A Lightweight Authentication and Key Exchange Protocol for IoT

Abdulrahman BIN Rabiah*, K. K. Ramakrishnan*, Elizabeth Liri* and Koushik Kar†
*Department of Computer Science and Engineering, University of California, Riverside
Email: abinr001@ucr.edu, kk@cs.ucr.edu, eliri001@ucr.edu
†Department of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute
Email: koushik@ecse.rpi.edu

***Abstract*— Security in IoT environments is critical, as there are many situations where IoT devices provide sensory information that needs to be shared securely. However, providing authenticated and secure communication for IoT devices can be a challenge. IoT devices have many constraints, including limitations in computation, power, memory and energy. Moreover, they often have to go through a gateway/sink to connect to the network. For secure communication to the rest of the network, the IoT device needs to trust the gateway/sink, and this requires a means for the device to authenticate the gateway and vice-versa. We also seek to support secure communication even when the IoT device and gateway are disconnected from the rest of the network. In this paper, we provide a lightweight authentication and key exchange protocol for such IoT environments where the IoT device and gateway are communicating over a wireless channel. Our protocol depends on each pair of devices having two unique keys, a master key and an initial session key, provided at configuration time. The session key is constantly changing, and is used as the key for exchanging frames securely during a session. The protocol is lightweight and uses only symmetric-key cryptography and Hashed Message Authentication Code (HMAC)-based key derivation function (HKDF) to provide authentication, key exchange, confidentiality and message integrity. The protocol does not depend on any Trusted Third Party (TTP), and is a good fit for disconnected IoT environments. The keys are never exchanged over the network, providing perfect forward secrecy. The protocol is efficient in the amount of computation required, memory and energy usage.**

I. Introduction

The use of IoT devices is growing and sensors are becoming increasingly ubiquitous in our lives. There are a number of areas where critical functions, such as monitoring a patient's health status and the environment, depend on data collected from IoT devices. Some functions may frequently send large amounts of data, such as for video surveillance. Such data must be shared in a secure and authenticated manner.

IoT devices are resource constrained, being limited in processing, memory and the availability of power, especially because a lot of them are powered by batteries. They also mostly use wireless as the physical medium of communication. The IoT device uses the gateway/sink as the primary entity through which it communicates with the rest of the environment. For this reason, we seek to have secure and authenticated communication between the IoT device and the gateway/sink that it communicates with. However, traditional authentication and key establishment protocols are not entirely suitable in the IoT environment, because of its specific constraints. Using public key cryptography for authentication and key establishment is not feasible because of the dependence on a Trusted Third Party (TTP), and its high computational requirement. Public key cryptography places a severe demand on compute capability, memory and battery consumption that cannot be met in typical IoT devices. IoT environments may also need to operate in a disconnected mode, with no access to a TTP.

Authentication and key exchange between two nodes without a trusted third party requires some form of a priori establishment of a shared secret between the nodes. In addition, it is important to eliminate the single point of 'failure' (exposure of the secret) in the system. For this reason, we seek to have more than one secret key. Each secret key serves a different purpose. Moreover, we think it is important that the protocol provide Perfect Forward Secrecy (PFS). PFS means that even if the long-term secret key is known somehow at some point, all past session communications must still be secure. In addition, a different session key for each session allows a smaller number of messages encrypted with one session key, making it more difficult for the attacker to find session keys using cryptanalysis. Moreover, even if the attacker is able to find a session key somehow, that session key (being specific to that session) is not useful in decrypting data of future sessions. This motivates the use of limited-life session keys in the lightweight authentication and key exchange protocol for IoT environments that we propose in this paper.

