International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 05 | May-2018 www.irjet.net p-ISSN: 2395-0072

Security in IOT Patching

Mrs. Bharathi R[1], Tanuja S [2], Tanushree S[3]

[1]Associate Professor, Dept. of CSE, BMSIT&M, Bangalore-560064, India,
[2,3]B.E Student, Dept. of CSE, BMSIT&M, Bangalore560064, India,

---

Abstract - *In the entangled settings of IoT (Internet of Things) conditions, keeping various heterogeneous gadgets updated is a hectic work, particularly with regard to adequately finding target gadgets and quickly conveying the product updates. In this paper, we change over the conventional programming updating procedure to a dispersed administration. We set a motivator framework for loyally transporting the patches to the beneficiary gadgets. The motivator framework spurs autonomous, self-intrigued transporters for helping the gadgets to be refreshed. To guarantee the framework accurately works, we utilize the blockchain framework that authorizes the dedication in a decentralized way. We likewise display a detail for the proposed convention and approve it by show checking and reproductions for accuracy.*

***Key Words*: Software update, Decentralized system, Recipient, Patch, PATCHTRANSPORTER, Blockchain.**

1. INTRODUCTION

For quite a long while now, the quantity of things associated with the Internet—including telephones, brilliant watches, wellness trackers, home indoor regulators, and different sensors—has surpassed the human populace. By 2020, there will be several billions of such devices on the web. The prospering size of the Internet of Things mirrors the speediest monetary development at any point experienced for any division in the historical backdrop of human progress.

Our computerized frameworks are defenseless against vindictive programmers endeavoring to increase unapproved get to take individual information what's more, other data, hold the data they take for recover, and notwithstanding cut frameworks down totally, as occurred with the assault on Dyn. The result is a progressing weapons contest amongst programmers and PC security specialists, constraining whatever remains of us to live on a treadmill of security updates to the product we keep running on our different PCs.

The IoT gadgets are fundamentally, little PCs. Hence, for administration and security reasons, they should be routinely updated. Makers and specialist co-ops convey programming refreshes called 'patches', to the gadgets (from a little paired alteration to a whole firmware). Regardless of the significance of applying patches to the gadgets, it is quite hard to keep a huge number of gadgets in an avant-garde state [1] by two issues: to start with, Over-The-Air (OTA) conveyance of programming updates to various nodes by means of multi- hop wireless channels is agonizing. The execution of the OTA update is immediately corrupted as the quantity of nodes increments [2]. Second, the assault surface including all the handing-off hubs and beneficiaries is excessively vast to secure all.

2. LITERATURE SURVEY

To conquer the two issues, there have been endeavors on programming updates in the IoT situations. Dulug [4] centers on the powerful OTA conveyance. It dependably exchanges the product updates by means of remote channels. This dependable conveyance of programming update is stretched out by portable sources (initiators) in [5] and by distributed dissemination in [6]. Concentrating on the second issue, Lee et al [7] as of lately displayed CodeDog in-network monitoring of programming update conveyance with semantic fingerprints of payload. Cheng et al. in [8] proposed a method called as traffic aware patching in which we get to know the optimal node needed to be updated to limit the engendering of attacks. Moreover, a few open source ventures are being created for programming (software) updates. Two tasks, swupdate [9] and RAUC [10], give a product update for embedded system and main goal is to increase dependability and to see the fail-safe installation .On the opposite side, Mender [11] and Resinhup [12] exhibit server and client for the OTA update.

Nonetheless, because of the huge populace and the assortment of administrations in IoT condition, finding targets and conveying patches are as yet difficult. To adequately keep up the product update benefit, a devote administration server is required. It ought to be very strong to the power what's more, organize misfortune, and have the capacity to monitor a rundown of each gadget [13]. Be that as it may, for little makers, it is hard and exorbitant to keep up such exceptionally accessible and productive update administrations. Also, an IoT gadget comprises of programming segments from numerous suppliers. On the off chance that we require to earnestly update a normally utilized part, e.g., openssl, the objectives would be various gadgets from the various merchants.

