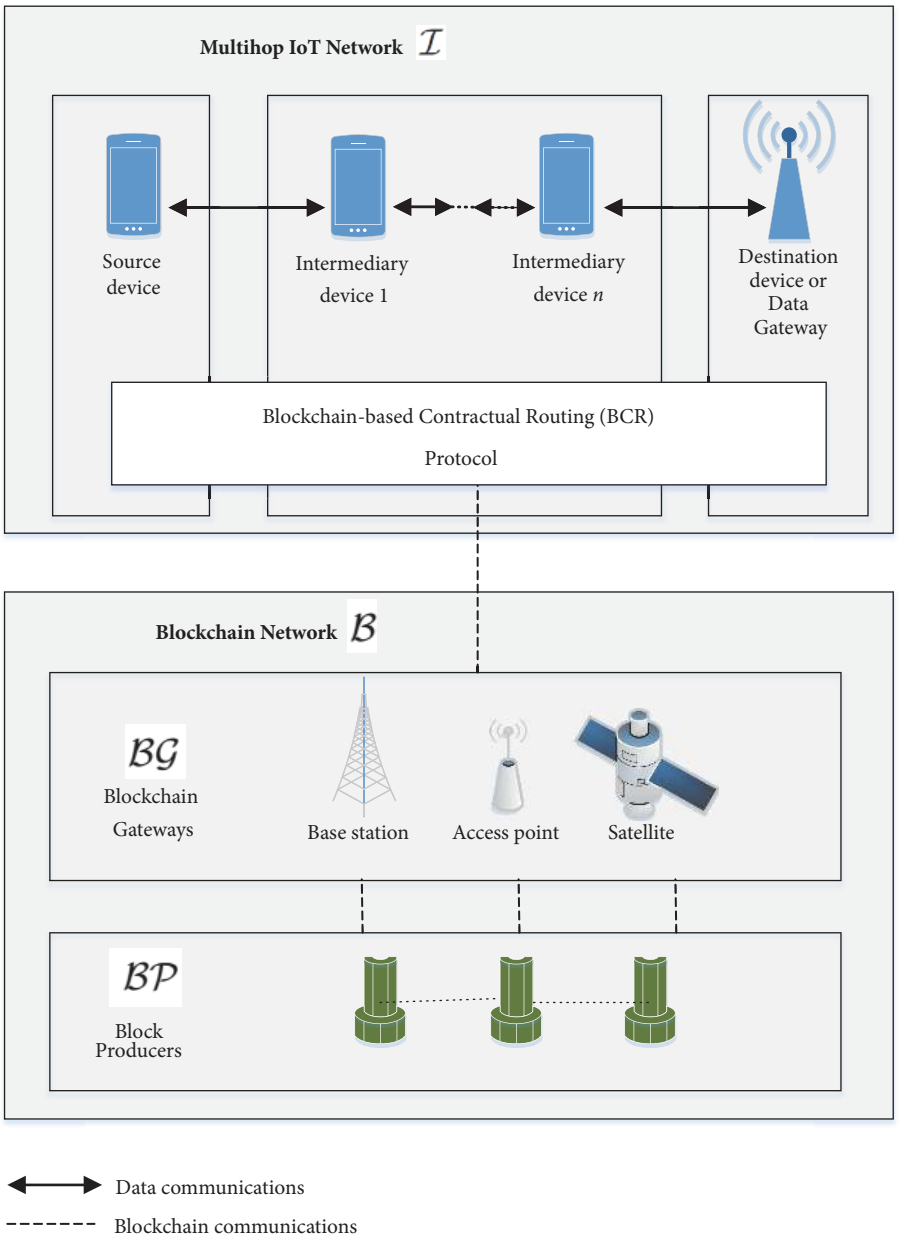

Figure 1: Setup for a decentralized communications network for IoT devices.

(iii) **Chain Growth property $\mathcal{G}$ with parameters $\mathcal{S} \in \mathbb{N}$ and $\tau \in (0, 1]$**, where, for any honest blockchain nodes, $\mathcal{BN}_1$ with Chain $\mathcal{C}$ for any $\mathcal{S}$ block times at least $\tau.\mathcal{S}$ blocks will be added to the blockchain. τ is called the speed coefficient.

The above parameters imply that the public ledger has the following two properties [27]:

(i) ***Liveness***: A submitted transaction from a network node to the blockchain block producers will appear in a block after a sufficient period of time. In other words, all transactions originating from the network nodes will eventually end up at a block within the blockchain.

(ii) ***Persistence***: Persistence means that once a transaction goes into the blockchain of one honest block producer, it will be included with very high probability in every honest block producer's blockchain and be consequently assigned a permanent position in the blockchain.

(2) Components: Our proposed blockchain network contains block producers $\mathcal{BP}$ and blockchain gateways $\mathcal{BG}$ as components:

(i) ***Blockchain gateways*** $\mathcal{BG}$: The blockchain gateways enable communication between IoT devices and the blockchain network. These gateways may be cellular base stations, Wi-Fi access points, or satellites.

(ii) ***Block Producers*** $\mathcal{BP}$: Each block producer receives transactions from the IoT network and assembles them into a block. It then attempts to add the newly generated block into the blockchain. Block producers may belong to IoT device vendors but none of them are trusted by other block producers. They must come to a consensus through blockchain consensus mechanisms about the transactions. Depending on the applied consensus algorithm, different security assumptions should be considered to preserve the properties of liveness and persistence. For example, the honest block producers should control at least 75% of the processing power in the block producers network if the Proof-of-Work (PoW) consensus mechanism is used [28].

(3) Capabilities: To apply blockchain technology to our system model, the blockchain network $\mathcal{B}$ should be capable of running programs. Several works have developed programming frameworks that take in high-level programs as specifications and generate cryptographic implementations [29–31]. The idea of programmable *smart contracts* dates back nearly twenty years [32]. Ethereum [29] is the first Turing-complete decentralized smart contract system. A contract can be run by calling on one of its functions, where each function is defined by a sequence of instructions. The smart contract maintains an internal state and can receive/transfer blockchain tokens from/to users or other smart contracts. Users send transactions to the Ethereum block producers network to invoke functions. Each transaction may contain input parameters for the contract and an associated token amount which is transferred from the user to the smart contract. The authors of [30] propose Hawk as a framework for building privacy preserving smart contracts. The Hawk compiler is in charge of compiling the program to a cryptographic protocol between the blockchain and its users. Hyperledger [31] is another blockchain development platform which supports smart contracts. Smart contracts on the Hyperledger platform are called chaincodes.

