IETE Technical Review

ISSN: 0256-4602 (Print) 0974-5971 (Online) Journal homepage: http://www.tandfonline.com/loi/titr20

Securing Multicast Group Communication in IoT-Enabled Systems

Subho Shankar Basu & Somanath Tripathy

To cite this article: Subho Shankar Basu & Somanath Tripathy (2018): Securing Multicast Group Communication in IoT-Enabled Systems, IETE Technical Review, DOI: 10.1080/02564602.2017.1407681

To link to this article: https://doi.org/10.1080/02564602.2017.1407681

Published online: 20 Mar 2018.

IETE TECHNICAL REVIEW, 2018
https://doi.org/10.1080/02564602.2017.1407681

Securing Multicast Group Communication in IoT-Enabled Systems

Subho Shankar Basu and Somanath Tripathy

Department of Computer Science, Indian Institute of Technology, Patna, India

ABSTRACT
Current trend is being extended from the traditional Internet to the small, cheap, and low-power Internet of Things (IoT) in which the objects are being equipped with a device having computation and communication capabilities. As a result, all these objects can be connected to the Internet and have the capability to communicate among each other. This connection infrastructure among the objects would face different types of malicious attacks. Hence securing these objects is a primary goal. There are a lot of security mechanisms available today, but most of them are quite heavy in terms of computation and communication. As the IoT objects have very limited resources and mostly run on battery power, it is difficult to embed intensive computations on these resource-constrained devices. Datagram Transport Layer Security (DTLS) protocol has been standardized to work in cohesion with the CoAP protocol to provide security. But DTLS does not fit well for multicasting, though it is a quite common need for IoT environments. Indeed there are some adaptations for DTLS protocol to function in a multicast environment, but it consumes much communication and computation resources. We propose a mechanism called S-CPABE (Segregated Ciphertext Policy Attribute-Based Encryption) based on CPABE, particularly targeting the multicast needs and tailoring to the IoT framework. The novelty of S-CPABE lies on providing equivalent security as CPABE with reduced resource requirements at the low-power end devices. This mechanism perfectly meets the needs for secure multicast in an IoT environment and consumes much less resources as compared to DTLS.

KEYWORDS
Attribute-based encryption; Internet of Things; IoT security protocols; Multicast security; Resource-constrained devices

1. INTRODUCTION

"Internet of Things" (IoT) is going to change the world in a significant way in the near future. It is going to be inherently applied to our day-to-day living and make our lives much easier. As today no one can think of a single day without Internet, a day would come when we cannot think of a single moment without the aid of IoT. Today's Internet is mostly limited to devices like personal computers, laptops, smartphones, tablets, etc. But the idea of IoT is to have the things around us communicate among each other and the Internet. So the smallest day-to-day objects we regularly use, say, for example, a piece of pen, will be connected to the Internet and order a refill on its own when it is out of ink. Things will have the sufficient intelligence to interact with other things, the environment around them, and of course human beings [1–3]. A lot of research is, therefore, taking place in this field from both industry and academia. And with so many different technologies available today from Zigbee, WiFi, Bluetooth, NFC, LPWANs, and 5G cellular technologies, IoT frameworks are turned into a reality with possible applications in smart homes, smart environment, smart agriculture, etc. [4,5].

Meanwhile, there are a number of challenges of which security is the major concern. As everything gets connected to the Internet, devices get more prone to risk of threats and malicious attacks. People or other objects may post harmful contents to these networks, steal data from these devices, and make improper and illegal use of these devices [6]. So design of the security aspects in parallel with the networking solutions is essential for the safer and popular deployment [7,8]. The next challenge is the constraint in terms of available resources on these devices. As these devices have very less memory, processing capabilities, communication bandwidth, and mostly powered by batteries, light-weight protocols are a must to conserve energy. Third, as they are small and cheap, there are billions of those devices communicating with one another.

Further, in many applications, there is the necessity of multicast (one-to-many) communications. As an example application, let us consider that car manufacturing company wants to have a ubiquitous connectivity to all its manufactured cars and post updates to these cars as and when required on demand. Say the company has already launched a batch of cars and wants to upgrade the firmware for the pollution controller installed in these cars. That batch size is unknown and may be in thousands and does not define a group or of any particular model. But the company knows that they all have the same engine model and manufactured in a particular time frame. These two attributes define the set of cars destined for the update and is sufficient to work in an Attribute-Based Encryption (ABE) scheme. Also, cars from other manufacturers should be unaware of this update and the data in this case has to be secretly and confidentially shipped to only those destined cars. The company sends a multicast request with the updated firmware to be installed on these sets of cars. Thus at any point of time, the company has full definition of how to address the cars and with what data to be fetched to them seamlessly. Also, it is to be noted that the company need not be knowing each and every individual identifier of the cars and hence forms a scalable environment. It generalizes the destined cars by some set of characteristics or attributes. To address this issue, we exploit the characteristics of ABE mechanism in the proposed design.