The trouble in dealing with a product update causes real world problems. Late examinations demonstrate that an enormous number of gadgets are as yet working with no better than average insurance from cyber-attacks [14]. The out-dated (or never-updated) gadgets wind up potential dangers. The harm from the cyber-attacks by compromise IoT gadgets, such as Mirai Botnet [15], is massively expanding. Assailants rapidly focus on the gadgets, having

frail login certifications or known vulnerabilities, by fast looking and examining (e.g., through Shodan) and perform massive attacks [16]. In any case, more than 200,000 gadgets in 2017 still required the security patch for the Heartbleed attack that had happened in 2014 [17]. Moreover, the software update process itself additionally should be painstakingly overseen. In August 2017, erroneously updating confused gadgets caused the failing of several IoT shrewd locks utilized for the Airbnb service [18].

In this way, for high accessibility of software update and fast patch delivery, we apply a disseminated benefit way to deal with the product update process. Specifically, we are incidentally propelled by the productivity of attackers, who are self-intrigued and distributed. In the disseminated benefit for programming update, a free laborer, called a 'transporter,' brings the product patch and conveys it to a legitimate gadget. The transporter wins a charge if the patch is satisfactory. This circulated approach for programming update has benefits in accessibility and proficiency. It is likewise valuable to each member. Since a gadget pays the expense for the principal conveyance of patch, we can energize the fast conveyance of fixes by contending transporters. The suppliers don't have to keep up an expensive administration server what's more, the clients can securely utilize the gadget in the up to date structure.

To understand this appropriated approach, a strong reward framework ought to be ensured for supporting reasonable trade of the incentive and the fix. The reward frameworks on disseminated laborers have been developed as Internet administrations, e.g., Amazon Mechanical Turk [19]. The reasonable trade, which is customarily called as the 'delivery versus payment' issue, is dealt with by convoluted lawful exposition or then again escrow administrations of the assistance of a go between [20,21]. At the point when a purchaser and a vender can't concur on settling a trade, the middle person explores the debate and selects a champ, who will get the support. As of late, the cryptographic money frameworks in light of blockchain, Bitcoin and Ethereum, are utilized to execute the dependable arbiter [22,23] in an appropriated way. The blockchain framework helps the reasonable trade to be robotized and reliable, since the savvy contract over blockchain can be filling in as an implementer of the predefined rules for reasonable trade.

Nonetheless, building a strong reward framework utilizing blockchain still has troublesome assignments. A few analysts [24– 28] are as of now announcing the security worries in the blockchain framework. Specifically, the assaults on the keen contracts [27] can acquire the immense financial harm as we could find in the DAO Hack [29], and Parity Multisig Wallet Hack [30]. Along these lines, the reward framework ought to shield itself from the blockchain-particular attacks. Besides, to counteract conceivable vulnerabilities in executing the fix conveying framework on blockchain, the reward framework ought to be composed against the reasonable security worries with regards to blockchain. Nonetheless, the past reasonable trade techniques [22,23] don't consider the assaults utilizing the low-level vulnerabilities in smart contracts,e.g., misused special cases, gas utilization, and manufactured opcodes [27,28].

To aggregate up, the circulated programming refresh administration can empower versatile programming refresh for IoT, however it can be acknowledged just if a basically secure and reasonable reward framework bolsters the procedure. Along these lines, in this paper, we propose PATCH TRANSPORTER, a disseminated, boosted patch conveyance approach for the IoT conditions. We outline a disseminated programming update benefit in view of the strong reward engineering with blockchain elements. Also, to guarantee the reasonableness for each member, every member is ensured with self-checking forms, for example, the receipt and bundle validation. We characterize the parts of suppliers, transporters, and beneficiaries in conveying patches. A supplier gets ready encoded fix bundles for gadgets. A self-intrigued transporter accumulates the fix bundles, finds the objective gadgets for conveyance. Once the bundle is effectively conveyed, the beneficiary gadget makes a savvy contract as a receipt, which contains the conveyance charge for transporter and its reclamation condition. In the event that the transporter can likewise approve the receipt, it uncovers the last decoding key, which can open the fix payload, to guarantee the expense. We characterize three properties to dependably accomplish the objectives of all members. To keep the properties, we likewise draw conceivable domain specific attacks on the blockchain compensate framework for conveying patches. Through model checking and simulation, we check that the proposed framework effectively fulfills the properties and prevents the attack

This paper makes the accompanying commitments:

1. We characterize the properties for the circulated programming update administrations and draw reasonable area particular vulnerabilities for blockchain remunerate framework.

2. We propose a boosted, dispersed fix conveyance framework, which ensures the reasonable trade and is secure against the blockchain related assaults.

3. We check the accuracy of the proposed framework and the wellbeing against the attacks through model checking and reproductions.

