applied sciences

MDPI

Article

Vulnerabilities and Limitations of MQTT Protocol Used between IoT Devices

Dan Dinculeană and Xiaochun Cheng *

Faculty of Science and Technology, Middlesex University, London, NW4 4BT, UK; dd693@live.mdx.ac.uk
* Correspondence: x.cheng@mdx.ac.uk; Tel.: +44-(0)-208-411-4979

Received: 31 October 2018; Accepted: 3 February 2019; Published: 27 February 2019

Abstract: With the proliferation of smart devices capable of communicating over a network using different protocols, each year more and more successful attacks are recorded against these, underlining the necessity of developing and implementing mechanisms to protect against such attacks. This paper will review some existing solutions used to secure a communication channel, such as Transport Layer Security or symmetric encryption, as well as provide a novel approach to achieving confidentiality and integrity of messages. The method, called Value-to-Keyed-Hash Message Authentication Code (Value-to-HMAC) mapping, uses signatures to send messages, instead of encryption, by implementing a Keyed-Hash Message Authentication Code generation algorithm. Although robust solutions exist that can be used to secure the communication between devices, this paper considers that not every Internet of Things (IoT) device or network design is able to afford the overhead and drop in performance, or even support such protocols. Therefore, the Value-to-HMAC method was designed to maximize performance while ensuring the messages are only readable by the intended node. The experimental procedure demonstrates how the method will achieve better performance than a symmetric-key encryption algorithm, while ensuring the confidentiality and integrity of information through the use of one mechanism.

Keywords: Internet of Things (IoT); Message Queue Telemetry Transport (MQTT); Keyed-Hash Message Authentication Code (HMAC); confidentiality; integrity

1. Introduction

The proliferation of embedded devices that possess their own memory, processing power, and now also have the ability to communicate with each other, has given birth to new attack vectors that have proven to be difficult to secure against. According to [1], interconnected devices such as sensors, appliances, and cameras represent some of the components of this network, which was named the Internet of Things (IoT) by Kevin Ashton at a conference in 1999. The IoT became possible by the sudden increase in smart devices that manufacturers developed and released on the market. This was accomplished without having properly considered all aspects of security and device limitations. It is assumed that by the year 2020, most of the devices that the consumer will have access to will be able to connect to the Internet. Kevin also stated that most of the information available right now was recorded using different conventional methods (e.g., typing).

This aspect, coupled with the fact that humans in the modern era have a limited amount of time and ability to properly capture different aspects of life, gave birth to the idea that a machine which can record and even process data automatically could be built and programmed. As described in Reference [2], in October 2016, the Dyn DNS company was hit by a distributed denial-of-service attack (DDoS) orchestrated by a botnet. The Mirai malware was stated to be the "primary source of attack traffic". This malware was able to infect a large number of smart devices by using brute force in order to get access to their interfaces and then upload a copy of itself onto these devices. The DDoS was effectively a flood of TCP and UDP packets that were destined for port 53. This denial of service attack and numerous others have made it clear that IoT requires robust protocols to not only ensure the security of the devices themselves, but also to protect the outside world from the potential danger they pose.

2. Security of the Internet of Things

2.1. The Message Queue Telemetry Transport (MQTT) Protocol

As stated in Reference [3], the Message Queue Telemetry Transport (MQTT)is a protocol used for communication within an IoT environment that functions on top of the Transport Control Protocol. The protocol was created by IBM as a machine-to-machine, lightweight communication method. MQTT was standardized by ISO/IEC 20922 and was further accepted as part of OASIS. At its core, MQTT is a messaging protocol that uses the publish-subscribe communication model, where the clients themselves do not require updates, thus in turn causing a reduction of used resources, which makes this model optimal for use in a low-bandwidth environment.

Furthermore, the protocol functions on a server-client system where the server, called a broker, pushes updates to MQTT clients. The clients won't send messages directly to each other, instead relying on the broker for this. Every MQTT message contains a topic, organized in a tree-like structure, to which the clients can subscribe or publish. The broker receives published messages from clients that contain a certain value or command and relays the information to every client that has subscribed to that specific topic. As can be seen, the MQTT protocol was designed for asynchronous communication, where subscriptions or publishing to or from different entities take place in a parallel order. The protocol is also able to provide reliable transfers by choosing between three types of reliability mechanism, also called Quality of Service (QoS).

