International Journal of Communication Networks and Information Security (IJCNIS) Vol. 14, No. 1, April 2022

Lightweight Scheme for Smart Home Environments using Offloading Technique

Ahmad A. Al-daraiseh[1], Rasel Chowdhury[2], Hakima Ould-Slimane[3], Chamseddine Talhi[4], and Mohamammed Bany Taha[5]

[1,5]Faculty of IT, American University of Madaba, Jordan

[2,3,4]Department of Software Engineering and Information Technology, Ecole de technologie sup´erieure, Canada´

Abstract: Internet of Things (IoT) as an emerging technology has been transforming the different aspects of our world from simple preprogrammed coffee machine to smart farming. Due to the human nature that strives for more convenience, humans are becoming more dependent on these automated IoT devices and smart environments such as smart phones, wearable devices, smart homes etc. In order to provide better QoS, these devices need to work together and share data internally and with different service and cloud providers. Since these devices are resource constrained, IoT technology heavily depends on the cloud for processing, analytics and storage. Data produced and shared by such devices may contain lots of personal identity information (PII). Usually, the users of these devices are unaware of the sensitivity of information that is being transmitted or do not possess control over the data that is being sent to the service provider, or to the cloud. Although, the cloud services and service providers are supposed to be very secure, and there is a number of security measures implemented to secure end to end communications, IoT lacks the mechanism for securing the data generated by different devices and for proper access control. In this article we are proposing an approach for the security, privacy and access control of users' data using Attribute Based Encryption (ABE). Smart homes are used as a case study.

Keywords: Data privacy preserving, Attribute-Based Encryption, Encryption Outsourcing, Applied cryptography, Smart Home

1. Introduction

The Internet of Things (IoT) is one of the most popular of all the emerging technologies due to the increasing demand for smart devices and the availability of robust cloud services and internet connectivity. In the majority of applications, IoT devices are resource constrained to cut on cost. Cloud Computing platforms play a crucial role to satisfy the massive and dynamic demand produced by different resources constrained devices. IoT applications exist in many different fields of which many are focused on human convenience [1]. In this paper, smart home is considered as a case study. Smart home is part of IoT domain where a house is automated with the help of sensors, actuators and smart devices ranging from light control, HVAC systems to intrusion detection systems which allow the home owner to have a great deal of control over the house.

In a smart home environment, a large amount of data is received and sent to and from different sensors and devices. Also, many commands are communicated to control different devices such as coffee machines, garage door, lights and many others. The data generated by different devices may contain personal and confidential information also known as Personally Identifiable Information (PII) [2], which can be used to interpret the behavior of the persons living in the smart home.

To leverage the popularity of IoT smart devices, Cloud and internet connectivity, many third-party service providers were established to provide different services based on the needs of the smart home residents. As shown in figure 1, where data from different devices is sent to a smart home gateway which relays it to the Cloud where a third-party service provider can access it. Several technologies emerged for the communications of IoT devices such as 6LoWPAN, BLE, ZigBee, XBee, CoAP [3]. The majority of IoT devices and technologies are prone to security flaws. [4]. Due to its privacy and sensitivity, it is very clear that IoT's data needs to be very well secured.