3. METHODOLOGY

3.1 Blockchain and Smart Contracts

Blockchain

A blockchain is presented as the infrastructure of Bitcoin [31] to empower a dispersed, verifiable ledger. At first place, Bitcoin uses blockchain just to build its transaction structures on its highest point, in any case, now the utilization of blockchain isn't restricted to the digital

currency system. An extensive variety of businesses are currently accepting blockchain to enhance their frameworks from budgetary organizations [32] and authorizing reports to mechanizing installments and association. Specifically, Bitcoin [31] and Ethereum [33] are illustrative cryptographic money frameworks based over the blockchain. In this way, we accept on both frameworks in the accompanying clarifications on blockchain.

Address and transaction: The essential components of blockchain framework are address and transaction. The address speaks to a record, which stores the cash (cryptographic money). While making an address, a user produces an public key and a private key pair. The public key is changed into an address and the private key is utilized to sign transactions that transfer the money in address to other address. In this way, just the proprietor of the private key can make an exchange that moves the cash in the deliver to other addresses. An exchange fundamentally speaks to the development of the cash from its contributions to its yields. Theoretically, the input and the output can be dealt with like locations, at the same time, more particularly, the output for the most part is a condition on an address and the condition implies that whoever can demonstrate the possession of the address can utilize this cash. In this manner, the output is associated with another output of the accompanying exchange that demonstrates the possession, e.g., signing by the private key. That is, each exchange ought to be confirmed or authenticated. Henceforth, the adjust of an address is the aggregate of accessible yields, which is called 'Unspent Transaction Output (UTXO)'.

Block and blockchain:

The blockchain framework comprises of distributed nodes. Every node is essentially synchronized by means of peer-to-peer correspondences. At the point when a transaction is distributed to a node, it is proliferated to different nodes. A node that completely takes an interest in the blockchain framework called a 'miner'. The miner node approves every one of the exchanges and manufactures a 'block' from the recently validated transactions. A block comprises of the transactions, the hash of the past block, and a proof of its legitimacy, which is utilized as a part of accord calculations. Users can allude to [34] on basic instruments for building the blockchain framework.

Smart Contracts

Other than the addresses and transactions, the smart contract adds programming constructs to the blockchain frameworks. When all is said in done, a small contract is a program code that can be executed in the blockchain and activated by transactions. In this manner, as of late, the closeness between the small contracts furthermore, the customary appropriated objects was investigated in [35]. Each blockchain framework has diverse small contract models. In Bitcoin, the small contract is actualized as content at the information and yield of a transaction. Fundamentally, most Bitcoin exchanges utilize traditional contents to demonstrate the legitimacy of the new exchange, and, all the more particularly, to demonstrate the responsibility for private key comparing to the target UTXO [36]. Notwithstanding, by adding more conditions to the content with utilizing different parameters, e.g., locktime, the Bitcoin transaction can deal with more refined behaviors [37]. On the other hand, Ethereum has a more summed up small contract demonstrate than Bitcoin. The small contracts in Ethereum are executed over the Ethereum Virtual Machine (EVM), which dwells in each miner. In Ethereum, a brilliant contract code is modified in the area particular programming dialects, for example, Solidity [38], Vyper [39], Serpent [40], and so forth., and assembled to an EVM bytecode.

3.2 Blockchain and IoT

The key difficulties in acknowledging secure administrations with IoT gadgets are connected to the confined execution of the gadgets. So as to bring down the cost, the IoT gadgets have low-controlled handling units and no security co-processors, for example, the Trusted Platform Module (TPM). Hence, the main restricted utilization of asset costly cryptographic activities is permitted. Besides, since the IoT gadgets might be battery-fueled and have long-rest cycles, the convention with various message trades isn't appropriate.