When compared to other protocols like HTTP, the MQTT protocol has a considerably smaller footprint, making MQTT, as stated above, much more suitable for resource-constrained environments. Although the MQTT protocol has many advantages, not every MQTT-based broker has similar or comparable abilities for entity authentication or encryption. Eclipse's open-source application, called Mosquitto, is able to provide most of standardized features of the MQTT protocol, such as SSL/TLS and client certificate support. The Mosquitto broker, by default, does not provide security for its messaging scheme and authentication information is sent in plaintext; therefore, it requires security mechanisms to protect the transferred information.

2.2. Security Overview of MQTT

As previously mentioned, MQTT features different security mechanisms, but most of them are not configured or provided by default, such as data encryption or entity authentication. Authentication mechanisms, such as using the physical address of the device (MAC), exist and are controlled by the broker by registering a device's information once it tries to connect. Access authorization can be done by the broker using a mechanism called an Access Control List (ACL). The ACL, as the name implies, contains records of information such as the identifiers and passwords of the different clients that are allowed to access different objects and can also specify what functions the client can perform on these.

According to Reference [4], confidentiality is a major requirement of a secure system and can be accomplished at the application layer by encrypting the message that needs to be published. This method of encryption can either be implemented as client-to-broker or end-to-end. In a client-to-broker type of encryption, the broker decrypts the information that is being published to a topic and respectively encrypts the values that it needs to send to other clients. In an end-to-end situation, the broker cannot decrypt the information being published to topics and it forwards the ciphertext to other devices. In the latter method, the broker needs fewer computational resources and less energy as it only functions as a courier and does not require any additional modules that can encrypt/decrypt messages.

Nonetheless, additional security mechanisms can also be implemented on lower layers. According to Reference [3], one way to reliably ensure the security of a communication channel at the transport layer is by using Transport Layer Security protocol (for TCP) or even Datagram Transport Layer Security (in the case of UDP). Additionally, according to [4], encryption at the link layer can be achieved by using one of the many algorithms available, such as Advanced Encryption Standard (AES) in Counter Block Mode or AES in Counter with CBC-MAC mode, also called CCM mode. This type of security mechanism provides some additional advantages compared to other methods, such as increased efficiency due to the hardware acceleration capabilities found on radio chips.

3. Research Setting

As previously discussed, there are many issues and limitations to IoT devices that need to be addressed in order to secure a communication channel between them. Taking into account that these devices are resource-constrained, it might not be easy to develop robust security mechanisms. As an example, TLS protocol could be implemented to secure a communication channel, however, the overhead generated by this could be too much for small, resource-limited devices, or the devices might not even support the protocol.

This paper will focus on implementing the Message Queue Telemetry Transport protocol using the ‘paho-mqtt’ library, deploying Mosquitto as the broker within a network designed with single board computers, will underline the software limitations, and will provide a different approach to achieve confidentiality and integrity of transmitted data. The proof-of-concept script, which is written in Python, will contain several functions necessary to implement, test, and deploy the Value-to-HMAC mapping method as a solution to secure data sent between MQTT clients and the broker.

3.1. IoT Network Design for Experiment

As mentioned, the network was comprised of single board computers connected to a switch on which the main script was deployed. The MQTT clients were coded in Python, using the ‘paho-mqtt’ library which implements the MQTT protocol. The script allows the clients to be configured as either publishers or subscribers. The Onion Omega2+ was chosen as the single board computer due to it having a lower price point compared to a Raspberry Pi device, as well as coming with a pre-installed and lightweight version of Linux Operating System, the Linux Embedded Development Environment based on OpenWRT.

3.2. Value-to-HMAC Mapping

The method was designed as an alternative solution to achieving confidentiality of information while potentially being faster than a symmetric-key encryption algorithm. Because it is based on creating signatures from data, it is also able to provide integrity.