A common practice in IoT security nowadays is to have IoT devices already provisioned with digital certificates, signed by the manufacturer, which are used for authentication and key exchange. However, this common approach requires IoT devices to have high computational capability and high power consumption, as digital certificates rely on public key cryptography. In contrast, we propose using symmetric key cryptography, which is much simpler. Moreover, the common approach requires more memory as public key cryptography uses longer keys compared to symmetric key cryptography and digital certificates, which are a few KB have to be stored. Our proposed protocol has a much smaller memory requirement. Relying on digital certificates without contacting a TTP creates the single point of 'failure' as the only system secret for

Workshop on Decentralized IoT Security and Standards (DISS) 2018
18 February 2018, San Diego, CA, USA
ISBN 1-891562-51-7
https://dx.doi.org/10.14722/diss.2018.23004
www.ndss-symposium.org

authentication is the private key. If the only system secret is compromised somehow, with this common approach, the authentication and key exchange can be done by the attacker; in contrast, the proposed protocol does not depend solely on one system secret for authentication and key exchange, so even if one of system secrets is found somehow, the authentication and key exchange process cannot be spoofed by the attacker.

The protocol we propose has two keys shared between the IoT device and the gateway: a shared long-term symmetric master key, denoted by K_m, and a shared short-term symmetric session key, denoted by K_s. The initial session key, is denoted by K_{iks}. Both the master key and the initial session key are inserted manually once into the IoT device and the gateway/sink. Each gateway authenticates the IoT device and vice versa using messages exchanged between the two devices, which are encrypted using the shared K_m and hashed using the shared K_{iks}. Moreover, the protocol ensures that both ends continue to have a shared K_s, that is used for encrypting and hashing session messages. More importantly, the secret keys are never exchanged explicitly over the network. A re-authentication and update of session key, K_s, takes place periodically. The messages exchanged between updates is termed a 'session'. Each session has a threshold, which is a maximum number of frames exchanged in that session. The update of the session key, K_s, is based on a random set of the frames exchanged in the previous session as well as the previous K_s, and the master key K_m. As a result, even if the attacker found the K_s somehow at some point, he would not be able to calculate next session keys and decrypt messages unless he has *both* the K_m *and* the current K_s. Additionally, since it is hard to have a perfect channel where all frames are captured by the attacker [12], our protocol forces the attacker to face three different obstacles. Overcoming the IoT system's security is thus much more difficult.

In addition, our protocol takes into consideration the various constraints and requirements of IoT environments. It uses lightweight mechansims: symmetric key cryptography and a Hashed Message Authentication Code (HMAC)-based Key Derivation Function (HKDF) [2], [6] for confidentiality, message integrity, authentication and key exchange. It seeks to minimize the number of messages (and total bytes exchanged) for authentication and key exchange [3], [7], [9]. It also does not rely on a central, trusted third party in any way. This provides a secure, fast as well as energy and space efficient authentication and key exchange protocol.

The rest of the paper is organized as follows: The next section describes related work. Section III describes the system model and assumptions, including the network attack models considered. Section IV discusses the proposed protocol in detail and Section V discusses the analysis of the security of the proposed protocol. We conclude in Section VI.

II. Related Work

There are a number of authentication and key establishment protocols that have been proposed for use in an IoT environment. Some use public key cryptography for authentication and key establishment [5], [10], which is expensive considering the resource-constraints of IoT devices. Others require authentication and key establishment using a central trusted third party [8], such as a Certificate Authority (CA). However, this is difficult to use in disconnected environments. Another approach is to depend on the hardware capability for introducing randomness for authentication and key establishment using Physically Unclonable Function (PUF) [1]. While the PUF approach is helpful, it is still not yet widely deployed in devices, which prompts us to look at alternatives.