Thus, multicast is inherently required in IoT scenarios to facilitate a sender to communicate a message to a group of receivers. Typical architectures for multicasting have a central group controller which manages the group. Interested members join into the group and share a group key among themselves for secure communication. The group key is also renewed whenever there is a change in the members joining or leaving the group to maintain forward and backward secrecies. The sender sends a message encrypted with a group key and the receivers decrypt the message using the same group key. But the greatest disadvantage of this mechanism is that it is a centralized system and the group controller becomes the central point of attack. Also, it does not scale well for billions of devices which is the need for IoT. This motivated us to think of a decentralized system which would suffice the needs of IoT and also be scalable. This paper proposes a secure multicast protocol called S-CPABE (Segregated Ciphertext Policy Attribute-Based Encryption) based on CPABE mechanism that quite fits well to the IoT environment. Also, there is a beautiful flexibility in ABE mechanisms where the group can be dynamically formed without any static identity of the group by defining the group with a set of attributes. By constraining and relaxing the definition of the attribute set of the members, any intended group can be dynamically defined. Relaxing the group definition results in a broadcast while constraining the group definition results in a very precise view of which all members belong to a group. To secure these multicast communications among these low-power devices, an efficient protocol is needed. We have particularly addressed this need of designing a suitable secure multicast protocol for the IoT.

In a multicast environment, the actual data transfer is encrypted by a symmetric algorithm using this common group key, as symmetric algorithms are lighter in terms of computation and thus suited for IoT applications. The main challenge thus boils down to finding a way to distribute the common group key to the group members. Pre-shared keys are not a solution because they will not satisfy the requirements of dynamic group characteristics in most IoT applications. Existing solutions like DTLS is good for unicast applications but does not scale for multicast ones. So, for a very dynamic multicast environment, a flexible but moderately light-weight solution is required.

This motivated us to apply the concepts of ABE-based systems and design the S-CPABE protocol. ABE systems are highly flexible and need little knowledge of the participating members in a multicast group unlike DTLS, but come at the cost of an increased computation. The main idea towards the design of S-CPABE is to take the advantages of CPABE mechanism, on the one hand, and making it to work the same way effectively, but reducing the load on the resource-constrained end nodes, by off-loading the majority of the tasks to the powerful gateway, but still in a consistent way so as to fulfil the basic working principle of CPABE.

The rest of the paper is organized as follows. Works done in this direction for IoT is precisely discussed in Section 2. Preliminaries of ABE is briefly discussed in Section 3. The proposed scheme is described in Section 4 with its security model in Section 5. It is then analysed in Section 6 and advantage of S-CPABE over DTLS for multicast is discussed in Section 7. Finally, Section 8 concludes the work.

2. RELATED WORK

Different security approaches including [9,10] have been proposed, but they do not fit intuitively to the IoT working environment. DTLS [11] has been standardized to work in the security layer for CoAP. DTLS multicast security [12] makes use of the DTLS protocol in a multicast environment for IoT. It does not propose any new architecture for multicasting, but uses the same security feature of DTLS for unicasting, clubbed with the multicast feature from IP layer. Thus essentially it does not solely serve the multicasting phenomenon, but may be said rather as a fit of the DTLS unicast mechanism in a multicast environment. DTLS mechanism [11] with group communication features [13] forms a basis of the DTLS multicast mechanism. The DTLS multicast mechanism essentially forms multiple DTLS tunnels from the group controller to the end receivers for the distribution of the group secret key. Each consequent update of the group key due to a change in the group members is carried out by the group controller through reassigning a new group key to all the present set of group members participating in that group [13]. So for each member of the group, this DTLS multicast mechanism maintains a dedicated tunnel which remains active for the entire lifetime of the member in the group. In case the tunnel is destroyed or compromised for some reason, the same has to be negotiated and re-established between the group controller and that group member. As DTLS is a synchronous and connection-oriented protocol, and requires negotiation between the two parties, this DTLS multicast requires huge communication delay due to the exchange of large certificates and other keying materials. Considering the environment for IoT having huge number of communicating devices, this flat architecture would not be possible in reality. Moreover, there arises other disadvantages like interference issues, network jamming, and congestion due to so many DTLS tunnels. The discussion of limitations of this scheme is deferred to the analysis in Section 7.

Some interesting approaches based on the DTLS protocol have been studied in Axiom [14]. The authors nicely put forward a mechanism to avoid the costly DTLS handshake, thus saving resources. But this approach assumes a common Group Security Association previously agreed upon by the parties. In our paper, we tried to deal with this challenge using the ABE-based mechanism. Also, Porambage et al. [10] come up with a couple of elliptic-curve cryptography variants to manage the group key. But they also have some assumptions where the initiator of the message and the recipients know the public keys of one another and also make use of some handshake mechanisms. Interestingly, they also make use of Shamir's secret sharing mechanism and deriving the group key using Lagrangian interpolation, which is indeed the core concept of the ABE-based mechanisms. Their study results are also quite impressive and show a stable behaviour with the increasing number of nodes to 1000. Our results from the S-CPABE also co-relates to them with the stable decryption performance on the end nodes.

For secure multicasting requirements, intuitions would say to choose principles from ABE-based mechanisms with very little knowledge of the network in advance, and the same has also been observed from Porambage et al. [10]. There have been various other approaches to the best use of CPABE to distribute group secrets, handle multicast security issues, updation of group keys, and efficient key management techniques [15,16]. But to the best of our knowledge, there has been no previous approach to the segregation of CPABE to work in a modular fashion as our work suggests. The approach is not too complicated, but just some few tunings in this way give remarkable results as have been observed from our implementations. This, of course, shows the possibility for further ways of using ABE-based approaches.