The design of the Value-to-HMAC mapping is based on the idea behind a rainbow table attack, where an attacker is able to retrieve the original password from a hashed value by using large pre-computed tables of hashes. Because hashing algorithms are publicly accessible, anyone would be able to generate their own hash tables if they had knowledge of this implementation; therefore, in order to overcome this obstacle, the method makes use of the Keyed-Hash Message Authentication Code algorithm to generate signatures.

According to Reference [5], HMACs are used to check the integrity and origin of a message by generating a hash from the message and a pre-shared secret key. For an attacker, it would be unfeasible to generate table mappings without knowing the secret key, as this would require generating hashes for every possible combination of values that a system transmits using every possible secret key. Additionally, according to Reference [6], a hash function needs to meet specific security objectives such as preimage resistance, second preimage resistance, and collision resistance. Preimage resistance refers to the one-wayness of a function, where it would be unfeasible to find a string of data that generates a specific digest of a given function. Second preimage resistance refers to the unfeasibility of finding a second string of data that would generate a similar hash for a given message. Finally, the collision resistance property describes the computationally exhausting and time-consuming nature of finding two different messages for which a hash function would generate the same output.

The HMAC mapping method was created to provide a different way of obscuring the contents of data in transit, while being faster and providing similar security benefits to encryption. The method involves using a HMAC function (Figure 1) to create a signature from the source data and a secret key and send the hash to its destination. On the sender's side, the method will generate a Keyed-Hash Message Authentication Code using a secret key and the data that needs to be transmitted. The receiver will need to generate a table to help map the possible values to signatures and will use this table to recover the original data. Therefore, on the receiver side, a table will be generated that contains a pair of values, the data, and a HMAC digest of the value. The receiver would perform a search on the table using the received HMAC and if a match is found, it will then recover the original value.

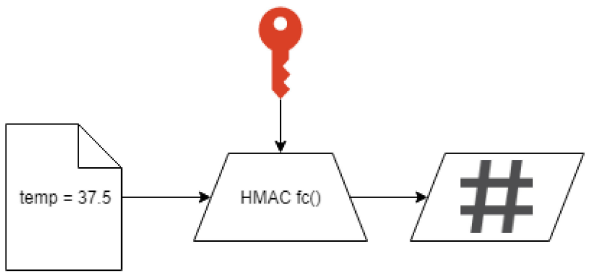

Figure 1. Hash-based Message Authentication Code model.

The main objective that the method needs to achieve is the safe distribution of secret keys to all parties. In order to be secure, different keys need to be created and distributed to clients even if they are subscribing to the same topic. If a node was corrupted by an external attacker, disabling one key used by one client is much more efficient than disabling one key used by many, as it will require the additional generation of new keys in this scenario. The method of mapping a HMAC digest to the original value is able to achieve confidentiality and integrity of data if the secret key is only known by the parties that want to share information, underlining the importance of protecting the key and table mapping file.

4. Research

4.1. Computational Complexity of Blake2

To assess the security of Blake and Blake2 hashing algorithms, the authors of Reference [7] used a boomerang attack and compare their complexity. Towards this goal, the authors proposed using a 7.5 round-based boomerang attack on Blake2s with a complexity of 2^{184}. As described, the attack is able to succeed, having a probability of p^2q^2, and the boomerang attack type used is I, with a complexity of 2^n. Finally, the paper suggests that the complexity of attacking Blake2 is higher than for Blake, as demonstrated.

The authors of [8] presented a novel cryptanalysis method using biclique techniques, which generate results for key recovery on the AES-256 block cipher, having a computational complexity of $2^{254.4}$. Additionally, according to [9], the complexity of an Algebraic attack on a cipher such as AES is caused by the non-linear component of the block cipher. As suggested, the complexity increases whenever the number of monomials will increase.