As one of the handy arrangements, the IoT organize externalizes security highlights, and the blockchain is all around coordinated for the reason. Specifically, the blockchain-based administrations for IoT is worthwhile to unravel the ordinary difficulties of IoT security, for example, capacity constraints, inadequate architecture, inaccessibility of administrations, manipulation susceptibility, and so on .Specifically, the small contract helps the IoT gadgets to appoint the security capacities by sending a smart contract as a self-ruling specialist in the blockchain

4. SYSTEM DESIGN

Distributed Software Update Service

We propose PATCHTRANSPORTER, a boosted fix delivery approach in light of the blockchain framework. The model for software updates can be ordered as in Figure 1. Initial, a solitary supplier is operated for software update management and it is responsible for updating every one of the gadgets (Figure 1a). At the point when a gadget has various software segments, different suppliers handle the product updates to numerous gadgets in the meantime (Figure 1b). In the two cases, a supplier needs to deliberately deal with its own particular administration server for programming refreshes with high accessibility and versatility, which is challenging for small manufacturers. In this way, our approach is to give distributed services of software update for IoT situations (Figure 1c). We initially decouple patch generation and conveyance. As a conveyed software update benefit, the self-intrigued specialist, called 'transporter', gets the patch and safely conveys it to gadgets

distributed way. We likewise utilize the blockchain framework to administer the members to take after the standards for conveying patches at the core. A member can accomplish its objective, i.e., a patch or a fee, just when it acts legitimately. With a strong reward system, the transporters competitively discover targets and convey the patch, in this way the productivity of fix conveyance can be amplified.

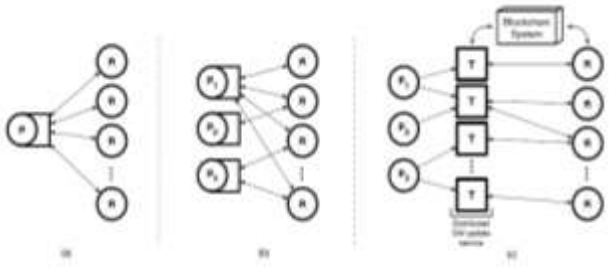

Figure 1. Models of software update: (a) the one-to-many model; (b) the many-to-many model (multiple providers or vendors); (c) the proposed distributed software update model, where *P* is the provider or manufacturer, *R* is the recipient device, and *T* is the patch transporters.

We characterize the parts of members to portray our approach. A 'provider' readies a product fix as a patch package, which is a deliverable unit of a fix and doubly encrypted. A 'transporter' recognizes the target of a patch packages and after that conveys to a target 'recipient'. On the off chance that the patch package is appropriately delivered, the recipient pays an expense to the transporter. Figure 2 demonstrates the general procedure of patch transporter.

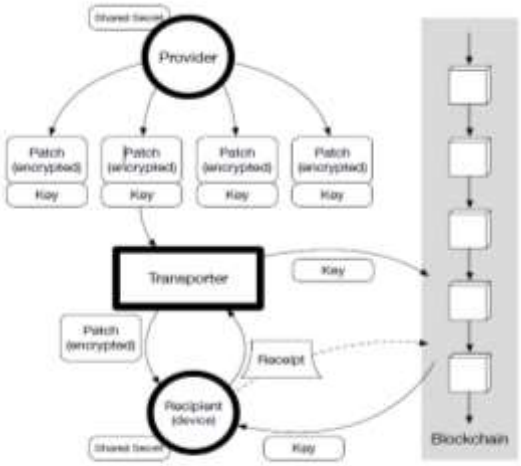

Figure 2. PATCHTRANSPORTER process.

We characterize the accompanying properties to guarantee the delivery system is accurately working.

Definition 1 (Faithful Delivery)**.** If a transporter picks up a patch package from a provider and a target of the package is identified, the transporter should deliver the package without any modification.

Definition 2 (Authenticated Origin)**.** A recipient should be able to authenticate the origin of the delivered patch package.

Keeping in mind the end goal to fulfill the properties, we encrypt the patch package with the key s, which is a reshaped key just between the supplier and the receiver. On the off chance that a device comprises of software components from numerous vendors, it is might also have set of s. The integrity of patch package, including a product fix and metadata, is additionally protected by a keyed hash with s. At last, we additionally characterize the property for the fairness of the patch delivering system.

Definition 3 (Fair Reward)**.** A transporter should be paid by a predefined fee for every faithful delivery of a unique patch package, whose origin is authenticated.

We utilize blockchain to ensure Fair Reward. For this reason, the patch payload in the i-th patch package is doubly encoded by key_i^E and additionally s. The transporter hides its decoding key key_i^D from the beneficiary while conveying the patch package. The recipient checks the respectability of the conveyed fix package and confirms the origin with s. In the event that it is legitimate, it issues a receipt implanting a smart contract code and a charge. The smart contract code of the receipt is introduced with the hashed value of key_i^D, and it is set to transfer the expense to who uncovers key_i^D. Thus, the transporter can get the charge by sending an exchange with public uncovering k_i , so the recipient can learn it at the same time. That is, the receipt sent in blockchain empowers the atomic exchange of k_i and an expense between the transporter and the recipient.