3. BACKGROUND OF ABE MECHANISMS

This section briefly discusses the basics of ABE mechanism for better understanding of S-CPABE. ABE mechanisms are a generalized form of Identity-Based Encryption (IBE) [17] which has an identity for each and every member who is participating in the secure message transfer. An email-id or a unique personal identification number may be used as a unique id. Public key of a receiver is derived from the public parameter publicized to everyone and the identity. So, there is no need to transfer the public key of the recipient before the actual encryption of the message. In an ABE mechanism, each recipient has a set of attributes which define the characteristics or the identity of it. There are two different forms of ABE: CPABE [18] and KPABE [19]. We have chosen CPABE in this work according to which the encryptor encrypts the message with the help of the public parameter and an access structure which needs to be satisfied by the decryptor to decrypt the message. Each receiver has a private key given by the private key generator (PKG) which is used to decrypt the message, provided it indeed satisfies the access structure formed by the sender. The whole mechanism is based on Shamir's secret sharing mechanism [20]. The root key in the access structure is split down recursively to the leaves at the sender's side. At the receiver's side, the process is reversed and the root secret is re-generated bottom-up from the leaves recursively by the use of Lagrange's polynomial interpolation. The whole mechanism consists of four main phases, namely Set-up, Keygen, Encrypt, and Decrypt as discussed in [18].

Set-up(k) – (P, MSK): In the set-up phase, PKG accepts the security parameter k as input and outputs a public parameter P and a master secret key MSK. The public parameter P is distributed to all other members participating in the group, whereas the MSK is kept secret with the PKG only.

Keygen(MSK, S) – D: Keygen phase enables the PKG to authenticate each individual member of the group. Then it takes the MSK and the set of attributes S held by the member as input to output a private key D. PKG sends D to the corresponding member over a secure channel.

Encrypt(P, T, Msg) – CT: In the encrypt phase, the sender outputs the ciphertext CT, using the public parameter P, the access structure T with which it wishes to be satisfied by the target nodes, and the message Msg.

Decrypt(P, CT, D) – Msg: In the decrypt phase, if a member satisfies the access structure T, it takes in as input the public parameter P, the ciphertext CT and its private key D to extract the original message Msg back.

4. THE PROPOSED SCHEME S-CPABE

4.1. The Approach

Our approach is based on the ABE cryptographic mechanisms, particularly CPABE. Essentially the proposed scheme *S-CPABE* segregates the CPABE to push the intensive computations to the gateway for reducing the computational burden from low-power end devices.

In a typical IoT environment, we observe that the gateway nodes have huge responsibility in terms of caching and proxying. These gateways act as the principal node of contact for all the nodes in its range. It essentially acts as the guardian for the nodes involving filtering of unwanted data, forwarding the right data to the concerned node, allow nodes to sleep wherever possible to save power, apply security mechanisms as it is the basic point of entry in the constrained network, and also deals with possible caching and proxying capabilities. In a word, the gateway acts as a very important junction node. Since the IoT network involves a gateway for each constrained network, we focus on making use of the gateway capabilities. Also, the major fact is that the gateways are considered to be powerful resources in terms of computation, memory, and communication bandwidth running on an external power supply, and are thus capable of doing intensive operations. We particularly exploit this feature of the gateways.

According to our problem description, if a multicast secure message is to be sent over a network, it is not worthy to use a multiple unicast. Therefore, this approach (S-CPABE) encrypts the packet using CPABE encryption and uses IP multicast to send the encrypted packet to the destined network or nodes. So in this case, all the nodes which satisfy the access structure defined by the encryptor can only decrypt the packet. If we follow raw CPABE, each individual end node (which are indeed resource constrained) has to decrypt the packet using the decrypt function. This would not be possible for the end nodes because of their inability to deal with heavy cryptographic operations like CPABE. Moreover, if multiple nodes have the same attribute configuration and satisfy the access tree in exactly the same way, then the decryption in those nodes will be identical, so redundant. Thus if there is a message destined for them, then they will decrypt the message in exactly the same way. Due to this redundancy, there is a wastage of energy as decryption phase is quite heavy in terms of computation. So considering an IoT network involving hundreds of millions of nodes, the total decryption energy consumed by the end nodes become huge. S-CPABE reduces this consumption significantly by intelligently pushing the common intensive operations to the gateway.

4.2. S-CPABE (Segregated CPABE) Operational Details

In CPABE, the end nodes are solely responsible for the decryption and no intermediate nodes are involved or capable enough to decrypt correctly, so as to achieve end-to-end security. It makes good sense if these decryptions are pushed off to the gateway which is powerful enough to do intensive computations. But it destroys the requirement for end-to-end security unless the mechanism has been tailored properly. So essentially what we propose is a partial decryption approach in the gateway by segregating the CPABE decryption process. The gateway will do a major part of the decryptions but fails to perform the whole. The partially decrypted data is forwarded to the end nodes which then completes the rest of the decryption. Thus it prevents the gateway from extracting the plain text (message or key) to preserve end-to-end security, at the same time relieves the end nodes to perform the major computations to save energy.

S-CPABE uses all the operations including Set-up, Keygen, and Encrypt of CPABE [18]. But the Decryption process is divided into two phases to be carried out (a) partly by the proxy or gateway and (b) completed by the end nodes at the last step. All the operations are discussed as follows.

Set-up: Set-up phase is carried out by taking an input security parameter and outputs the public parameter PK and the master secret key MK. Typically PK = G_0, g, h =

g^{β}, $f = g^{1/\beta}$, $e(g, g)^{\alpha}$ and MK is (β, g^{α}). e is the bilinear mapping and G_0 is a bilinear group of prime order p with generator g. α and β are two random numbers in Z_p, the set of positive integers.