4.2. Performance Assessment

In order to compare the performance of the HMAC algorithm based on different cryptographic hash functions, timers have been placed before and after the execution of each function to calculate an approximate execution time. Additionally, in order to smooth out the irregularities in time measurements caused by hardware interrupts, each pair of timers recorded a large number of iterations for each function; therefore, the values described in the graphs represent 1000 iterations of each function. The time measurement script was written in Python 3 and has made it possible to take advantage of the most accurate timer functions available within Python. Furthermore, the script makes use of libraries, such as 'hashlib' and 'pycrypto', which have implemented AES and HMAC algorithms, as well different cryptographic hash functions.

The following timings were generated on the resource-constrained device, the Omega2+. Upon execution, it will run the HMAC generation speed test for a number of cryptographic hash functions currently used in different cryptosystems. As shown in the table below, some hash functions are faster than others at generating a digest on a 32-bit CPU. Additionally, it can be noticed that the encryption phase of AES in Cipher Block Chaining mode is considerably slower than operation of producing a Hash-based Message Authentication Code digest (Figure 2).

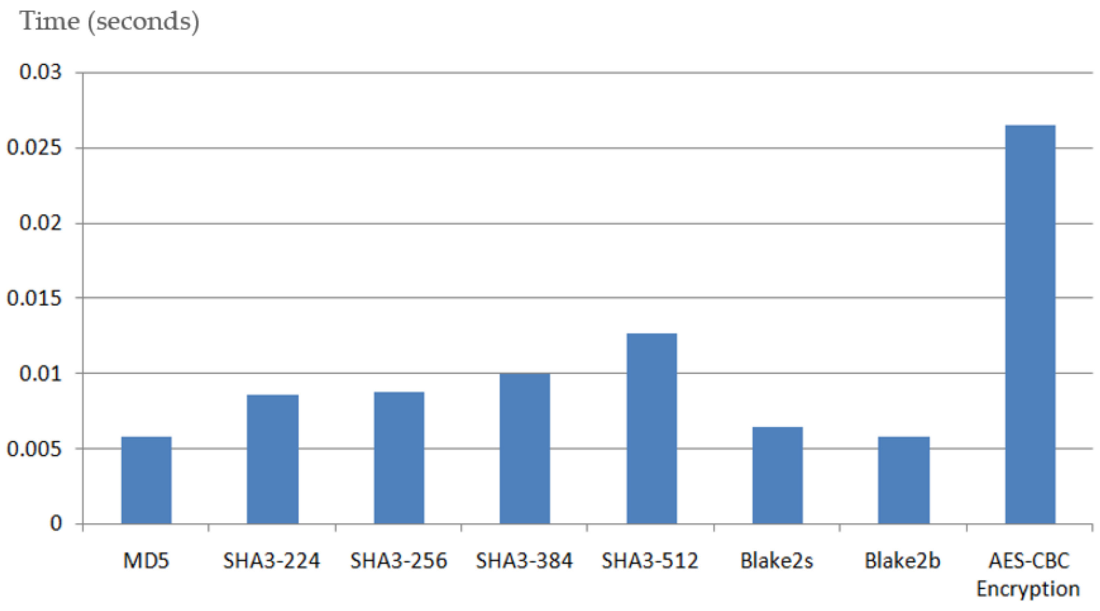

Figure 2. Average time measurements of HMAC generation using different hash functions.

As shown by the bar chart, an ideal algorithm for the Onion Omega2+ should be chosen from SHA3-224, SHA3-256, or Blake2. According to References [10,11], the Blake2 hashing algorithm is considerably faster at generating a digest, even when compared to SHA, SHA2, or SHA3, and it is more suitable for resource-constrained devices. Additionally, Blake2s was optimized to work on a 32-bit CPU and produces and output hash of 32 bytes (256 bits). The increase in performance on the Onion Omega, when comparing Blake2s to SHA3-224 or SHA3-256 digest generation is on average 20%. The reason why the comparison is made between Blake2s and SHA3-224 and SHA3-256 is because these algorithms, from the SHA3 category, are faster on a 32-bit CPU.

5. Algorithm Comparison

Based on the dataset and information presented above, using the Value-to-HMAC mapping method could greatly improve performance when compared to a symmetric encryption algorithm. On the Onion Omega2+, which has a 32-bit CPU, a cryptographic hash function that performs well can be chosen from SHA3-224, SHA3-256, and Blake2s, as suggested by the dataset (Table 1).