All the IoT devices, block producers, and gateways agree on the monetary value of a token. One of the ways for an IoT device to acquire tokens is by direct deposit from its owner into its blockchain address. The tokens can also be acquired from smart contracts. When an IoT device provides services, such as routing services for other IoT devices, the tokens assigned to a smart contract can be transferred from the smart contract address to the IoT device address on the blockchain.

4. Attack Model

In this section, we define the attackers' capabilities when they attack the BCR protocol. Attackers can be classified into two main categories: selfish and malicious nodes. A selfish node does not intentionally disrupt routing, but it drops other nodes' routing messages while using their resources to route its own messages. Detecting and mitigating a selfish node is difficult, since the node does not actively violate the routing protocol rules. Malicious nodes purposefully disrupt routing messages [22]. An attacker is a dishonest IoT device which holds a sufficient number of tokens to allow it to join a network and then attempts to interfere with the network's routing process by preventing honest IoT devices from accessing the data gateways.

(1) Anonymity: The network does not use any centralized authority to authenticate IoT devices. Any IoT device can generate its own private/public key pair. Based on the generated public key, the IoT device derives its own blockchain address. This provides anonymity for the network nodes because no one knows the identity of the owner of a new blockchain address.

(2) Token-based Authorization: Every IoT device which possesses a sufficient number of blockchain tokens is authorized to generate a smart contract and request a route to a gateway.

(3) Attacker's Violation Scope: An attacker can manipulate the routing protocol in its own IoT device. Therefore, it can violate the routing protocol procedures and rules. It is assumed that honest IoT devices have not been compromised; that is, the attackers are unable to access the private keys within honest IoT devices. An honest IoT device can process and properly follow the contractual routing protocol. For example, if an honest IoT device receives a smart contract with a zero-token bond, it will treat this as an invalid request.

(4) Attacker Exhaustion Defense Strategy: The defense strategy in the BCR protocol does not instantly halt an attack but, instead, it deters the attacker by gradually exhausting the attacker's tokens. Each honest IoT device has an internal mechanism which blacklists malicious IoT devices that interfere in the routing of previous data packets by preventing the packets from reaching a gateway. When an IoT device B is blacklisted by another IoT device A, A will prevent B from participating in the next smart contracts for a specified period.

(5) Sequential Punishment: If an attacker drops a data packet, every other intermediary IoT device on that route will be penalized by having to pay tokens to its previous intermediary IoT device. Each intermediary device will be paid in turn by the next intermediary device on the same route. This sequential punishment mechanism allows the routing protocol to punish the attacker which drops data packets.

(6) Transparency: All network nodes and attackers have access to the blockchain gateways and can acquire a copy of the blockchain data to learn about the smart contracts.

(7) Block producers: The blockchain is not compromised since it is assumed that the blockchain consensus algorithm works correctly. Thus, attackers cannot place a false transaction within a block in the blockchain through the block producers network.

The aim of the BCR protocol is to discourage attackers from interfering with packet routing, as such interference requires the expenditure of tokens. This mechanism permits different vendors' IoT devices to build trust in one another based on their past behaviors as they seek a route to a gateway, without the need for centralized certificated authority.

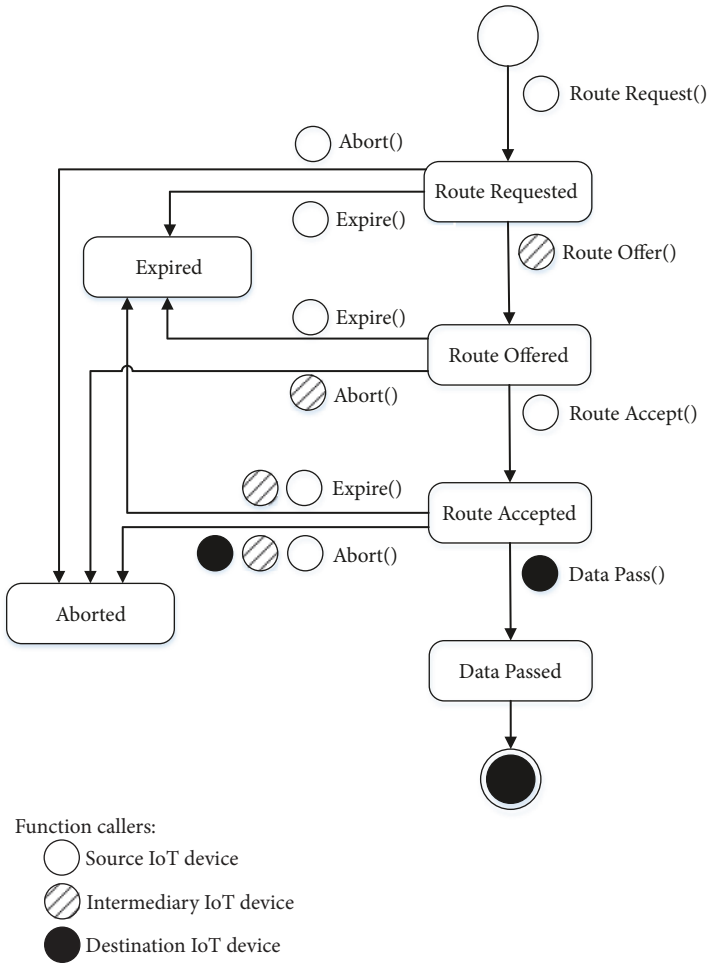

Figure 2: The protocol state machine of BCR protocol has 6 states. Transition between states occurs when IoT devices call functions inside smart contracts.