Keygen(MK,S): Keygen enables PKG to generate the private keys of each individual member using the master key MK and the attribute set S. The key is computed as

$$SK = \left(D = g^{(\alpha+r)/\beta},\ \forall j\epsilon S : D_j = g^r.H(j)^{r_j}, D_j^{'} = g^{r_j}\right). \tag{1}$$

Here r and r_j are randoms in Z_p and H is a function which maps an attribute to an element in the group.

Encrypt(PK,M,T): Sender encrypts the message M using the public parameter PK and the access structure T, as follows:

$$CT = \left(T, \tilde{C} = Me(g,g)^{\alpha s}, C = h^s,\ \forall y\epsilon Y : C_y = g^{q_y(0)},\right.$$
$$\left. C_y^{'} = H(\text{att}(y))^{q_y(0)}\right). \tag{2}$$

Here s is the secret at the root of the access tree, $q_y(0)$ is the secret at node y, and Y is the set of leaf nodes in T. All the rest of the symbols including $\tilde{C}$, C_y, C'_y, etc. are representations for different mathematical expressions used in the encryption and the decryption process.

Decryption Phase I – Partial decryption at the gateway:

The same mathematical construct as that of CPABE [18] is followed in S-CPABE for the decryption. The function is defined as:

$$\begin{aligned}\text{DecryptNode}(CT, SK, x) &= \frac{e(D_i, C_x)}{e(D_i^{'}, C_x^{'})} \\ &= e(g,g)^{rq_x(0)}.\end{aligned} \tag{3}$$

The DecryptNode function outputs the correct result for a node only if it satisfies the corresponding attributes in the access structure, otherwise returns $\perp$. The interesting aspect of this function is that for each node the same expression value $F_x = e(g,g)^{rq_x(0)}$ is the outcome. In phase I, we keep on doing this at the gateway until we reach one level from the root, namely the node A on the left of the access tree shown in Figure 1. Thus the gateway can recursively compute F_A as follows:

$$\begin{aligned}F_A = \textstyle\prod_{Z\epsilon S_x} F_z^{\Delta_i, S_x^{'}(0)}, i = \text{index}(z),\\ S_x^{'} = \{\text{index}(Z) : Z\epsilon S_x\}\\ = e(g,g)^{rq_A(0)}.\end{aligned} \tag{4}$$

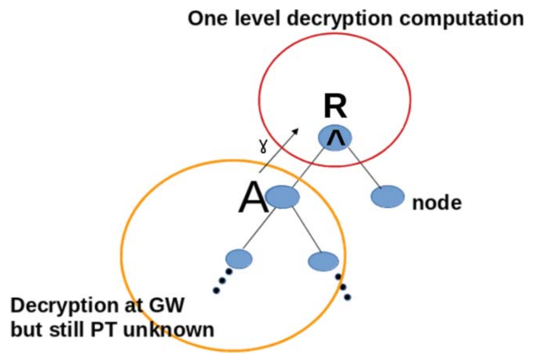

Figure 1: Segregation in S-CPABE

At the end of this phase, F_A is transferred securely as $\gamma = (CT, CT^1)$ using another level of CPABE (namely CPABE-2) to the end nodes. The ciphertext CT^1 is as in Equation (2) with corresponding access structure τ' as shown in Figure 2, which would be the logical OR of the node-ids of all the authorized nodes. This is to disallow the unauthorized end nodes to decrypt.

Decryption Phase II – Completion of the last decryption at the end nodes:

All the destined end nodes which satisfy the access tree τ' are able to decrypt CT^1 to obtain F_A. Further, each destined node satisfies the attribute “node” in 1, so they can satisfy this access tree after obtaining F_A. This attribute called “node” signifies that it is the last end node that does the final decryption. So essentially it is a sort of metadata or a tag that defines that it is an end node and not an intermediary node. So all the end nodes have the attribute “node”, but none of the intermediary devices whether gateways, proxies, routers, etc. have this attribute. This essentially preserves the end-to-end requirement as originally required by CPABE. So they can compute the final step of the decryption computation as in CPABE for the root node R with significant reduced computation:

$$\begin{aligned}F_R = \textstyle\prod_{Z\epsilon(A,\text{node})} F_z^{\Delta_i, S_x^{'}(0)}\\ = \prod \left(e(g,g)^{rq_{\text{parent}(z)}\text{index}(z)}\right)^{\Delta_i, S_x^{'}(0)}\\ = \left(e(g,g)^{rq_R(1)}\right)^{\Delta_1 S_1(0)}.\left(e(g,g)^{rq_R(2)}\right)^{\Delta_2 S_2(0)}\\ = e(g,g)^{r[q_R(1)\Delta_1 S_1(0)+q_R(2)\Delta_2 S_2(0)]}\\ = e(g,g)^{rq_R(0)}\\ = e(g,g)^{rs}.\end{aligned} \tag{5}$$

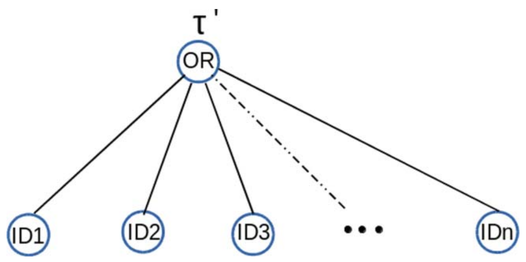

Figure 2: Access tree for CT^1

Now the message M or the group key can be computed as

$$\tilde{C}/(e(C,D)/F_R) \\ = \tilde{C}/\left(e\left(h^s, g^{(\alpha+r)/\beta}\right)/e(g,g)^{rs}\right) = M. \quad (6)$$

Note that each end node executed this process, once more earlier to obtain F_A from CT^1. Thus, each node requires two one-step CPABE decryptions for obtaining the plain text (group key).