One approach examines key establishment separately [12], utilizing the randomness in wireless channels to update the session key. This approach depends mainly on the reasonable assumption that the wireless channel is not perfect (loss free). The design uses a different session key for each session, a design philosphy that we leverage as well. A session is defined by a threshold number of "One Time Frames" (OTFs), where this number is agreed upon a priori. OTFs are frames transmitted *and* received exactly once (i.e., without retransmission of those frames). It assumes that both entities start with a publicly well-known fixed value as the initial session key. This key is used to encrypt the session's frames initially. During a session, OTFs are stored at both ends, and after a threshold of OTFs are communicated, the session keys are updated. To update the session key, the current session key and OTFs are used together to incrementally generate a new session key. The design depends on the attacker not having the same channel conditions as the receiver and therefore will see a different subset of OTFs. However, an attacker with a perfect channel (e.g., being very close to the transmitter) makes this scheme susceptible to both passive (e.g., eavesdropping) and active attacks (e.g., hijacking). Our protocol therefore removes this dependency on OTFs and the attacker having an imperfect wireless channel, and builds on the technique proposed in [12].

III. IoT Environment Constraints and Requirements

IoT environments are associated with a number of constraints. Foremost, IoT devices have limited energy as they are primarily dependent on batteries (e.g., disposable coin cell batteries that might last for 1-2 years at most with limited use). Second, IoT devices have limited processing capabilities and limited memory, with only a few KB of RAM and tens of KB of EEPROM. IoT devices rarely have fast re-writable non-volatile data storage.

In addition, the IoT device and gateway may have to operate in a disconnected environment, limiting access to a central entity or a trusted third party. Finally, the approach for managing these devices needs to be scalable as there are likely to be a very large number of IoT devices, and it is important to have as little human intervention as possible.

A. Network Model and Assumptions

The generic network topology that we consider is shown in Figure 1. There are multiple IoT devices and a single Gateway, and each IoT device communicates with other nodes only through the Gateway. We assume, in this paper, that MAC layer protocol on the wireless link delivers packets reliably, in-sequence.

A new IoT device added to this IoT network will first have to authenticate with the gateway and vice-versa to establish communication with the rest of the environment. We assume

Fig. 1: Network Topology

that both the IoT device and the gateway have a limited amount of non-volatile storage in the form of an EEPROM that can store shared keys. We further assume that a limited amount of manual configuration is feasible, so that the administrator can set up shared keys for the gateway and IoT device on a pairwise basis. We assume that an attacker does not have physical access to the devices and thus cannot access the configured keys that are stored on the devices.

B. Attack Scenarios

In this section, we consider the range of possible ways that an attacker with access to the link may seek to exploit vulnerabilities in the protocol we design. We then describe (briefly) how these attacks are prevented by our protocol described in this paper.

1) An attacker may try to sniff or modify session messages.
2) An attacker may seek to discover the secret keys by snooping on the authentication and key exchange traffic.
3) An attacker may try to modify or spoof authentication and key exchange messages to cause disruptions.
4) An attacker may try to launch replay attacks to cause disruptions.
5) An attacker may launch man-in-the-middle attacks and have unauthorized access to confidential messages and possibly alter them.

We believe that our protocol is designed to thwart all of these attacks, and we have performed a security analysis of the protocol (described briefly in Section V) to verify this.

IV. Lightweight Authentication and Key Exchange Protocol

As mentioned earlier, for each Gateway-IoT device pair, we maintain two shared secret keys: a shared long-term symmetric master key, called K_m; and a second, shared short-term symmetric session key, called K_s. Initially, the K_m and K_s are manually inserted into both devices, in non-volatile storage such as EEPROM. The specific initial value of K_s, that is added manually is denoted by K_{iks}. Each IoT device and Gateway pair has a unique pair of K_m and K_{iks}, specific for their use and both K_m as well as K_{iks} play an important role in authentication in our protocol. IoT devices and the Gateway are authenticated using messages encrypted with the shared K_m and hashed with the shared K_{iks} while both nodes use the shared K_s to encrypt and hash messages during a session. The protocol makes use of authenticated encryption, namely the Counter with CBC-MAC (CCM) block cipher mode [4], [11], to ensure confidentiality and message authentication in all phases. With CCM mode, CBC-MAC is used to calculate a Message Authentication Code (MAC) for the whole frame (header, nonce and payload) using a symmetric secret key (either K_m or K_s, depending on the phase) and the Counter mode is used to encrypt the payload as well as the MAC using a nonce and symmetric secret key (either K_{iks} or K_s, depending on the phase) whereas the header needs to be in plaintext so that the other end can learn some information about the frame in order to process it, such as the MAC addresses. A new session begins when the number of session frames exchanged during the previous session reaches a preset threshold (agreed upon a priori by both sides).