5. The Blockchain-Based Contractual Routing (BCR) Protocol

We first provide an overview of a general approach towards designing routing protocols. Existing routing protocols typically consist of two major phases. Phase 1 is for route establishment, while Phase 2 is for route maintenance. In Phase 1, a source IoT device sends a Route Request (RREQ) control message to find a route to a destination device. Each intermediary or destination device which receives the RREQ packet can respond by sending a Route Reply (RREP) message to the source IoT device. A Route Error (RERR) message is used to notify other devices that a certain device is no longer reachable, and they have to remove that route from their routing table.

In the proposed BCR protocol, each source IoT device creates a smart contract to request a route to a destination or data gateway for a specific period instead of creating RREQ control messages. Each smart contract created by an IoT device has a separate address within the blockchain that is generated by a block producer when placing a smart contract in a block. The IoT device can broadcast this address to its neighbors to inform them about a new routing request. The BCR protocol is implemented using smart contracts within the blockchain. The IoT devices request that the functions within the smart contract follow the BCR protocol. Thus, transmission of control messages in existing routing protocols is replaced by smart contract function calls in the BCR protocol. The BCR protocol is next explained in detail.

5.1. BCR Protocol States. Figure 2 shows the state machine diagram of the BCR protocol smart contract. The smart contract states are described below:

(i) **Route Requested:** When a source IoT device needs to reach a gateway, it creates a smart contract within the blockchain and sends the smart contract address to its neighbors. It also sets the state field within the smart contract to *Route Requested*. IoT devices do not necessarily need to know the data gateway address but can instead use an IPv6 address scheme, such as FF01::2, which allows devices to address any gateways or routers in the network [33]. The source IoT device transfers some of its own blockchain tokens as a bond to a smart contract address to create

a smart contract. The possibility of earning tokens encourages intermediary IoT devices to respond to the route request (*Route_Request_Bond*). The source IoT device also specifies the period for which the state of the route request within a smart contract is valid (*Route_Request_Expiry*). This smart contract is termed the original contract.

(ii) **Route Offered:** Each neighboring IoT device, which has a valid route entry to a gateway and would like to participate in relaying data packets (*Route_Offer_Validity*), can respond to an original smart contract. The intermediary IoT device offers its services to the source device by calling on a function within the original contract and transferring some of its own tokens to the smart contract address (*Route_Offer_Bond*). The function call goes to the block producers' network which changes the state of the received smart contract to *Route Offered*. A maximum of 3 route offers from different intermediary IoT devices can be stored in each contract. If the neighboring intermediary IoT device is unaware of a route to the data gateway or destination, it can still participate in relaying data packets by creating a new smart contract, namely, the intermediary contract. The intermediary contract stores the address of the originally issued smart contract or another intermediary contract in the *Parent_Contract* parameter.

(iii) **Route Accepted:** The source IoT device determines whether to accept an offered route to send its data packets. It selects the next neighbor to reach a gateway based on its own internal policies. It can choose a low-cost route offered by one of its neighbors or multiple neighbors to act as a relay(s) in order to increase the security and throughput of data packets.

(iv) **Route Passed:** When data is received by the data gateway, the smart contract state is changed to *Data Passed* by the gateway. If an IoT intermediary device B offers a route, but is unable to successfully relay the source IoT device's data packets within the specific time mentioned in the smart contract, the source IoT device will place the B's address to its internal blacklist for a limited period (*Blacklist_Timer*). The source IoT device will add its current blacklist addresses to any newly created smart contract's blacklist (*Blacklisted_Addresses*).

(v) **Aborted:** At any time, each device in the IoT network can abort the routing process by calling on the Abort function inside the smart contract. However, the smart contract Abort function acts accordingly based on its caller IoT device type and the current state of the smart contract.

(vi) **Expired:** As the BCR protocol has various timers, an IoT device can request that the Expire function inside a smart contract to review the timers and take action accordingly.

5.2. BCR Protocol Transitions. A protocol transition specifies the required conditions that triggers a state change. IoT devices perform the trigger when calling up a function inside the BCR protocol smart contract. We next review the parameters used by the functions inside the BCR protocol smart contract. Then, we explain the functions of the smart contract. The IoT devices call on these functions to run the BCR protocol.

(1) BCR protocol parameters: BCR protocol parameters within a smart contract are used by smart contract functions and can be seen publicly. The BCR protocol parameters within an IoT device are set by the IoT device based on its own internal policy. Each IoT device can have its own values for these internal parameters. The required parameters for a BCR protocol as listed in Table 1:

(i) *Contract_Address* stores the smart contract address. A smart contract can be dynamically created inside a blockchain by a source IoT device, or previously created by the IoT device owner. In the latter case, the IoT device owner, after creating a smart contract inside a blockchain, writes the address inside the IoT device.

(ii) *State* indicates the current state of a smart contract. Possible states are *Route Requested*, *Route Offered*, *Route Accepted*, *Data Passed*, *Expired*, and *Aborted* as explained in the previous section.

(iii) *Source, Intermediary,* and *Destination* store the addresses of the source, intermediary, and destination IoT devices. The source IoT device has requested access to a data gateway. The intermediary devices are ready to relay the data packets from the source IoT device to a destination or data gateway. This field in each smart contract stores up to three intermediary IoT device addresses. Destination IoT device is the destination node to be reached. In the Performance Evaluation section, we attempt to reach a data gateway network address as the destination, for example, FF01::2, that refers to any routers in an IPv6 network.

(iv) *Route_Request_Expiry (RRE)* is the expiry time until which the route request is valid.

(v) *Route_Request_Bond (RRB)* is set by the source IoT device and shows the number of tokens that the source IoT device will pay to the intermediary IoT device if the route to the destination works properly and the destination receives the data packets.

(vi) *Route_Offer_Validity (ROV)* shows the period for which the route offered by an intermediary IoT device to a source IoT device is valid. In other words, the intermediary IoT device relays the data packets to a gateway for the source IoT device only for a period which is specified by the ROV parameter.

(vii) *Route_Offer_Bond (ROB)* is the number of tokens an intermediary IoT device puts as a bond to guarantee that the intermediary IoT device can successfully pass the data packets to the gateway.

Table 1: BCR protocol parameters.