Table 1. Timing sample for HMAC generation and symmetric key encryption with different message sizes.

Message Length (Bytes)	8	16	32	64	128	256
MD5	0.003603	0.003682	0.003599	0.003794	0.003874	0.00404
SHA3-224	0.004464	0.00438	0.004791	0.004308	0.005777	0.00486
SHA3-256	0.00435	0.004455	0.004393	0.004464	0.004383	0.004802
SHA3-384	0.004407	0.004636	0.004402	0.004468	0.004769	0.005222
SHA3-512	0.004294	0.00426	0.004255	0.004265	0.004669	0.005444
Blake2s	0.003249	0.003188	0.003203	0.003182	0.003338	0.003622
Blake2b	0.003565	0.003619	0.003642	0.003599	0.00414	0.003944
AES-CBC Encryption	0.005253	0.005292	0.005894	0.006181	0.006816	0.009479
AES-CBC Decryption	0.003437	0.003563	0.003788	0.004225	0.005263	0.006448
AES-CBC Encryption + Decryption	0.00869	0.008855	0.009682	0.010406	0.012079	0.015927

According to [11], currently no vulnerabilities or attacks have been discovered for Blake2. Another aspect that underlines the advantage of providing confidentiality using a hash digest is that the digest itself cannot be decrypted the same way a symmetric-key algorithm can; an important property of a cryptographic hash function is that it is nearly impossible or unfeasible to reverse and retrieve the original data having only knowledge of the hash digest and hash function that created it. An attacker would have to compute the hash digest for every possible combination of characters that make up a message. HMAC creates a digest similar to a hash function, but it requires an additional input, which is represented by the secret key. Computing the hash digest for every possible message and key combination and comparing the hashes created against the captured HMAC digest would make this attack unfeasible.

As a comparison, AES in CBC mode only provides confidentiality and requires an additional mechanism to provide integrity. This can be accomplished by using an encrypt-then-hash method by generating the HMAC from the ciphertext or generating the HMAC signature from the information and then encrypting both together using the AES algorithm and following a hash-then-encrypt methodology. Implementing AES-CBC and an integrity checking mechanism as a solution for securing the data would be even slower than the method presented above.

The HMAC mapping method would be ideal for deployment in an environment with predictable messages such as sensor data or controller commands. The execution time of the value retrieval function is unnoticeable, even with a large list of values. However, having multiple tables to translate HMACs from multiple devices could prove to be an issue; therefore, the more values needing to be mapped, the more time it will take to generate the table and the more storage space the table will occupy.

Although there are other security solutions that are more comprehensive, such as SSL/TLS, this paper considers that IoT devices have a very limited amount of resources and cannot use these protocols efficiently. The Value-to-HMAC mapping method could be used as a base for future improvements and additions.

Due to its design, the method is able to achieve confidentiality and integrity by using only one mechanism, making it significantly faster than using an encryption algorithm in conjunction with an integrity checking mechanism. The chart in Figure 2 uses the dataset created by the script's timing function and displays a comparison between different HMAC functions (based on different hashing algorithms) and the encryption phase of AES in CBC mode.

This method involves information passed onto the broker as a hash. The Keyed-Hash Message Authentication Code algorithm takes a variable (e.g., a temperature value) as an input and a secret key in order to produce the digest. On the receiving device, a mapping table is created from the range of values, a chosen hashing algorithm, and using the shared secret key (Figure 3).

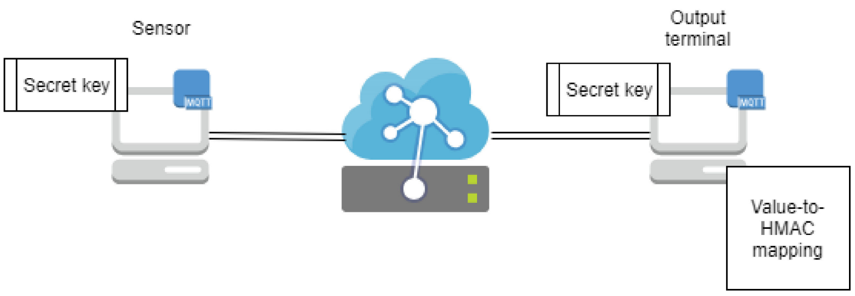

Figure 3. Message Queue Telemetry Transport nodes.