K_s is updated regularly for each session, and is never exchanged explicitly over the network. In addition when updating K_s, the previous value of K_s is used, so that the value of K_s is cumulative, meaning the new session key K_s depends on the previous session's K_s.

The protocol we propose takes advantage of the randomness in regular frames and uses it to generate more random session keys. Both entities integrate randomness by incorporating a randomly selected set of future session frames into a newly generated K_s. This set of future session frames is agreed on and communicated securely using another secret key, K_m, every time a K_s update takes place. Since the frame threshold is known by both entities, the range of frame sequence numbers till the next K_s update can be calculated. Using a random number generator, the Gateway selects a random set of future frame sequence numbers within this range so that both entities keep only those frames in a buffer. For each session, the larger the number of frame sequence numbers selected, the more random the new K_s is. This randomness provides better security for the exchange of K_s. However, we note that IoT devices have limited memory, which can limit the number of session frames they can buffer.

Thus, with every new session, it becomes harder for the attacker to keep up with the K_s updates. To violate confidentiality, the attacker would need to have both the K_m and K_s. Although it is hard to have a perfect channel where all frames are captured [12], we make no assumption about the ability of the attacker to have or not have a perfect channel to capture all the frames sent by the IoT device or the Gateway. When there is a failure in synchronization of the K_s at either end for any reason, whether malicious or non-malicious, either device can always initiate a reset of the K_s to a new random key derived from K_{iks}, along with nonces, so they can always have a stable communication channel.

A. Initial Setup Phase for a New IoT Device

When an IoT device first joins the network, the initial setup needed for both the IoT device and the gateway is to authenticate each other and negotiate a K_s. This is done using the shared K_m and the K_{iks} that have been installed manually

Acronym	Definition
K_m	Shared long-term symmetric master key
K_s	Shared short-term symmetric session key
K_{iks}	Initial K_s
ID_I	IoT device ID
ID_G	Gateway ID
$Nonce_1$	Randomly generated value used once
$Nonce_2$	Randomly generated value used once
$RandFrmSeqs_i$	Random set of sequence numbers of session $(i-1)$ frames to be used in session i's session key update
$RandFrmSeqs_{i+1}$	Random set of sequence numbers of session i frames to be used in session $(i+1)$'s session key update
$RandFrm_i$	Random frames from last session (session $(i-1)$) based on $RandFrmSeqs_i$
$KeyLength$	Desired length of the derived key

TABLE I: Glossary

into both the gateway and new IoT device. Figure 2 shows the protocol interaction.