	Location	Parameter Name	Abbreviation
1	Inside smart contract	*Contract_Address*	
2	Inside smart contract	*State*	
3	Inside smart contract	*Source*	
4	Inside smart contract	*Intermediary*	
5	Inside smart contract	*Destination*	
6	Inside smart contract	*Route_Request_Expiry*	RRE
7	Inside smart contract	*Route_Request_Bond*	RRB
8	Inside smart contract	*Route_Offer_Validity*	ROV
9	Inside smart contract	*Route_Offer_Bond*	ROB
10	Inside smart contract	*Blacklisted_Addresses*	
11	Inside smart contract	*Selected_Route*	
12	Inside smart contract	*Timestamp*	
13	Inside smart contract	*Parent_Contract*	
14	Inside smart contract	*Hop*	
15	Inside smart contract	*Gas*	
16	Inside IoT device	*Blacklist_Timer*	
17	Inside IoT device	*Max_Hop*	

∗**Location** shows whether the parameter is used within a smart contract or an IoT device.

(viii) *Blacklisted_Addresses* stores a list of device addresses which are not allowed to participate in the smart contract for a certain period of time (*Blacklist_Timer*). This parameter is set by the source IoT device every time one of its neighbors fails in relaying data to a data gateway. Therefore, the intermediary addresses are restricted from putting forward any smart contract offer.

(ix) *Selected_Route* stores the intermediary address which is selected by the source IoT device for data packet forwarding. This address is selected from one of addresses in *Intermediary* parameter.

(x) *Timestamp* logs the time at which the smart contract is created in the blockchain. This field is set by block producers.

(xi) *Parent_Contract* stores the address of the previously issued smart contract. If the smart contract is an original one not preceded by a previously issued smart contract, the *Parent_Contract* parameter is empty. After receiving a smart contract, the IoT device checks this parameter to ensure that the previous smart contract was not self-issued. Using this mechanism, the routing protocol avoids a loop from occurring in the routing protocol.

(xii) *Hop* stores the number of hops from the source IoT device to the current intermediary IoT device. The intermediary device, after receiving a smart contract, checks its own routing table. If no route to a data gateway is found, it creates a new smart contract and sets this field in the newly created smart contract by increasing the *Contract_Hop* parameter value in the previous contract. Intermediary nodes use this parameter to prevent the creation of a routing loop if the parameter exceeds a *Max_Hop* or maximum value.

(xiii) *Gas* is a term used in the Ethereum blockchain to define the cost of calling on a function inside a smart contract via a source or intermediary IoT device. *Gas* shows the number of tokens that an IoT device should pay to the block producers when a smart contract's internal functions are run by the block producer.

(2) BCR protocol functions: The transition between smart contract states is performed by calling on the smart contract functions. Every time an IoT node calls on a function, some tokens as specified in the *Gas* of the function will be moved from the IoT device blockchain account to that of the block producer.

(i) **Route Request():** Each IoT device, whenever it needs to reach a destination or data gateway, can request that the blockchain producers create a smart contract on the blockchain. The source IoT device digitally signs a transaction for this purpose and sets the smart contract's parameters. This function is shown in Algorithm 1.

(ii) **Route Offer():** This takes place when an intermediary IoT device establishes a route to the destination or data gateway in its internal routing table and is ready to relay data packets to it for a source IoT device. Each contract accepts up to three route offers from intermediary devices. This function is shown in Algorithm 2.

(iii) **Route Accept():** Whenever a source IoT device decides to accept an offered route, it calls on the Route Accept function within the blockchain. The Block Producer runs this function if the function caller IoT

```
1: function ROUTE REQUEST(DESTINATION, RRB, RRE, BLACKLIST, PARENTADDRESS(OPTIONAL), HOP(OPTIONAL) )
2:     Transfer Gas tokens from the function caller to the block producer.
3:     Transfer RRB tokens from the function caller to the current contract address
4:     Set RRE to Route_Request_Expiry
5:     Set Blacklist to Blacklisted_Addresses
6:     if this is an original smart contract then
7:         Set Hop to 0
8:     end if
9:     if this is an intermediary smart contract then
10:        Set Hop to Hop
11:        Set Parent_Contract to ParentAddress
12:    end if
13:    Set Timestamp to Now
14: end function
```

Algorithm 1: Route Request function.

```
1: function ROUTE OFFER(ROB, ROV)
2:     Transfer Gas tokens from the function caller to the block producer.
3:     if the function caller address is not in Blacklisted_Addresses and the number of offers is less than three then
4:         Transfer ROB tokens from the function caller to the current contract address
5:         Set ROV to Rout_Offer_Validity
6:     end if
7: end function
```

Algorithm 2: Route Offer function.

```
1: function ROUTE ACCEPT(INTERMEDIARY)
2:     Transfer Gas tokens from the function caller to the block producer.
3:     if the function caller is Source then
4:         Move the intermediary to Selected_Route
5:         Transfer the ROB tokens of the other intermediary devices back
6:     end if
7: end function
```

Algorithm 3: Route Accept function.

device's address is identical to that of the source IoT device within the smart contract. This function is shown in Algorithm 3.

(iv) **Data Pass():** Whenever a destination IoT device receives data packets, it can call on the Data Pass function within the blockchain. The block producer runs the function if the function caller address is the same as the destination address within the smart contract. This function is shown in Algorithm 4.

(v) **Expire():** Whenever a destination IoT device receives the data packets, it can call on the Data Pass function inside the blockchain. The Block Producer runs the function if the function caller's IoT device's address is identical to that of the destination IoT device's address within the smart contract. This function is shown in Algorithm 5.

(vi) **Abort():** Whenever an IoT device wishes to leave the contract, it can call on the Abort function. Depending on the state of the contract, the Abort function returns the tokens to the IoT devices. This function is shown in Algorithm 6.

6. Performance Evaluation

We now study the performance of the BCR protocol in a network with no CA or node authentication support. We compare the performance of the BCR with that of the AODV routing protocol. We also assess the impact of Blackhole and Greyhole attacks on the BCR protocol.