In this regard, only the nodes that possess the secret key can recover the original value, through a simple search within the previously generated mapping table. The HMAC mapping method is ideal for environments where the data produced is limited or within certain intervals of values (e.g., temperature sensor). A complete data flow of the experimental procedure is presented in Figure 4.

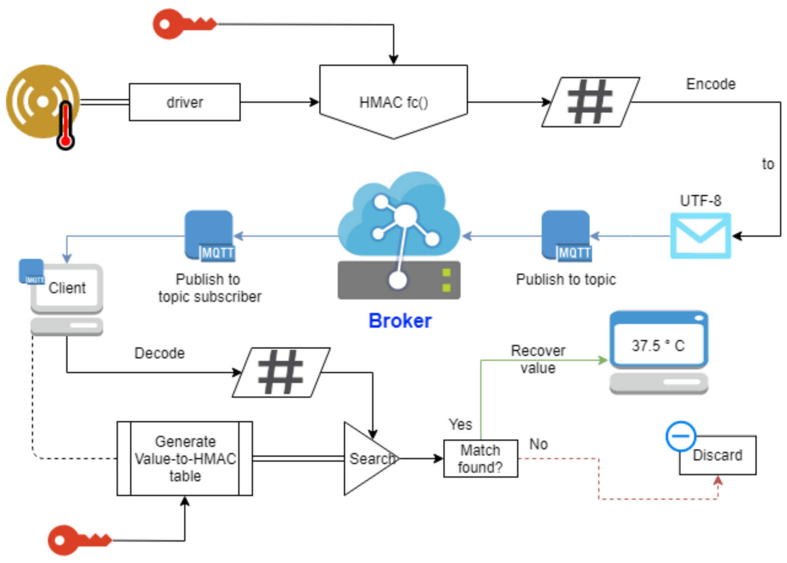

Figure 4. Data flow.

The hash tables are generated on each node that requires certain information and do not need to be shared. However, initial table generation will add some overhead, as demonstrated by the dataset. The initial cost of creating the table is also affected by the amount of value mappings it needs to insert into the table. The graph below shows the timings of table generation for 1000, 10,000 and 100,000 values, respectively.

Figure 5 compares table generation timings for each case against the search function, further consolidating the fact that the latter has no significant impact on performance.

Additionally, because the keys are not stored on the broker, the amount of keys needing to be shared will not have a significant or prolonged impact on the network's performance. The bandwidth required for the key distribution process will only be occupied during the initial exchange phase. Another factor that will affect the performance in a large network is how often the keys need to be replaced or how long the keys are valid for.

This design has a proven experimental advantage of being significantly faster than using an encryption algorithm in conjunction with an integrity checking mechanism. Therefore, it has a lower performance impact on the IoT device and on the network when compared to using a SSL/TLS suite. The script that produced the dataset was run on an Onion Omega 2+ device in order to simulate a resource-constrained device.

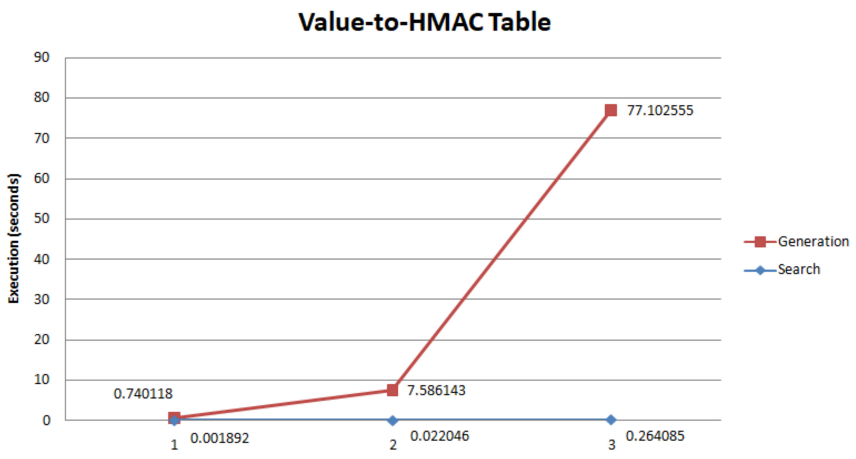

Figure 5. Timings of HMAC table generation and search function.