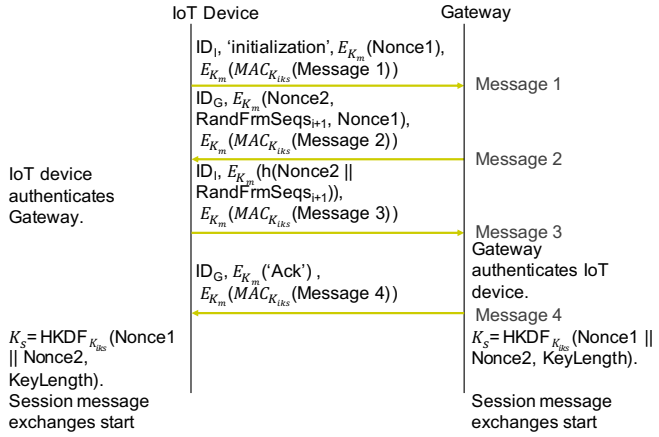

Fig. 2: Initial Setup Phase

a) Message 1: The IoT device sends its ID_I, an 'initialization', a Nonce1 and $\mathbf{MAC}_{K_{iks}}$(Message 1), encrypted with the K_m to the gateway. In order to prove its identity, the authenticator, which is the IoT device at this point, provides a new challenge, Nonce1, that will be used by the supplicant, Gateway. Additionally, using K_{iks} to calculate the $\mathbf{MAC}$ for the whole message, including the nonce verifies both the data integrity and the authenticity of the message, thus helping to prevent disruptions that may be caused by an attacker who has determined the K_m somehow.

b) Message 2: Upon receipt of the first message from the IoT device, the Gateway first selects a random set of sequence numbers of future frames, $RandFrmSeqs_{i+1}$, to be considered when updating the K_s next time. This is communicated, along with a Nonce2 (a new challenge picked by the Gateway) and Nonce1, to the IoT device, as part of Message 2. The Nonce2, $RandFrmSeqs_{i+1}$ and Nonce1 are encrypted with the master key, so that it is confidentially exchanged with the IoT device. Thus, when this session starts, the IoT device will retain a copy of frames with those sequence numbers to be used in the next K_s update.

In order for the IoT device to authenticate the Gateway and also confirm that it received the right set of $RandFrmSeqs_{i+1}$, the Gateway also calculates the $\mathbf{MAC}$ of the whole message, including Nonce1 and $RandFrmSeqs_{i+1}$ using the K_{iks} as a key and sends it to the IoT device. The IoT device can thus verify that the Gateway has the correct K_{iks} and has received Nonce1 correctly, when it calculates the corresponding $\mathbf{MAC}$ using K_{iks}. All the parameters sent by the Gateway, ID_G, the encrypted Nonce2, along with the $RandFrmSeqs_{i+1}$, and the Nonce1 are encrypted using K_m and hashed using K_{iks}.

c) Message 3: When the IoT device receives Message 2, it verifies that the hash value of the $\mathbf{MAC}$, that it just received matches what it calculates based on the decrypted message using K_m. The IoT device marks the Gateway as authenticated and confirms that the IoT device has received the right set of $RandFrmSeqs_{i+1}$. Moreover, the IoT device verifies that the Gateway has the correct K_{iks} and has received Nonce1 correctly from Message 1, thus authenticating the Gateway.

The IoT device calculates the hash of Nonce2 and $RandFrmSeqs_{i+1}$. $\mathbf{h}$(Nonce2$||RandFrmSeqs_{i+1}$) and $\mathbf{MAC}_{K_{iks}}$(Message 3) are communicated to the Gateway for it to authenticate the IoT device and for it to confirm that the $RandFrmSeqs_{i+1}$ was received correctly by the IoT device. The IoT device sends ID_I, and the encryption of $\mathbf{h}$(Nonce2$||RandFrmSeqs_{i+1}$) and $\mathbf{MAC}_{K_{iks}}$(Message 3) using K_m.

d) Message 4: When the Gateway receives Message 3, it verifies the $\mathbf{MAC}$. If valid, the Gateway marks the IoT device as authenticated, and it knows the IoT device has received the right set of $RandFrmSeqs_{i+1}$. This also lets the Gateway know that the IoT device has the correct K_{iks} and has received Nonce2 correctly from Message 2. The Gateway sends ID_G, the encryption of an ack and $\mathbf{MAC}_{K_{iks}}$(Message 4) using K_m to the IoT device for it to confirm that Message 3 was received correctly by the Gateway. The Gateway now sets K_s to a random key derived from Nonce1 and Nonce2, used as a salt input (a random value), using K_{iks} as a HKDF key.

When the IoT device receives Message 4, it verifies the $\mathbf{MAC}$. If valid, the IoT device knows Message 3 was received correctly. The IoT device sets K_s to the random key derived from Nonce1 and Nonce2 using K_{iks} as a HKDF key.