6.1. Simulation Setup. We investigate the BCR protocol by developing a simulator using the Ethereum blockchain and Solidity language [29] to provide a proof of concept of the protocol. The average time between two consecutive blocks

```
1: function DATA PASS()
2:    Transfer Gas tokens from the function caller to the block producer
3:    if the function caller is Destination then
4:       Transfer the Rout_Request_Bond and Rout_Offer_Validity tokens to the Selected_Route
5:    end if
6: end function
```

Algorithm 4: Data Pass function.

```
1: function EXPIRE()
2:    Transfer Gas tokens from the function caller to the block producer.
3:    if state is Route Requested then
4:       if current time is more than Route_Request_Expiry then
5:          Transfer Route_Request_Bond tokens back to Source
6:          Transfer Route_Offer_Bond tokens back to Intermediary
7:       end if
8:    end if
9:    if state is Route Offered then
10:      if current time is more than Route_Offer_Validity then
11:         Transfer Route_Request_Bond tokens back to Source
12:         Transfer Route_Offer_Bond tokens back to Intermediary
13:      end if
14:   end if
15:   if state is Route Accepted then
16:      if the function caller is Intermediary or Destination then
17:         Transfer Route_Request_Bond and Route_Offer_Bond tokens to Source
18:      end if
19:      if the function caller is Source then
20:         Transfer Route_Request_Bond and Route_Offer_Bond tokens to Selected_Route
21:      end if
22:   end if
23: end function
```

Algorithm 5: Expire function.

in a blockchain is called block time. Since the Ethereum block time is 14 seconds, it may not be suitable for real time telecommunication applications as it is too long for interactive applications. In the EOS blockchain, the block time is much shorter, 0.5 sec, that makes it suitable for real implementation of the BCR protocol.

We study different scenarios for Greyhole and Blackhole attacks [34]. The source IoT device generates one Route Request smart contract for each 1000-byte data packet. The simulation parameters are summarized in Table 2.

The performance of the BCR protocol is evaluated based on the following metrics:

(i) *Packet Delivery Ratio (PDR)* is given by

$$PDR = \frac{D_{\text{rcv}}}{D_{\text{total}}}, \tag{1}$$

where D_{rcv} is the number of data packets successfully received by the gateway and D_{total} is the total number of data packets sent by the source IoT device.

(ii) *Throughput (TP)* is the average number of data packets successfully received per second by the gateway and is given by

$$TP = \frac{D_{\text{rcv}}}{T_{\text{sim}}}, \tag{2}$$

where T_{sim} is the simulation duration.

(iii) *Routing Overhead (RO)* is given by

$$RO = \frac{D_{\text{net}} + D_{\text{ctrl}}}{D_{\text{net}}}, \tag{3}$$

where D_{net} is the total number of passed data packets. We considered 1000 data packets for each smart contract. D_{ctrl} is the total number of control messages; that is, the number of function calls in smart contracts. Each function call in a smart contract is assumed to need a 100-byte control packet.

(iv) *Route Acquisition Latency (RAL)* is the average time between the generation of a smart contract and

```
1: function ABORT()
2:     Transfer Gas tokens from the function caller to the block producer.
3:     if state is Route Requested and the function caller is Source then
4:         Transfer Route_Request_Bond tokens back to the function caller
5:     end if
6:     if state is Route Offered then
7:         if the function caller is Source then
8:             Transfer Route_Request_Bond tokens back to the function caller
9:             Transfer Route_Offer_Bond tokens of all intermediary devices back to them
10:        end if
11:        if the function caller is Intermediary then
12:            Transfer Route_Offer_Bond tokens back to the function caller
13:        end if
14:    end if
15:    if state is Route Accepted then
16:        if the function caller is Intermediary or Destination then
17:            Transfer Route_Request_Bond of the Selected_Route and Route_Request_Bond tokens back to Source
18:        end if
19:        if the function caller is Source then
20:            Transfer Route_Request_Bond of the Selected_Route and Route_Request_Bond tokens back to Intermediary
21:        end if
22:    end if
23: end function
```

Algorithm 6: Abort function.

Table 2: Simulation parameter values.

	Device Type	Parameter Name	Value
1	S	Route_Request_Bond (RRB) (tokens)	100
2	S	Route_Request_Expiry (RRE) (sec)	800
3	I	Route_Offer_Bond (ROB) (tokens)	10
4	I	Route_Offer_Validity (ROV) (sec)	650
5	I	Route_Request_Bond (tokens)	(*prevRRB*)–10
6	I	Route_Request_Expiry (sec)	(*prevRRE*)–150
7	S,I	Blacklist_Timer (sec)	300
8	S,I	Max_Hop	5
9	-	simulation period (sec)	3600

∗ S: Source IoT Device.
∗ I: Intermediary IoT Device.
∗ (*prevRRB*): As shown in line 1, the value for RRB in an original smart contract is 100 tokens. When an intermediary IoT device receives an original smart contract, it sets the RRB value of its own intermediary smart contract to the value of RRB value of the received original smart contract minus 10, as 100 − 10 = 90. This trend continues for next intermediary smart contract; that is, if the previous contract is an intermediary smart contract, the RRB value would be (*previousRRB*) − 10.
∗ (*prevRRE*): As shown in line 2, the value for RRE in an original smart contract is 800 seconds. When an intermediary IoT device receives an original smart contract, it sets the RRE value of its own intermediary smart contract to the value of RRE value of the received original smart contract minus 150, as 800 − 150 = 650. This trend continues for next intermediary smart contract; that is, if the previous contract is an intermediary smart contract, the RRE value would be (*previousRRE*) − 150.

the reception of the first valid route offer from an intermediary device. This is calculated only for the contracts of data packets successfully received by the gateway:

$$RAL = \frac{\sum_{i \in S} \left(T_{i,\text{rep}} - T_{i,\text{req}} \right)}{|S|}, \tag{4}$$

where S is the set of successful smart contracts, $T_{i,\text{req}}$ is the time at which a contract is generated to request a route for data packet i, $T_{i,\text{rep}}$ is the time at which the first valid route offer for data packet i is received by the source IoT device, and $|S|$ is the size of set S.

We conduct the simulations/numerical experiments for a network topology with 14 devices, as shown in Figure 3. The source device has three possible paths to reach the data gateway (destination device). The devices 8, 3, and 4 are the first, second, and third malicious devices, respectively.