Although the Hash-based Message Authentication Code mapping method provides better performance and is able to ensure confidentiality and integrity of information, the communication using the Message Queue Telemetry Transport protocol is vulnerable to DoS attacks which need to be mitigated. A solution that could deal with this type of attacks is presented in [12].

A different implementation of the HMAC algorithm is covered in Reference [13], where the authors assess the performance of HMAC functions, based on different hashing algorithms, in order to achieve integrity of information. However, the method described here, Value-to-HMAC mapping, uses a different approach by creating an HMAC digest and a mapping table to achieve both integrity and confidentiality.

According to [14] the authors present a key management method based on elliptic key cryptography. The method focuses on providing security assurances while also being able to ensure decrease in transmission overhead, lower storage requirements as well as lower energy consumption. As demonstrated in the article, the probability of compromise will always be zero, regardless of how many nodes are compromised because each sensor makes use of a public/private key pair.

Moreover, in [15] the authors assess different key management schemes, key sharing mechanisms must be chosen based on the requirements of each implementation. The authors analyze techniques ranging from the simplest one, called 'Single network-wide key' where a single key is placed onto each node to be used for encryption and decryption, to more complex methods such as public key, key predistribution schemes, dynamic key management and hierarchical key management.

6. Conclusions

This paper covers a novel approach to achieving confidentiality and integrity of information and demonstrates through an experimental procedure that Value-to-HMAC mapping will perform better than a symmetric-key encryption algorithm as a solution to providing confidentiality.

Additionally, this method has the secondary advantage of achieving integrity, as only message hashes that match an entry in the pre-generated table on the receiving node will be accepted. The method will be ideal for an industrial environment, where nodes need to share predictable data such as sensors or controllers.

Moreover, it is important to consider not only the data that will be shared but also the platform on which the script will be run. If a node requires information from multiple sources, this will mean generating and storing multiple mapping tables, and in turn, more secret keys will be needed. Key and publish requests need to be managed properly in order to protect against Denial of Service attacks. However, as of version 1.0, the script can only achieve confidentiality and integrity of information. Additional mechanisms need to be implemented to deal with client authentication, key distribution and management system, as well as mechanisms to mitigate attacks such as Denial of Service and replay attacks.

Among increasing researches on IoT security [16–19], this paper presents a solution to meet specific application constraints.