4.3. S-CPABE for Secure Multicast in IoT

The proposed S-CPABE operations for secure multicast in IoT is as shown in Figure 3 and briefly explained as follows:

- Each node (server) with its attribute set registers to its gateway (proxy) during the deployment, before providing services.
- To extract the information confidentially from a group, user/requester generates a random key and encrypts it using CPABE. This key can be used by the servers to encrypt the information to be sent to the user. Similarly, to send sensitive information to a group of servers, the user can send the key encrypted by CPABE and message encrypted with this random key using a symmetric key encryption algorithm. Note that the access structure is constructed as A AND node, where "node" is the attribute that is present in all the end nodes under the gateway to participate in that multicast, but must not be in the gateway. So gateway cannot satisfy the access structure completely but satisfies until node A, which may be further nested down with another sub-level of the access tree.
- Gateway decrypts it partially (up to the level one less than root) and sends to the group of authorized servers/ nodes as discussed in the previous section.
- Each end node decrypts completely to extract the key. This key can then be used for sending the information or decrypting the information sent by the user/requester.

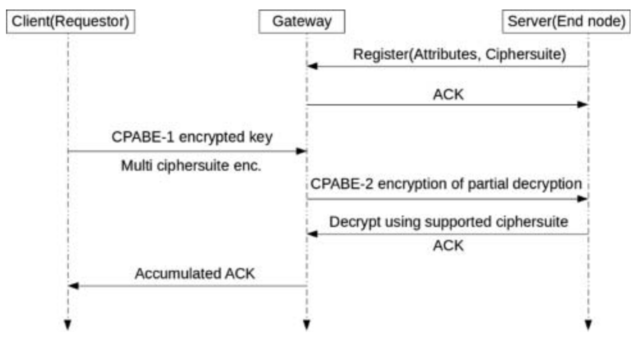
Client(Requestor)
Gateway
Server(End node)
Register(Attributes, Ciphersuite)
ACK
CPABE-1 encrypted key
Multi ciphersuite enc.
CPABE-2 encryption of partial decryption
Decrypt using supported ciphersuite
ACK
Accumulated ACK

Figure 3: Flow diagram of S-CPABE

4.4. Cost of Key Management in S-CPABE

Key management in S-CPABE is relatively simple. ABE-based systems have a great advantage that they need a one-time initialization of their private keys from the PKG after which it never needs to contact the PKG again. So in a sense if there are no more new members joining a group, the PKG can be removed from the system to make it even more secure. This one-time initialization can be done on a bootstrap to the network for the first time using any other protocol, or even a pre-shared key commissioned from the factory can be safely considered without the loss in generality or security. The next phase of the protocol for the encryption and decryption phases is even light-weight compared to other asymmetric protocols as the receivers do not need to explicitly send their public keys to the senders before they need to send any information. The public key of the receiver is derived from the combination of the public parameter PK and the access structure. This even reduces the number of message transfers for public key distributions. Also, the sender needs no knowledge *a priori* of the number of destined receivers, their specific identities, certificates, etc. Unlike DTLS-based mechanisms, there also needs no state maintenance like the tunnels. The sender thus has a very open possibility to define the target receivers based on their attributes which basically define the target group. This is thus the best suit for a multicast environment.

5. SECURITY MODEL

The security model of S-CPABE is based on CPABE which can be shown by the challenger and adversary concept or popularly known as the CK model.

- *Set-up*:The challenger runs the normal set-up algorithm and gives the public parameters PK to the adversary.
- *Phase 1*:The adversary chooses multiple sets of attributes and for each attribute set has a private key.
- *Challenge*:The adversary gives two equal length messages M_0 and M_1 and a challenge access structure A* to the challenger, so that A* is not satisfied by any of the attribute sets from phase 1. The challenger flips a random coin and encrypts M_b with A* and gives back the ciphertext CT* to the adversary.
- *Phase 2*:Phase 1 is repeated so that none of the set of attributes satisfies the access structure corresponding to the received challenge.
- *Guess*:The adversary outputs a guess b^1 of b. The advantage of an adversary in this game is obtained as $\Pr[b = b^1] = 1/2$. That is, even if the adversary knows the two possibilities of the solution message, still he cannot guess it deterministically by looking at the ciphertext. This aligns with the concepts of proof of CPABE [18].

6. EFFICIENCY OF S-CPABE

This section analyses S-CPABE in both security and computation aspects.

6.1. Security Analysis

The proposed mechanism is based on the segregation of CPABE. In this segregation, approach security of the proposed architecture is based on that of CPABE [18].

- *Curious Gateway:* Our model assumes that Gateway is trusted to perform all the operations as prescribed, but curious to learn the plain text (message or group key). It fails to extract the plain text (encrypted at the user/ requester) from *CT* as it fails to satisfy the access tree shown in Figure 1 due to the lack of the attribute "node".
- *Curious Revoked node:* It is simple to exclude a subgroup of nodes which are not destined for a particular message. Gateway forms the access structure shown in Figure 2 using the authorized destined nodes but excluding the revoked ones which might also have the attribute "node", but does not satisfy the sender's access tree. Thus the revoked nodes cannot satisfy the access tree τ' and fails to obtain F_A. Therefore, the revoked nodes cannot obtain the plain text.
- *Curious New Node:* Managing new node is also simple using S-CPABE. A new attribute (att) is assigned to this new node. Sender constructs the access structure similar to that of Figure 1 except that the right side of the Root would be (node OR att). Now the new node can extract the plain text further. But this new node fails to decrypt the previous messages as it can satisfy the access tree neither in Figure 1 nor in Figure 2.