Attack model: In this paper, we concentrate more on the domain-specific attacks that can be utilized in the real time implementation with the blockchain framework than the assaults of communication itself, e.g., jamming, since the security against the generic network system attacks has just been investigated in various research papers. In our attack model, we expect that any member can perform malignant practices to satisfy their objectives at the limited cost. On the off chance that the transporter is vindictive, it can fashion the conveyance bundle keeping in mind the end goal to get the expense without conveying a legitimate package or to devour the figuring asset of the recipient. It can likewise perform relay assaults by sending a similar package more than once. In the event that the supplier and the beneficiary are pernicious, they will likely give the transporter a chance to convey the bundle without paying the expense. Hence, we assume that the vindictive provider and beneficiary can trade a small message hidden from the transporter. Moreover, an outside aggressor can watch the procedure in the wireless channel and the blockchain to catch the expense between the recipient and the transporter. In light of this attack model, we broke down the proposed procedure with the vulnerabilities determined in the writing of escrow convention [23], blockchain [24– 26], and savvy contract [27,28], with a specific end goal to distinguish the possible attacks.

i) Malformed Receipt.

In enacting the receipt on the blockchain, two stages are required. To begin with, the transporter or the recipient conveys the receipt on the blockchain system. Second, the transporter asserts the expense by uncovering the description key, key^{D}. In any case, the recipient makes the brilliant contract code in the receipt, and the transporter can't adjust it since it is embodied as a signed transaction. Therefore, if a beneficiary is malevolent, it can betray the transporter by giving a deformed smart contract code in the receipt.

– Revealing key with exception.

The transporter plays out the second step by distributing a recover exchange that uncovers the decryption key to claim the charge. On the off chance that a malicious recipient infuses an illicit instruction to the smart contract code of the receipt, the reclaim transaction of the transporter is done as a failure (and return every one of the progressions). Be that as it may, the transaction with the key is now transmitted [27]. The malignant recipient can take in the key from the transaction without paying any charge.

– Gas-consuming receipts

This is an Ethereum-particular issue. When executing a shrewd contract in Ethereum, the guest should pay the cost of execution, called 'gas.' More particularly, each low-level guideline has its own particular gas cost contingent upon the effect on registering control/stockpiling of blockchain (For instance, the gas cost of ADD is 3 yet that of CREATE, which makes another record, is 32,000 as per.). On the off chance that PATCHTRANSPORTER is actualized on the Ethereum framework, the transporter pays the gas expenses to execute the savvy contract code in the receipt. Notwithstanding, a malevolent beneficiary can infuse the gas-expensive tasks in the receipt and influence the transporter to pay the high expenses in reclaiming the key. (Late research [28] demonstrates seven gas-expensive examples in the brilliant contract of Ethereum.) Besides, if the transporter couldn't appropriately expect the cost of the gas-exorbitant activities, the execution of the reclaim exchange is prematurely ended with uncovering the key due to the lacking gas special case or the abundance of the square gas constrain.

ii) Fee Interception:

Unless the receipt entirely checks the recipient, the consequence of the receipt progresses toward becomes as transaction-ordering dependent [24]. In blockchain, the miner who makes the block can choose the transaction's orders. Hence, if two clashing exchanges happen in the meantime, the champ is reliant on the miner's choice. For our situation, when the transporter sends the exchange to reclaim the key, a hacker can send another exchange with a similar key gained from the transporter's exchange. On the off chance that the miner chooses aggressor's exchange to begin with, the assailant can know the fee.

iii) Transporter Outcast:

Another purpose behind the two fold encryption of the bundle is to make each bundle unique. Thus, if the transporter does not take an interest in the production of fix bundle, at that point the supplier utilizes its own particular key pool. For the situation that the supplier and the beneficiary are cooperatively bargained, the supplier can subtly hand over the other decryption key, keyDi, to the beneficiary. At that point, the transporter conveys the fix bundle however can't get the receipt for the conveyance, since the beneficiary definitely knows all the keys for decrypting the fix, s and keyDi.

iv) Denial of Service Attack on Recipients.