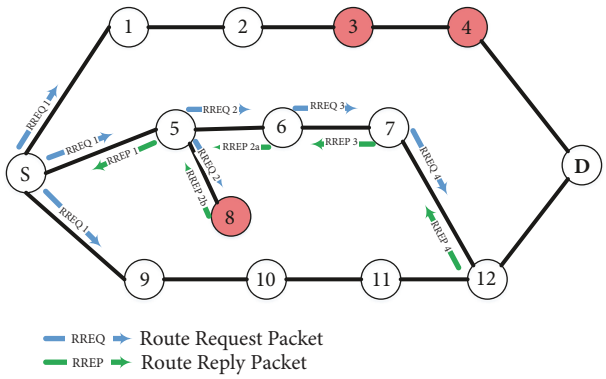

FIGURE 3: The route establishment process in BCR. The source and destination nodes are labeled S and D, respectively.

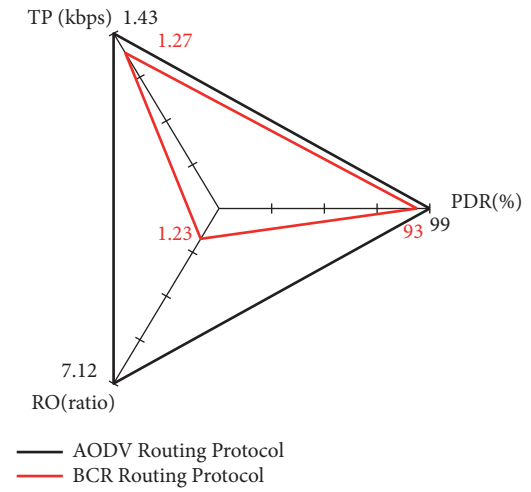

FIGURE 4: A comparison of the BCR and AODV routing protocols based on PDR, TP, and RO performance.

The departure of data packets at the source device follows a Poisson process with an average packet interarrival time of 5 seconds.

6.2. Simulation Results. In this section, we compare the performance of the BCR protocol with that of the AODV routing protocol. We also assess the performance degradation of the BCR protocol in the presence of Blackhole and Greyhole attacks. In a Blackhole attack, the malicious device replies to the route request smart contracts by offering wrong routes in order to disturb the network. In Greyhole attacks, the malicious device passes or drops each data packet with probability 0.5. The malicious device aims to confuse its neighbors as to whether it is malicious or not.

(1) Comparison of BCR and AODV routing protocols: We evaluate the performance of AODV using ns-3 simulator. The ns-3 is an open source software providing a discrete-event network simulator for Internet research and educational use [35]. The ns-3 complies to the technical norms of standard organizations for emerging networks like 3GPP, IEEE, and Wi-Fi Alliance. This is the main reason we choose ns-3 as a prototyping tool for the performance analysis presented in this paper. We obtain the simulation results using the same data traffic and network topology as for BCR.

Figure 4 shows a comparison of the BCR and AODV routing protocols. The BCR protocol has a lower PDR (93%) than AODV (99%). The TP of the BCR protocol is 1.27 kbps which is 9% less than the AODV TP of 1.43 kbps. However, AODV incurs much higher RO ratio (7.12) than that of the proposed routing protocol (1.2). This is because, unlike AODV, our proposed routing protocol does not require IoT devices to start route establishment processes for sending each packet.

(2) Blackhole and Greyhole attacks: Figures 5–8 show the PDR, TP, RO, and RAL for BCR as a function of the number, N, of malicious nodes in the absence of attacks (i.e., $N = 0$) and in the presence of Blackhole and Greyhole attacks (i.e., $N \geqslant 1$).

Figure 5 shows the PDR of BCR for Blackhole and Greyhole attacks. It can be seen that the BCR protocol is less vulnerable to Blackhole attacks than to Greyhole attacks. This is due to the unpredictable behaviour of the Greyhole.

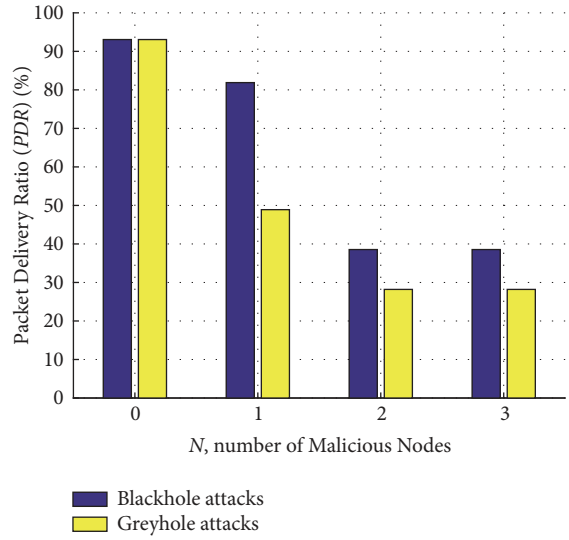

FIGURE 5: PDR of the BCR protocol in the absence of any attacks ($N = 0$) and in the presence of Blackhole and Greyhole attacks ($N \geqslant 1$).

Figure 6 shows that the TP of BCR for different number of malicious nodes N. When $N = 3$, the TP decreases to almost one third of its value at $N = 0$. This is due to the presence of the malicious devices on two of the three possible paths from the source to the destination. This shows that BCR can complete the route establishment phase successfully without a CA.

Figure 7 shows the RO of BCR. The RO increases from 32% when there is no attack (i.e., $N = 0$) to 69% for Greyhole attacks with $N = 3$.