6.2. Authentication

The original version of CPABE inherently supports authentication as the PKG only delivers the private key to the authenticated hosts. In the case of S-CPABE, also the private key distribution follows the same principles as that of CPABE and boils down to the same inherent properties of authentication. The only difference that arises is that the gateways also authenticate themselves along with the end nodes so that the forwarded message from the gateway to the end nodes in the legacy network is also authenticated messages.

6.3. Computational Analysis

We formed a test platform of working of S-CPABE to study its performance and compared with the traditional CPABE on the same set-up. For this study we have considered the end nodes as individual Linux PCs with the same configuration and comparing it with other implementations on the same set-up. So without loss of generality we can assume that if our approach works better than the others in the PC environment, then that should also hold in the constrained environment. Since the concern is the decryption overhead on the end nodes, the study focuses on the segregated decryption approach of SCPABE in comparison to the other protocols. According to our expectations we have got a good improvement over CPABE. At the same time, our results show that the S-CPABE approach is indeed an indication that ABE mechanisms can in fact be shipped to constrained environments as well if coded carefully.

The hardware set-up consists of a PC with Intel Core i3 1.7 Ghz CPU and 4 GB RAM running Ubuntu 14.04 LTS as the operating system. For the software, we have used the cpabe and libbswabe toolkit [18], running on the top of the pbc and gmp libraries. For the performance measurements, we have used the valgrind utility of Linux [21]. Our main optimizations for the design of S-CPABE were in the bswabe_dec function in the file core.c. The main motivation was to show the efficiency gain in the decryption strategy in the end nodes.

Table 1: Valgrind test results for CPABE

Phases	Latency (in instructions)	Memory (kB)
Set-up (PKG)	105.5 Mi	8.336
Keygen (PKG)	2 attrs = 220.3 Mi	12.9
	4 attrs = 440.4 Mi	14.73
	6 attrs = 661.3 Mi	16.68
Encrypt (Sender)	3 nodes (a and b) = 160.8 Mi	13.30
	7 nodes (a and b) v (c and d) = 267.9 Mi	15.34

Table 1 shows the valgrind results on our set-up for CPABE operations for Set-up, Keygen, and Encrypt. The latency has been measured in the number of (million) instructions and the memory requirements in (kilo) bytes. The Set-up and Keygen phases are done once by the PKG. As the PKG is considered to be unconstrained, so the memory and latencies as shown for these two phases do not affect the constrained environment. Our main focus in this paper has been on the decryption phase as most of the IoT-use cases for multicast are push based from a non-constrained device to thousands of low-power devices.

Table 2 shows the decryption times and memory requirements for S-CPABE and CPABE. It also shows the tunnel set-up times and memory requirements for DTLS and tinydtls. Since the working framework for symmetric key mechanisms like DTLS and asymmetric key mechanisms like ABE are different, hence comparing them has to be done carefully. DTLS creates and maintains tunnels from the source to the destination for the exchange of key parameters. The tunnel set-up time, namely the handshake, is time consuming and bandwidth heavy, but once the tunnel is formed and DH key is formed, the rest of the mechanism is light-weight as it is symmetric. So, in a sense, the DTLS protocol is heavy in terms of handshake communication, but light in terms of computation. On the other hand, the ABE mechanism follows a reverse topology. They are based on public key encryption mechanisms and hence heavy in terms of computation, but does not need the handshake phases as in DTLS and thus light in terms of communication and bandwidth usage. To this end, our focus was to keep them as far as possible on the same plate so as to get a good comparative result.

Table 2: Valgrind comparative results

Protocols	Latency (in instructions)	Memory (in bytes)
DTLS	200 Mi (Server) —— (Client) 180 Mi —— 20 Mi	223 kB (almost the same for server and client)
CPABE	ormid50^node dec = 158.4 Mi	54.45 kB
	5 node dec = 255.8 Mi	18.12 kB
S-CPABE	Std. 3 node dec = 155.7 Mi	16.97 kB
tinydtls	Server = 156 Ki	360 bytes <10 kB
	Client = 201.9 Mi	2.859 kB <10 kB

So we deal with the heavy sides of both of these protocols and measure the raw memory and time requirements for both of them. This makes sense as for the end nodes we would be interested on minimizing the total energy consumption on the nodes by reducing the memory usage, reducing the bandwidth usage and also reducing the computations. Hence we give the data results for memory and latency.

The DTLS of OpenSSL takes about 200 million instructions for both the client and the server calculated together. This is important as there needs to be formed one tunnel each for every participating entity, and we have to consider millions of these tunnels and their set-up times. Also, if tunnels are dropped somehow, then the same handshake or an abbreviated version of it has to be done again. For the memory requirements, both the client and the server reach close to about 223 kB. This considers the storage of the session key, epoch and sequence numbers, certificates, etc.