Since the beneficiary is a vitality obliged node, a noxious transporter may send a manufactured bundle to expend the vitality of beneficiary in handling the manufactured bundle. Specifically, when the Key Registration technique is utilized, which will be clarified in further Sections, the transporter creates a couple of public key and private key and registers a key for encryption and a hash estimation of the other key for demonstrating the decoding key. Since the supplier asks for just a key in the plain content form, a noxious transporter may enroll a key and a hash estimation of a wrong key. For this situation, the malignant transporter may unlawfully take the charge while the beneficiary expends the vitality to perform open key decryption with a wrong key. We outline PATCHTRANSPORTER to fulfill every one of the properties under the thought of these security assaults. Hence, we devise the structure of fix bundle and the receipt, and the receipt evaluation and key enlistment for the transporter. The complete process and the particular of the entire framework are described in the following Sections.

5. IMPLEMENTATION AND RESULTS

We additionally check the impact of the conceivable assaults on PATCHTRANSPORTER through the simulation.

Package modification.

The adjustment on the fix bundle should be possible on the fix payload or metadata. In simulation, we accept the adjustment assault and flipped an arbitrary byte of fix payload and metadata in the transporter, yet every change is accurately identified. Specifically, with the Key Registration strategy, since no member can't completely control the keys, any unnoticeable change on the payload isn't conceivable.

Malformed Receipt.

The distorted receipt is distinguished by the receipt validation. As we said earlier, the transporter initially preforms the structure validation. In the event that it can't discover the reclamation work, proof Of Key, the receipt is rejected. Else, it can have the tracing of the objective code. By contrasting the trace and that of the reference code, the transporter can approve it. Regardless of whether the two follows are not precisely coordinated, it checks whether there are the unforeseen revert or gas-exorbitant guidelines

in the target receipt. Accordingly, we set a vindictive beneficiary, who haphazardly includes revert (or invalid) or gas-exorbitant guidelines while making the receipt. Clearly, with the correct coordinating, each assault can be identified. We could observe that repetitive gas-expensive guidelines are effortlessly recognized since the EVM assembly is moderately basic and the clear dismantle process produces good results. Specifically, since the quantity of gas-expensive guidelines isn't changed with and without enhancement in Solidity 0.4.19, we could utilize the base edge, i.e., 1, for TH_G in recognizing the gas-exorbitant instructions. However, the length of ways to check startling return is reliant on the improvement. The most extreme diverse of way lengths is 13 between the optimized and unoptimized code. In this way, the limit for the way length, THL, ought to be higher than 13 to anticipate false positives caused by the enhancement alternatives from the same code. The recognition rate of unexpected revert is higher as THL is lower, however when THL = 13, we can get 75.7% and 78.7% for enhanced and unoptimized code. The constraint of the receipt approval can be enhanced by muddled investigation approaches, for example, representative execution [24]. In any case, since we accept it is performed on the transporter for each receipt, the exchange off amongst proficiency and rightness ought to be considered.

Fee Interception and Transporter Outcast.

Since we include the recipient of the expense as a contention of the constructor of the receipt, the expense can be exchanged to the enlisted recipient as it were. Hence, despite the fact that an hacker replays the key divulgence at a similar piece time, the recipient is the same. Due to the conveyed smart contract is unchanging; we can confirm that any endeavor to change isn't conceivable, including the proprietor change assault in the Parity Multisig Wallet Hack. What's more, when we utilize the Key Registration strategy, the encoded payload can be unscrambled by the private key, however just the transporter has it. Along these lines, the supplier and the beneficiary can't cooperatively bamboozle the transporter with the expectation of complimentary conveyance.

Denial of Service Attack on Recipients.

A malevolent transporter can fashion a fix bundle of a good format to expend the vitality of the beneficiary while preparing it. In our approach, the significant vitality utilization in the IoT beneficiary occurs in making a receipt and unscrambling bundle, which require public key cryptography activities (when the Key Pool strategy is utilized, the cost for the decoding can be diminished to that of symmetric encryption.). In this way, the one of the assailants' objectives is to mislead the beneficiary to make a false receipt. Be that as it may, the produced bundle can be distinguished in approving the metadata in PATCHTRANSPORTER. Also, regardless of whether the malignant transporter utilizes the hash estimation of a wrong key as portrayed in the earlier Section, it can't breeze through the approval test in the receipt smart contract. Therefore, the beneficiary can realize that the uncovered key isn't right and skirt the superfluous decrypting with the wrong key. Thusly, the vitality costly task is avoidable in patchtransporter.