B. Normal Communication between Gateway and IoT Device

During normal communication between the IoT device and Gateway, both devices have agreed upon a K_s, and they exchange messages securely using K_s. During this phase, both devices keep the frames corresponding to the current $RandFrmSeqs_{i+1}$ to be used for the next K_s update until the threshold of frames exchanged is reached. Not only is the $RandFrmSeqs_{i+1}$ not known to the adversary, it is also possible that not all of the frames of the session received by the adversary. IoT link layer protocols such as IEEE 802.15.4 are also capable of protecting the confidentiality and integrity of data. Therefore, our protocol provides the link layer with the K_s for this purpose. An approximate session interaction is shown in Figure 3. Normal session frames are encrypted and hashed with the K_s and sent to the other end. An ack, encrypted and hashed with the K_s, is sent by the receiver to acknowledge correct receipt of a frame. When the threshold of frames exchanged is reached, we go to the next phase for update of K_s.

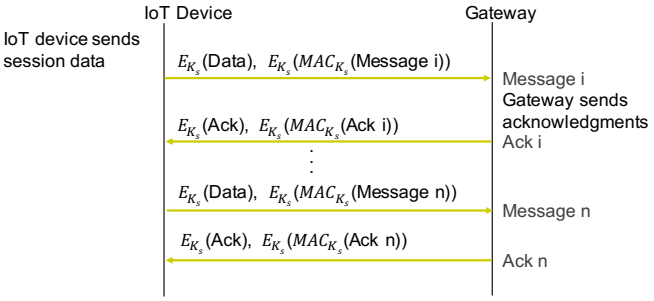

Fig. 3: Normal Communication between Gateway and IoT Device

C. Session Key Update

The update of K_s is shown in Figure 4.

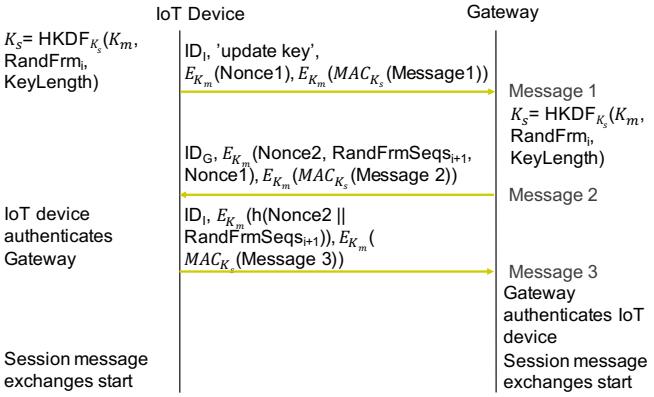

Fig. 4: Session Key Update

a) Message 1: The IoT device calculates K_s for the next session, which will be the HKDF of random frames based on $RandFrmSeqs_i$, used as an information input (context and application specific information), that it received in the last session, and K_m, used as a salt input, using the current K_s. Then, the IoT device sends ID_I, the command to 'update key' and Nonce1, along with $\mathbf{MAC}_{K_s}$(Message 1) encrypted with the K_m to the Gateway.

b) Message 2: When the Gateway receives the update key message, it verifies the threshold has been reached as well as the $\mathbf{MAC}$. It then calculates K_s for the new session, as the HKDF of the random frames based on $RandFrmSeqs_i$, used as an information input, and K_m, used as a salt input, using the current K_s. It calculates the $\mathbf{MAC}$ using K_s as a key which is communicated to the IoT device in this step. The Gateway sends ID_G and the encryption of Nonce2, $RandFrmSeqs_{i+1}$, Nonce1 and $\mathbf{MAC}_{K_s}$(Message 2) using the K_m to the IoT device.

c) Message 3: When the IoT device receives Message 2, it marks the Gateway as authenticated by verifying the $\mathbf{MAC}$ as in Sec IV-A. It calculates the hash of Nonce2 and $RandFrmSeqs_{i+1}$ to match the action at the Gateway. The IoT device sends ID_I and the encryption of $\mathbf{h}$(Nonce2$||RandFrmSeqs_{i+1}$) and $\mathbf{MAC}_{K_s}$(Message 3) using K_m. When the Gateway receives Message 3, it marks the IoT device as authenticated once the $\mathbf{MAC}$ is verified, and this completes the update phase.