The CPABE and S-CPABE data are given next. We observe that the memory requirements of both of these are far below than that of DTLS. Of course the memory requirements in the decryption phase will depend on the access structure and the secrets for the individual nodes for storage. So the larger the access structure, larger the data storage required. But in typical cases, the access structure should not be required to be too deep to address a multicast group unless it is very complicated. But whatever be the depth of the access structure, we will show next that the memory requirement for S-CPABE has a constant value. For normal CPABE, we have taken two cases, one with an access structure consisting of an OR of 50 nodes and then an AND with the last standing attribute “node” and the other with five nodes of decryption operations starting bottom-up. The first one requires memory of about 54.45 kB, while the next one which of course has lesser storage for the access structure requires about 18.12 kB. Marvellously we get an even lesser memory requirement of 16.97 kB for S-CPABE, and more importantly, this is a constant value for all the decrypting nodes. For the latency parameters as expected in CPABE, the time increases for increase in the total number of decryption nodes. But in our approach, the latency for the decryption at the end node is 155.7 Mi which is a considerable value compared to the rising values with the tree size of normal CPABE. Moreover, this value is also constant for all decrypting end nodes.

Also, our test for tinydtls [22] using the same platform requires about 202 Mi for both the client and the server. If this is possible for constrained nodes, then definitely the margin of 200 Mi sounds good to work on S-CPABE as well. Coming to the memory requirements, tinydtls requires memory less than 10 kB, which also satisfies to work on the motes like telosb and micaz as these devices have RAM in these ranges. Compared to this, the S-CPABE might not suit with the result values we get for unconstrained devices, but of course it can be further minimized as we see that the value 16.97 kB slightly overshoots 10 kB. Thus it is highly expected that this study will lead to further minimizations in the memory requirements for ABE systems and be deployed on constrained devices.

7. DISCUSSION

We now give some important points related to the efficient and flexible use of S-CPABE over normal multicast DTLS. First of all, for DTLS to work there needs to be set up as many DTLS tunnels as the number of participating members in the group. Not only that these DTLS tunnels have to be retained till the members leave the group. This connection maintenance is a heavy task which is totally absent in S-CPABE. A change in the group members results in the unicast transmission of the group secret key to the end nodes in the order of the number of DTLS tunnels or the group size. But in S-CPABE, we need only a single-encrypted message multicasted over the IP network layer. Next, for the TGK (TEK Generation Key) distribution handshake is a must for DTLS but for S-CPABE, a one-time tunnel could be created or even better than that the private keys could be embedded from the manufacturing factory itself. The S-CPABE mechanism is thus far more flexible than the DTLS one. Next, as in the case of a multicast in an IoT environment, it would not be possible to address each and every node in the network. Rather it would be easier to classify nodes in the network and send a message destined for them. Using ABE mechanisms make the exact sense to do this instead of DTLS. DTLS is a stateful protocol and thus not scalable whereas because ABE mechanisms are stateless S-CPABE is highly scalable. Also, in DTLS, handling mobility is difficult as there needs to be maintained the tunnel from the group controller. But that is not the case in S-CPABE and thus the architecture perfectly suits to the IoT where mobility is inherent. In DTLS, as the group controller has the collection of all the session keys for all the group members, it becomes a central point of attack and thus highly vulnerable. But as in ABE mechanisms, the PKG can vanish after generating all the private keys for the group members, the architecture is much more secure. The memory and latency requirements discussed previously also testify to the fact that, in overall, the S-CPABE mechanism works better than that of the multicast DTLS.

7.1. Summary

The S-CPABE approach is an extension of the CPABE mechanism to off-load the heavy cryptographic operations to the gateway of a legacy network of constrained nodes. The approach keeps the basic security of CPABE intact, but fits the implementation so that it would be able to work on low-powered end nodes. In fact, following this approach, the S-CPABE mechanism gives a greater flexibility and generality to distribute the cryptographic operation load to other nodes in the network, which are capable enough to do those computations. In an IoT environment with so many heterogeneous nodes and a widely distributed environment, the concept makes good sense to distribute the computation to powerful nodes. Also, with the possibility of resource and service discovery mechanisms in IoT, the distributed nature of S-CPABE gives better chances to fit in, by pushing the operations to powerful nodes. Proxying and caching are another two important features heavily used in IoT deployments and hence the distribution of tasks into multiple parties is quite a common phenomenon which adds to the advantage of the design, S-CPABE also makes use of the same design mechanism. Group communication and its security are still a challenge that needs to be addressed in IoT, and S-CPABE gives a quite satisfactory design, the results of which clearly show that it is suitable for resources with very limited memory and without the support of hardware accelerators for cryptographic operations. The novelty of the design resides in its ability to support ABE-style encryption mechanisms to support multicast security and at the same time able to be run on resource-constrained devices by a simple method of off-loading.

8. CONCLUSION

Multicasting is a huge need for the IoT as devices are in millions, and to deliver the same message to a group of nodes in a network is a very common necessity. Traditional unicast mechanisms are not a suitable choice, as so many multiple unicasts would increase the network bandwidth and cause congestions, collisions, packet losses, delay, network jamming, etc., thus making the overall network unhealthily operable or inoperable. As DTLS is a one-to-one secure communication mechanism, the unicast fit for a multicast environment is not suitable for IoT. This work proposes a better approach named S-CPABE which takes the advantages of the ABE mechanisms and reduces the overheads of the ABE mechanisms from the end devices. The idea relies on maintaining the security aspects of ABE mechanisms but relieving the resource-constrained end nodes by pushing the resource-intense computations to the gateway. The merging of the flexibilities of ABE security mechanisms for multicast, on the one hand, and reducing the computations on the end nodes, on the other hand, give a perfect notion of achieving a multicast secure environment for the resource-constrained IoT devices. The whole idea has been implemented and the results obtained indeed show that they are well within the memory and computation constraints required by a typical IoT node. On the top, the approach is independent of any physical layer technology used and fits perfectly for upcoming 5G technologies. This implementation of S-CPABE clearly shows a direction that ABE mechanisms can indeed be applied in real-world IoT deployments.