6. CONCLUSION

In the muddled IoT conditions, safely conveying programming patches is a hard furthermore, basic issue. In this paper, we propose another approach, PATCHTRANSPORTER, for conveying fixes through an impetus framework. To guarantee that the reward framework is reliable, we utilize the blockchain framework as a master of duties between a fix transporter and a beneficiary gadget. We additionally propose the self-checking procedure to shield the members from the space particular vulnerabilities in the blockchain remunerate framework. Besides, we build up the particular to unmistakably portray the proposed framework. Specifically, on the accuracy properties, the model checking with the detail and the reenactment with the genuine blockchain framework demonstrate that the proposed framework can give appropriate motivating force frameworks in decentralized methodologies.

7. REFERENCES

[1]. Mendez, D.M.; Papapanagiotou, I.; Yang, B. Internet of things: Survey on security and privacy. arXiv **2017**,arXiv:1707.01879.

[2]. Wang, Q.; Zhu, Y.; Cheng, L. Reprogramming wireless sensor networks: Challenges and approaches.IEEE Netw. **2006**, 20, 48–55.

[3]. Kim, D.; Nam, H.; Kim, D. Adaptive Code Dissemination Based on Link Quality inWireless Sensor Networks.IEEE Internet Things J. **2017**, 4, 685–695.

[4]. Dutta, P.K.; Hui, J.W.; Chu, D.C.; Culler, D.E. Securing the deluge network programming system.In Proceedings of the 5th International Conference on Information Processing in Sensor Networks, Nashville,TN, USA, 19–21 April 2006; pp. 326–333.

[5]. Hong, S.G.; Kim, N.S.; Heo, T. A smartphone connected software updating framework for IoT devices.In Proceedings of the 2015 IEEE International Symposium on Consumer Electronics (ISCE), Madrid, Spain,24–26 June 2015; pp. 1–2.

[6]. Liu, J.; Tong,W. A Framework for Dynamic Updating of Service Pack in the Internet of Things. In Proceedings of the 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing, Internet of Things (iThings/CPSCom), Dalian, China, 19–22 October 2011; pp. 33–42.

[7]. Lee, J.; Kwon, T. Secure dissemination of software updates for intelligent mobility in future wireless

networks.EURASIP J. Wirel. Commun. Netw. **2016**, 2016, 250.

[8]. Cheng, S.M.; Chen, P.Y.; Lin, C.C.; Hsiao, H.C. Traffic-aware Patching for Cyber Security in Mobile IoT.arXiv **2017**, arXiv:1703.05400.

[9]. Software Update for Embedded Systems. Available online: https://github.com/sbabic/swupdate (accessed on 12 February 2018).

[10]. Safe and sEcure Software Updates for Embedded Linux. Available online: https://github.com/rauc/rauc (accessed on 12 February 2018).

[11]. Mender: Over-the-Air Software Updates for Embedded Linux. Available online: https://mender.io (accessed on 12 February 2018).

[12]. RESINHost os UPdater. Available online: https://github.com/resin-os/resinhup (accessed on 12 February 2018).

[13]. Nguyen, R.; Stenberg, E. Software Update for IoT and the Hidden Cost of Homegrown Updaters.Availableonline:https://mender.io/resources/guides-andwhitepapers/_resources/Mender%2520White%2520Paper%2520_%2520Hidden%2520Costs%2520of%2520Homegrown.pdf (accessed on 12 February 2018).

[14].Quarta, D.; Pogliani, M.; Polino, M.; Maggi, F.; Zanchettin, A.M.; Zanero, S. An Experimental Security Analysis of an Industrial Robot Controller. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–24 May 2017; pp. 268–286.

[15]. Newman, L.H. The Botnet That Broke the Internet Isn't Going Away. 2016. Available online: https://www.wired.com/2016/12/botnet-broke-internet-isnt-going-away/ (accessed on 12 February 2018).

[16]. Yu, T.; Sekar, V.; Seshan, S.; Agarwal, Y.; Xu, C. Handling a trillion (unfixable) flaws on a billion devices:Rethinking network security for the internet-of-things. In Proceedings of the 14th ACMWorkshop on Hot Topics in Networks, Philadelphia, PA, USA, 16–17 November 2015; p. 5.

[17]. Eduard, K. Heartbleed Still Affects 200,000 Devices: Shodan. 2017. Available online: https://www.securityweek.com/heartbleed-still-affects-200000-devices-shodan/ (accessed on 12 February 2018).