D. Mutual Re-Authentication and Re-Establishment of Session Keys

When there is a failure in synchronization of K_s at either end for any reason, whether malicious or otherwise, both devices can renegotiate K_s securely. Recall that both K_m and the K_{iks} are maintained in both devices in non-volatile memory throughout. Thus, both devices use K_m and K_{iks} to re-authenticate and re-establish a newly generated random K_s derived from Nonce1, Nonce2 and K_{iks}, which makes K_s to be different with high probability, each time a reset of K_s is performed. For example, the IoT device or Gateway may reboot for any reason, resulting in both ends having to reset their K_s to a new random K_s and re-authenticate and re-establish the session key securely, as shown in Figure 5.

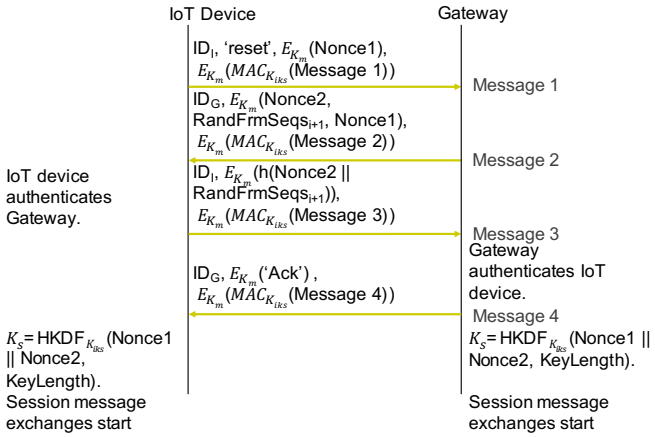

Fig. 5: Mutual Re-Authentication and Re-Establishment of Session Keys

This process is very similar to the initial authentication and key establishment discussed in Section IV-A, except for

the command to 'reset' instead of 'initialization'. The main distinction is that all outstanding frames are flushed from the memory at both devices so that both devices have same frames for the next update of K_s.

V. Security Analysis

We have done a careful security analysis of the proposed protocol for a range of possible attacks that was described in Section III-B. We believe that having both the keys, K_m and K_s provide better security as it makes it more difficult for the attacker to derive both keys. Our analysis showed that knowing just one of the two keys is not sufficient to violate the confidentiality and message integrity of our protocol. We also were able to show that the proposed protocol satisfies the important perfect forward secrecy (PFS) property. We highlight a few key points here:

1) The length of K_m and K_s makes it difficult to use brute force tactics to find them.
2) Knowing K_m alone is not sufficient to decrypt session messages since they are encrypted using another key.
3) The session keys are never explicitly exchanged in the messages; knowing K_m is not enough to know K_s.
4) Even if an attacker knows K_m, has a perfect channel, and has recorded all messages exchanged from the very beginning, they still cannot obtain a K_s because the keys are never exchanged explicitly over the network. In addition, the attacker cannot calculate current K_s because it depends on the previous K_s.
5) If an attacker knows K_s and even if he has a perfect channel, he cannot know the agreed upon random frames that will be used to construct the next session's key.
6) If only K_m is known and the IoT device is rebooted or the channel is jammed by the attacker who also records all the messages exchanged, the protocol remains robust against man-in-the-middle attacks and session hijacking.
7) If the attacker only knows K_m at the initial setup phase, man-in-the-middle or session hijack attacks still cannot succeed because the attacker has to have the other key, K_{iks}, to prove it is the legitimate entity to the other end.