Author Contributions: Conceptualization, D.D.; Methodology, D.D. software, open-source; Validation, X.C.; Formal analysis, D.D. and X.C.; Investigation, D.D.; Resources, D.D. and X.C.; Writing—original draft preparation, D.D.; Writing—review and editing, D.D. and X.C.; Visualization, D.D.; Supervision, X.C.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gupta, A. *IoT Hackers Handbook*; AttifyInc: Sunnyvale, CA, USA, 2017.
2. Feingold, J. Dyn issues analysis of cyberattacks. New Hampshire Business Re-View. 2016. Available online: http://www.nhbr.com/November-11-2016/Dyn-issues-analysis-of-complex-and-sophisticated-cyberattacks/ (accessed on 19 July 2018).
3. Nastase, L. Security in the Internet of Things: A Survey on Application Layer Protocols. In Proceedings of the 2017 21st International Conference on Control Systems and Computer Science, Bucharest, Romania, 29–31 May 2017.
4. Katsikeas, S.; Fysarakis, K.; Miaoudakis, A.; Bemten, A.V.; Askoxylakis, I.; Papaefsta-thiou, I.; Plemenos, A. Lightweight & Secure Industrial IoT Communications via the MQ Telemetry Transport Protocol. In Proceedings of the 2017 IEEE Symposium on Computers and Communications (ISCC), Heraklion, Greece, 3–6 July 2017.
5. Perrazzone, J.B.; Yu, P.L.; Sadler, B.M.; Blum, R.S. Cryptographic Side-Channel Signaling and Authentication via Fingerprint Embedding. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 2216–2225. [CrossRef]
6. Fischlin, M.; Janson, C.; Mazaheri, S. Backdoored Hash Functions: Immunizing HMAC and HKDF. In Proceedings of the 2018 IEEE 31st Computer Security Foundations Symposium, Oxford, UK, 9–12 July 2018.
7. Hao, Y. *The Boomeraang Attacks on BLAKE and BLAKE2*; Springer: Cham, The Netherlands, 2015.
8. Bogdanov, A.; Khovratovich, D.; Rechberger, C. Biclique Cryptanalysis of the Full AES. In *International Association for Cryptologic Research 2011*; ASIACRYPT 2011, LNCS 7073; Springer: Berlin/Heidelberg, Germany, 2011; pp. 344–371.
9. Siddavaatam, P.; Sedaghat, R.; Cheng, M.H. An Adaptive Security Framework with Extensible Computational Complexity for Cipher Systems. In Proceedings of the 11th International Conference for Internet Technology and Secured Transactions, Barcelona, Spain, 5–7 December 2016.
10. Pereira, G.C.C.F.; Alves, R.C.A.; da Silva, F.L.; Azevedo, R.M.; Albertini, B.C.; Margi, C.B. Performance Evaluation of Cryptographic Algorithms over IoT Platforms and Operating Systems. *Secur. Commun. Netw.* **2017**, *2017*, 2046735. [CrossRef]
11. Jain, A.K.; Jones, R.; Joshi, P. Survey of Cryptographic Hashing Algorithms for Message Signing. *Int. J. Comput. Sci. Technol.* **2017**, *8*, 18–22.
12. Kim, J.Y.; Holz, R.; Hu, W.; Jha, S. Automated Analysis of Secure Internet of Things Protocols. In Proceedings of the ACSAC 2017, Orlando, FL, USA, 4–8 December 2017.
13. Kiran, S.K.V.V.N.L.; Harini, N. Evaluating Efficiency of HMAC and Digital Signatures to Enhance Security in IoT. *Int. J. Pure Pllied Math.* **2018**, *119*, 13991–13997.
14. Du, X.; Guizani, M.; Xiao, Y.; Chen, H.H. A Routing-Driven Elliptic Curve Cryptography based Key Management Scheme for Heterogeneous Sensor Networks. *IEEE Trans. Wirel. Commun.* **2009**, *8*, 1223–1229. [CrossRef]
15. Xiao, Y.; Rayi, V.K.; Sun, B.; Du, X.; Hu, F.; Galloway, M. A Survey of Key Management Schemes in Wireless Sensor Networks. *J. Comput. Commun.* **2007**, *30*, 2314–2341. [CrossRef]
16. Du, X.; Xiao, Y.; Guizani, M.; Chen, H.H. An Effective Key Management Scheme for Heterogeneous Sensor Networks. *Ad Hoc Networks* **2007**, *5*, 24–34. [CrossRef]
17. Gao, C.; Siyi, L.V.; Wei, Y.; Wang, Z.; Liu, Z.; Cheng, X. An Effective Searchable Symmetric Encryption with Enhanced Security for Mobile Devices. *IEEE Access* **2018**, *6*, 2169–3536.

18. Wang, C.; Zhao, Z.; Gong, L.; Zhu, L.; Liu, Z.; Cheng, X. A Distributed Anomaly Detection System for In-Vehicle Network Using HTM. *IEEE Access* **2018**, *6*, 9091–9098. [CrossRef]
19. Wang, C.; Zhu, L.; Gong, L.; Zhao, Z.; Yang, L.; Liu, Z.; Cheng, X. Accurate Sybil Attack Detection Based on Fine-Grained Physical Channel Information. *Sensors* **2018**, *18*, 1424–8220. [CrossRef] [PubMed]