[18]. Thomson, I. Firmware Update Blunder Bricks Hundreds of Home 'Smart' Locks. The Register.Availableonline: https://www.theregister.co.uk/2017/08/11/lockstate_bricks_smart_locks_with_dumb_firmware_

upgrade/ (accessed on 12 February 2018).

[19]. Amazon Mechanical Turk. Available online: https://www.mturk.com/ (accessed on 12 February 2018).

[20]. Asokan, N.; Schunter, M.; Waidner, M. Optimistic protocols for fair exchange. In Proceedings of the 4th ACM Conference on Computer and Communications Security, Zurich, Switzerland, 1–4 April 1997; pp. 7–17.

[21]. Bao, F.; Deng, R.H.; Mao, W. Efficient and practical fair exchange protocols with off-line TTP. In Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, USA, 6 May 1998; pp. 77–85.

[22]. Juels, A.; Kosba, A.; Shi, E. The ring of Gyges: Investigating the future of criminal smart contracts.In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security,Vienna, Austria, 24–28 October 2016; pp. 283–295.

[23]. Goldfeder, S.; Bonneau, J.; Gennaro, R.; Narayanan, A. Escrow protocols for cryptocurrencies: How to buy physical goods using Bitcoin. In Proceedings of the 21st International Conference on Financial Cryptography and Data Security, Sliema, Malta, 3–7 April 2017.

[24]. Luu, L.; Chu, D.H.; Olickel, H.; Saxena, P.; Hobor, A. Making smart contracts smarter. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria,24–28 October 2016; pp. 254–269.

[25]. Halpin, H.; Piekarska, M. Introduction to Security and Privacy on the Blockchain. In Proceedings of the IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Paris, France, 26–28 April 2017; pp. 1–3.

[26]. Li, X.; Jiang, P.; Chen, T.; Luo, X.; Wen, Q. A survey on the security of blockchain systems. Future Gener. Comput. Syst. **2017**, doi:10.1016/j.future.2017.08.020.

[27]. Atzei, N.; Bartoletti, M.; Cimoli, T. A Survey of Attacks on Ethereum Smart Contracts (SoK). In Proceedings of the International Conference on Principles of Security and Trust, Uppsala, Sweden, 24–25 April 2017; pp. 164–186.

[28]. Chen, T.; Li, X.; Luo, X.; Zhang, X. Under-optimized smart contracts devour your money. In Proceedings of the 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), Klagenfurt, Austria, 21–24 February 2017; pp. 442–446.

[29]. Daian, P. Analysis of the DAO Exploit. 2016. Available online:
http://hackingdistributed.com/2016/06/18/analysis-of-the-dao-exploit/ (accessed on 12 February 2018).

[30]. Palladino, S. The Parity Wallet Hack Explained. 2017. Available online: https://blog.zeppelin.solutions/onthe-parity-wallet-multisig-hack-405a8c12e8f7 (accessed on 12 February 2018).

[31]. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: http://www.academia.edu/download/32413652/BitCoin_P2P_electronic_cash_system.pdf(accessed on 12 February 2018).

[32]. Kelly, J. Forty Big Banks Test Blockchain-Based Bond Trading System. Available online: http://www.reuters.com/article/banking-blockchain-bonds-idUSL8N16A30H (accessed on 12 February 2018).

[33]. Ethereum Project. Available online: https://ethereum.org (accessed on 12 February 2018).

[34]. Narayanan, A.; Clark, J. Bitcoin's academic pedigree. Commun. ACM **2017**, 60, 36–45.

[35]. Sergey, I.; Hobor, A. A Concurrent Perspective on Smart Contracts. arXiv **2017**, arXiv:1702.05511.

[36]. Antonopoulos, A.M. Mastering Bitcoin: Unlocking Digital Cryptocurrencies; O'Reilly Media, Inc.: Sebastopol,CA, USA, 2014.

[37]. Contract. (Bitcoin Contract). Available online: https://en.bitcoin.it/wiki/Contract (accessed on 12 February 2018).

[38]. The Solidity Contract-Oriented Programming Language. Available online: https://github.com/ethereum/solidity (accessed on 12 February 2018).

[39]. Vyper: New Experimental Programming Language. Available online: https://github.com/ethereum/vyper(accessed on 12 February 2018).

[40]. Serpent. Available online: https://github.com/ethereum/wiki/wiki/Serpent (accessed on 12 February 2018).