Figure 8 shows the RAL (in Block times) of BCR protocol. It can be seen that a route to the gateway is found in 5.5 Block times where there is no malicious device (i.e., $N = 0$). The RAL increases to 6.9 Block times when the network is under Greyhole attack by N=3 malicious nodes. The actual latency (in seconds) can be reduced by shortening the Block time

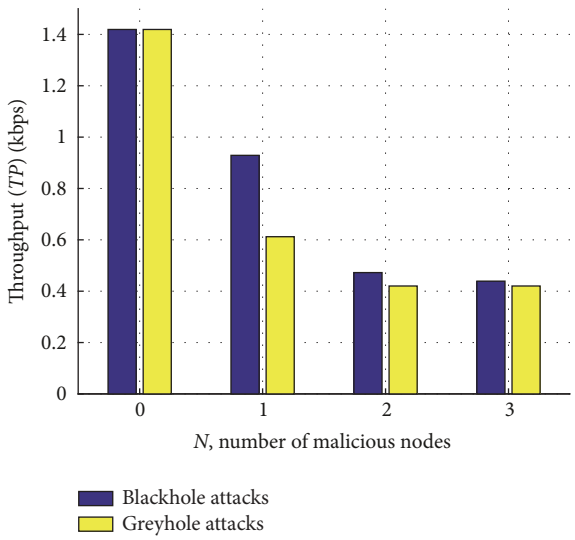

Figure 6: TP of the BCR protocol in the absence of any attacks ($N = 0$) and in the presence of Blackhole and Greyhole attacks ($N \geqslant 1$).

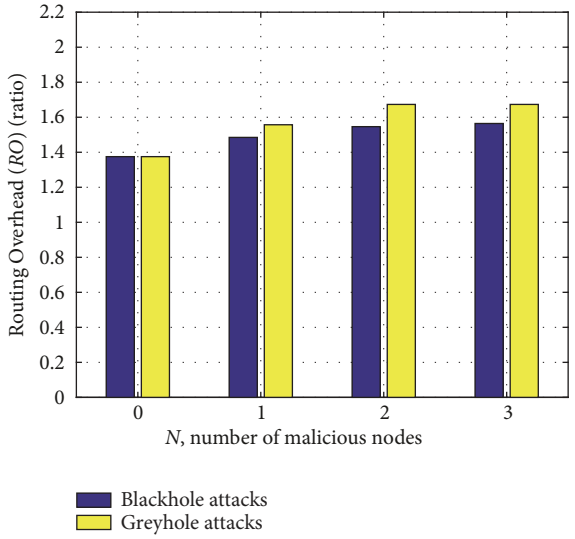

Figure 7: RO of the BCR protocol in the absence of any attacks ($N = 0$) and in the presence of Blackhole and Greyhole attacks ($N \geqslant 1$).

using other blockchain technologies such as EOS blockchain. With the Ethereum Block time of 14 seconds, the BCR protocol can be used only for delay-tolerant applications.

The EOS blockchain is a smart contract platform which is an alternative to the Ethereum blockchain. EOS uses a delegated proof of stake (DPoS) consensus algorithm in contrast to the energy-consuming PoW consensus mechanism used in Ethereum. Moreover, EOS can process 1,000-6,000 transactions per second while Ethereum can process only 15 transactions per second [29, 36]. These features make EOS more suitable for future development of the BCR protocol.

7. Conclusion

We have proposed a novel blockchain-based routing protocol for IoT networks, referred to as BCR, which can be

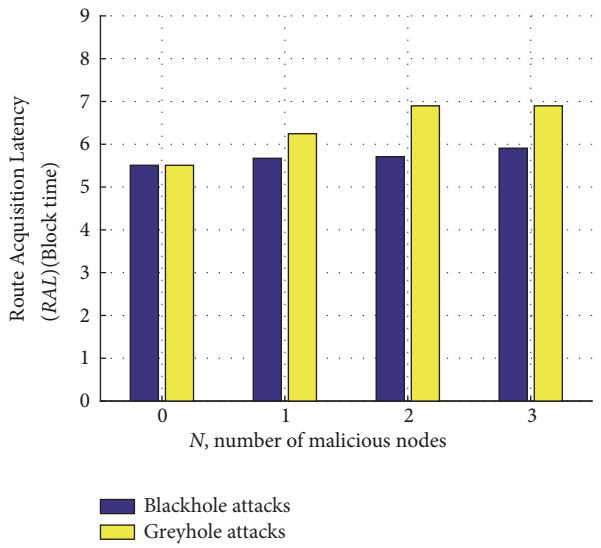

Figure 8: RAL of the BCR protocol in the absence of any attacks ($N = 0$) and in the presence of Blackhole and Greyhole attacks ($N \geqslant 1$).

enabled within a network of untrusted IoT devices. IoT devices can relay one another's data packets to gateways in a decentralized manner. The proposed BCR protocol does not require a central authority to authorize, add, or remove IoT devices, or a secret key sharing mechanism as required by traditional centralized routing protocols. We evaluated the performance of our proposed protocol compared to the AODV using extensive experiments. Our results show that the BCR reduces the routing overhead by a factor of 5 compared to the AODV. It is also resistant to Greyhole and Blackhole attacks. The proposed routing protocol can also be applied to *ad-hoc* networks.

Data Availability

The simulation data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada under Grant RGPIN 1731-2013 and by the UBC PMC-Sierra Professorship in Networking and Communications.

References

[1] G. Glissa, A. Rachedi, and A. Meddeb, "A secure routing protocol based on RPL for internet of things," in *Proceedings of the 59th IEEE Global Communications Conference, GLOBECOM 2016*, pp. 1–7, USA, December 2016.

[2] M. Bouaziz and A. Rachedi, "A survey on mobility management protocols in Wireless Sensor Networks based on 6LoWPAN technology," *Computer Communications*, vol. 74, pp. 3–15, 2016.

[3] "IOTA Whitepaper," 2018, http://iotatoken.com/IOTA_Whitepaper.pdf=0pt.

[4] D. Airehrour, J. Gutierrez, and S. K. Ray, "Secure routing for internet of things: A survey," *Journal of Network and Computer Applications*, vol. 66, pp. 198–213, 2016.

[5] J. W. Hui, "RFC 6553 - The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams," https://tools.ietf.org/html/rfc6553=0pt.

[6] C. Deepa and B. Latha, "HHSRP: A cluster based hybrid hierarchical secure routing protocol for wireless sensor networks," *Cluster Computing*, pp. 1–17, 2017.

[7] P. L. R. Chze and K. S. Leong, "A secure multi-hop routing for IoT communication," in *Proceedings of the 2014 IEEE World Forum on Internet of Things, WF-IoT 2014*, pp. 428–432, Republic of Korea, March 2014.