We do not include the details of our security analysis due to space limitations.

VI. Conclusions

We propose a lightweight authentication and key exchange protocol to help secure communication in IoT environments that typically communicate over wireless channels. The proposed protocol aims to provide secure, lightweight, energy and space efficient authentication and key exchange for resource-constrained devices without depending on a central trusted third party. The protocol's main strengths include having secret keys that are not explicitly exchanged over the air. The protocol frequently updates the symmetric session key used for encrypting and hashing the data exchanged based on a random (but previously agreed upon) number of frames of a session. We also introduce two obstacles that have to be compromised to break the protocol, namely a symmetric long-term key (K_m), a symmetric short-term key (K_s). In the proposed protocol, therefore, there is no single component that can be exploited by an attacker to compromise the authentication and key exchange protocol or the confidentiality and integrity of the information exchanged. For future work, we will enhance our protocol to require only one pre-shared key and still achieve PFS, but it would need to rely on a commit protocol. We are currently working on accommodating lossy links.

Acknowledgment

This work was supported in part by NSF grant CNS-1619441 and a gift from Futurewei Technologies.

References

[1] M. N. Aman, K. C. Chua, and B. Sikdar, "Secure data provenance for the internet of things," in *Proceedings of the 3rd International Workshop on IoT Privacy, Trust, and Security*, ser. IoTPTS '17. ACM, 2017, pp. 11–14.

[2] M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," in *Crypto*, vol. 96. Springer, 1996, pp. 1–15.

[3] L. Casado and P. Tsigas, "Contikisec: A secure network layer for wireless sensor networks under the contiki operating system," *Identity and Privacy in the Internet Age*, pp. 133–147, 2009.

[4] P.-A. Fouque, G. Martinet, F. Valette, and S. Zimmer, "On the security of the ccm encryption mode and of a slight variant," in *Applied Cryptography and Network Security*. Springer, 2008, pp. 411–428.

[5] H. R. Hussen, G. A. Tizazu, M. Ting, T. Lee, Y. Choi, and K.-H. Kim, "Sakes: Secure authentication and key establishment scheme for m2m communication in the ip-based wireless sensor network (6l0wpan)," in *Proceedings of Fifth International Conference on Ubiquitous and Future Networks (ICUFN)*, July 2013, pp. 246–251.

[6] H. Krawczyk, "Cryptographic extraction and key derivation: The hkdf scheme." in *CRYPTO*, vol. 6223. Springer, 2010, pp. 631–648.

[7] P. Mahajan and A. Sachdeva, "A study of encryption algorithms aes, des and rsa for security," *Global Journal of Computer Science and Technology*, 2013.

[8] P. Porambage, C. Schmitt, P. Kumar, A. Gurtov, and M. Ylianttila, "Pauthkey: A pervasive authentication protocol and key establishment scheme for wireless sensor networks in distributed iot applications," *International Journal of Distributed Sensor Networks*, vol. 10, no. 7, p. 357430, 2014.

[9] N. R. Potlapally, S. Ravi, A. Raghunathan, and N. K. Jha, "Analyzing the energy consumption of security protocols," in *Proceedings of International Symposium on Low Power Electronics and Design*, ser. ISLPED '03. ACM, 2003, pp. 30–35.

[10] Y. Qiu and M. Ma, "A mutual authentication and key establishment scheme for m2m communication in 6lowpan networks," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 6, pp. 2074–2085, Dec 2016.

[11] D. Whiting, R. Housley, and N. Ferguson, "Counter with cbc-mac (ccm)," Internet Requests for Comments, RFC Editor, RFC 3610, September 2003. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3610.txt

[12] S. Xiao, W. Gong, and D. Towsley, "Secure wireless communication with dynamic secrets," in *Proceedings of IEEE INFOCOM 2010*, March 2010, pp. 1–9.