REFERENCES

1. G. Kortuem, F. Kawsar, D. Fitton, and V. Sundramoorthy, "Smart objects as building blocks for the Internet of Things," *IEEE Internet Comput.*, Vol. 14, no. 1, pp. 44–51, Jan. 2010.

2. S. Singh, N. Saxena, A. Roy, and H. Kim, "A survey on 5G network technologies from social perspective," *IETE Tech. Rev.*, Vol. 34, no. 1, pp. 30–39, Jan. 2017.

3. D. Sahay and P. Ganesh, "Internet technology revolution," *IETE Tech. Rev.*, Vol. 14, no. 4–5, pp. 325–30, 1997.

4. H. Kopetz, "Internet of Things," in *Real-Time Systems*. New York: Springer, 2011, pp. 307–23. ISBN 978-1-4419-8237-7.

5. L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, Vol. 54, no. 15, pp. 2787–805, Oct. 2010.

6. R. H. Weber, "Internet of things – New security and privacy challenges," *Comput. Law Secur. Rev.*, Vol. 26, no. 1, pp. 23–30, Jan. 2010.

7. T. Heer, O. Garcia-Morchon, R. Hummen, S. L. Keoh, S. S. Kumar, and K. Wehrle, "Security challenges in the IP-based Internet of Things," *Wireless Pers. Commun.*, Vol. 61, no. 3, pp. 527–42, Dec. 2011.

8. F. Hamad, L. Smalov, and A. James, "Energy-aware security in M-commerce and the Internet of Things," *IETE Tech. Rev.*, Vol. 26, no. 5, pp. 357–62, Sep. 2009.

9. S. L. Keoh, S. S. Kumar, and H. Tschofenig, "Securing the Internet of Things: A standardization perspective," *IEEE Internet Things J.*, Vol. 1, no. 3, pp. 265–75, Jun. 2014.

10. P. Porambage, A. Braeken, C. Schmitt, A. Gurtov, M. Ylianttila, and B. Stiller, "Group key establishment for enabling secure multicast communication in wireless sensor networks deployed for IOT applications," *IEEE Access*, Vol. 3, pp. 1503–11, 2015.

11. N. Modadugu and E. Rescorla, *Datagram Transport Layer Security*. 2006. Available: https://tools.ietf.org/html/rfc4347

12. S. Keoh, O. Garcia-Morchon, S. Kumar, and S. Dijk, "DTLS-based multicast security for low-power and lossy networks (LLNs)," Work-in-progress, 2012.

13. A. Rahman and E. Dijk, "Group communication for coap," *Group*, 2011.

14. M. Tiloca, K. Nikitin, and S. Raza, "Axiom: Dtls-based secure iot group communication," *ACM Transa. Embedded Comput. Syst. (TECS)*, Vol. 16, no. 3, p. 66, Jul. 2017.

15. L. Cheung, J. A. Cooley, R. Khazan, and C. Newport, "Collusion-resistant group key management using attribute-based encryption," *Group-Oriented Cryptogr. Protocols*, p. 23, 2007.

16. D. Huang and M. Verma, "Aspe: Attribute-based secure policy enforcement in vehicular ad hoc networks," *Ad Hoc Netw.*, Vol. 7, no. 8, pp. 1526–35, Nov. 2009.

17. D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology–CRYPTO 2001*. Springer, 2001, pp. 213–29.

18. J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *IEEE Symposium on Security and Privacy, SP'07*, 2007, pp. 321–34. doi:10.1109/SP.2007.11

19. C.-J. Wang and J.-F. Luo, "A key-policy attribute-based encryption scheme with constant size ciphertext," in *2012 Eighth International Conference on Computational Intelligence and Security (CIS)*, Guangzhou, China, 2012, pp. 447–51.

20. A. Shamir, "How to share a secret," *Commun. ACM*, Vol. 22, no. 11, pp. 612–13, 1979.

21. J. Seward, N. Nethercote, and J. Fitzhardinge, "Valgrind, an open-source memory debugger for x86-gnu/linux," Available: http://www.ukuug.org/events/linux2002/papers/html/valgrind, 2004.

22. O. Bergmann, "Tinydtls," Available: http://tinydtls.sourceforge.net/.Visited, 2013.

Authors

Subho Shankar Basu received his masters degree in computer science from Indian Institute of Technology, Patna. His research interests are centred around low-power resource-constrained networks for the Internet of Things, dealing with their security and privacy issues and protocol optimization areas.

E-mail: shankar.mtcs13@iitp.ac.in

Somanath Tripathy received his PhD degree in computer science and engineering in 2007 from Indian Institute of Technology Guwahati. At present, he is an associate professor in Indian Institute of Technology, Patna. His research interests include light-weight cryptography, network security, security and privacy issues in IoT and Cloud system. He has published more than 50 research papers in peer reviewed journals and conferences. He is a senior member of IEEE and life member of Cryptology Research Society of India.

Corresponding author. **E-mail:** som@iitp.ac.in