[8] J. Duan, D. Yang, H. Zhu, S. Zhang, and J. Zhao, "TSRF: A Trust-aware Secure Routing Framework in Wireless Sensor Networks," *International Journal of Distributed Sensor Networks*, vol. 10, no. 1, pp. 1–14, 2014.

[9] X. Anita, J. Martin Leo Manickam, and M. A. Bhagyaveni, "Two-way acknowledgment-based trust framework for wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 9, no. 5, Article ID 952905, pp. 1–14, 2013.

[10] "A Foundation for Improved Protection and Automation," 2018, https://www.cisco.com/c/dam/en_us/solutions/industries/energy/downloads/bc-hydro-cisco.pdf.

[11] N. R. Yerneni and A. K. Sarje, "Secure AODV protocol to mitigate Black hole attack in Mobile Ad hoc," in *Proceedings of the 2012 3rd International Conference on Computing, Communication and Networking Technologies, ICCCNT 2012*, pp. 1–5, India, July 2012.

[12] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks," *Wireless Networks*, vol. 11, no. 1-2, pp. 21–38, 2005.

[13] J. Zhou and J. Cao, "OSR: Optimal and secure routing protocol in multi-hop wireless networks," in *Proceedings of the 32nd IEEE International Conference on Distributed Computing Systems Workshops, ICDCSW 2012*, pp. 187–193, China, June 2012.

[14] A. Jain and B. Buksh, "Solutions for secure routing in mobile ad hoc network (MANET): A survey," *Imperial Journal of Interdisciplinary Research*, vol. 2, no. 4, pp. 5–8, 2016.

[15] M. Kassim, R. A. Rahman, and R. Mustapha, "Mobile ad hoc network (MANET) routing protocols comparison for wireless sensor network," in *Proceedings of the 2011 IEEE International Conference on System Engineering and Technology, ICSET 2011*, pp. 148–152, Malaysia, June 2011.

[16] S. Boora, Y. Kumar, and B. Kochar, "A survey on security issues in mobile ad-hoc networks," *IJCSMS International Journal of Computer Science and Management Studies*, pp. 129–137, 2011.

[17] L. Anderegg and S. Eidenbenz, "Ad hoc-VCG: a truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents," in *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MobiCom '03)*, pp. 245–259, ACM Press, New York, NY, USA, September 2003.

[18] S. Zhong, J. Chen, and Y. R. Yang, "Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks," in *Proceedings of the Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Society (INFOCOM 2003)*, vol. 3, pp. 1987–1997, San Francisco, CA, USA, 2003.

[19] Y.-C. Hu and A. Perrig, "A survey of secure wireless ad hoc routing," *IEEE Security & Privacy*, vol. 2, no. 3, pp. 28–39, 2004.

[20] C. Perkins, E. Belding-Royer, S. Das et al., "RFC 3561- Ad hoc On-demand Distance Vector (AODV) Routing," *Internet RFCs*, pp. 1–38, 2003.

[21] Q. He, D. Wu, and P. Khosla, "SORI: A secure and objective reputation-based incentive scheme for ad-hoc networks," in *Proceedings of the 2004 IEEE Wireless Communications and Networking Conference, WCNC 2004*, pp. 825–830, USA, March 2004.

[22] B. David, R. Dowsley, and M. Larangeira, "MARS: Monetized Ad-hoc Routing System (A Position Paper)," in *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, pp. 82–86, Munich, Germany, June 2018.

[23] H.-Y. Huang and M. Bashir, "The onion router: Understanding a privacy enhancing technology community," in *Proceedings of the 79th ASIS&T Annual Meeting: Creating Knowledge, Enhancing Lives through Information & Technology*, p. 34, 2016.

[24] A. Biryukov and I. Pustogarov, "Proof-of-work as anonymous micropayment: Rewarding a Tor relay," in *Proceedings of the International Conference on Financial Cryptography and Data Security*, vol. 8975 of *Lecture Notes in Computer Science*, pp. 445–455, Springer, Heidelberg, 2015.

[25] R. Ananthapadmanabha, B. S. Manoj, and C. Siva Ram Murthy, "Multi-hop cellular networks: The architecture and routing protocols," in *Proceedings of the 12th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2001)*, vol. 2, USA, 2001.

[26] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Proceedings of the Annual International Cryptology Conference (CRYPTO'17)*, vol. 10401 of *Lecture Notes in Computer Science*, pp. 357–388, Springer, Cham, 2017.

[27] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: analysis and applications," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 281–310, 2015.

[28] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *Proceedings of the International Conference on Financial Cryptography and Data Security*, vol. 8437 of *Lecture Notes in Computer Science*, pp. 436–454, Springer Berlin Heidelberg, 2014.

[29] V. Buterin, *Ethereum: A Next-generation Smart Contract and Decentralized Application Platform*, 2014, https://github.com/ethereum/wiki/wiki/%5BEnglish%5D-White-Paper.

[30] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts," in *Proceedings of the 2016 IEEE Symposium on Security and Privacy, SP 2016*, pp. 839–858, USA, May 2016.

[31] E. Androulaki, A. Barger, V. Bortnikov et al., "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proceedings of the the Thirteenth EuroSys Conference (EuroSys '18)*, pp. 1–15, Porto, Portugal, April 2018.

[32] N. Szabo, "Formalizing and securing relationships on public networks," *First Monday*, vol. 2, no. 9, 1997, http://ojphi.org/ojs/index.php/fm/article/view/548=0pt.

[33] R. Hinden and S. Deering, "RFC 4291 - IP Version 6 Addressing Architecture," pp. 13–15, 2006, https://tools.ietf.org/html/rfc4291=0pt.

[34] R. Kaur and P. Singh, "Black hole and greyhole attack in wireless mesh network," *American Journal of Engineering Research (AJER)*, vol. 3, no. 10, pp. 41–47, 2014.

[35] NS-3 Project, "NS-3 - Network Simulator - Tutorial - Release 3.29," https://www.nsnam.org/docs/release/3.29/tutorial/ns-3-tutorial.pdf.

[36] "EOS.IO Technical White Paper v2," 2018, https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md.