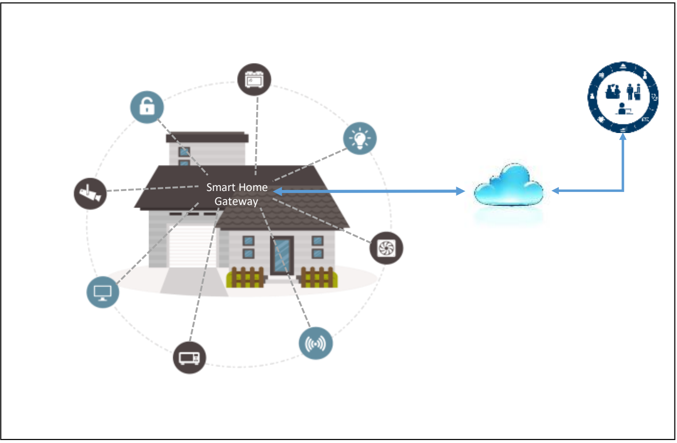


Figure 1. Generic smart home

Attribute Based Encryption (ABE) is a more suitable solution to provide the privacy and access control for smart home's data than other cryptography solutions [5]. None the less, ABE is known to be computationally heavy, and hence, applying it to provide the required privacy and confidentiality of this data will put more pressure on the smart devices in use. Knowing that the majority of IoT's devices are constrained devices, applying ABE on such devices is big challenge. Moreover, in smart homes, smart devices transfer data very frequently and this makes the situation more challenging. In this article, we provide fine grained access control to smart home environment using ABE to guarantee the confidentiality of the data before it is sent to the cloud. In our architecture, we assume that smart home components do not have the resources for performing costly encryption, and hence, the data is encrypted partially at the smart home gateway and most of the computation cost is delegated to a proxy server. In our architecture, we guarantee that the data cannot be revealed by the proxy or the Cloud platform that host the data. The rest of the paper is organized as follows: Section 2. discusses the related literature where ABE schemes were proposed for use on constrained devices. After which, in section 3, concepts of

CP-ABE and KP-ABE are discussed. In section 4, the proposed architecture and results are presented.

2. Related Work

Recently, a large number of researchers discussed the security issue in smart home environment [6] [7] [8]. Singh et al discussed the twenty important security considerations for IoT [9]. The authors mainly focused on access control and encryption of IoT device and considered how it would add more complexity to the existing solutions. Lin et al [10] discussed the privacy issues and security challenges in smart home environment such as confidentiality and access control. The authors showed few examples of vulnerabilities to be considered.

In 2006 Goyal et al proposed the first Key Policy Attribute Based Encryption (KP-ABE) scheme [11]. In KP-ABE the data is encrypted with attributes to generate the cipher-text (CT) and the client's Secret Key (SK) contains the access policy. If the attributes in CT satisfy the access policy in SK. This kind of ABE allows the authority that generated SK to determine who can decrypt CT.

In 2007, Bethencourt et al proposed the first Cipher-text Policy Attribute Based Encryption (CP-ABE) [12]. In this scheme, the data is encrypted with an access policy chosen by the user (encryptor) to generate the CT, and the secret key contains a list of attributes. The client's secret key SK can decrypt CT if and only if its attributes satisfy the policy that was chosen by the encryptor.

Ambrosin et al illustrated the performance of ABE schemes in IoT devices like Raspberry Pi, Intel Edison etc [13] using two types of ABE namely, CP-ABE and KP-ABE. The authors showed how performance and efficiency of the two schemes are different on different boards, in addition, the authors proved that the efficiency of these schemes is very weak on IoT devices (smart home components) comparing with regular computers (desktop computer). Moreover, their experiments showed that the performance of ABE scheme depends on several factors such as number of attributes and the cryptography curve used in the scheme. Yao et al [14] proposed KP-ABE scheme for IoT devices. The scheme based on Elliptic Curve. It has no bilinear pairing which explain the results presented by the authors compared to standard ABE scheme. However, their results showed that the computation increased as the number of attributes used to encrypt the data with increased. Thus, in case we have large number of attributes (i.e, $>$30) the scheme will be impractical. Tauati et al [15] proposed CP-ABE scheme for constrained devices. The scheme offloads the computation overhead to the nearest trusted node. Such nodes or devices called assistant nodes. The assistant nodes are unconstrained devices that can perform the encryption part on behalf of the constrained nodes. Chowdhury et al [16] integrated ABE with well-known smart home middleware openHAB and experimented with different settings using a test scenario.

Zhou et al [8] proposed CP-ABE for mobile cloud environment to reduce the computation cost on mobile devices; the authors idea is that each access tree consists of left and right sub-tree. Based on their assumption the right sub-tree usually has leave nodes (attributes) less than the left sub-tree. They took advantage of this assumption and encrypted only the right sub-tree on mobile device as it is a constrained device (according to the authors) and left the left sub-tree to be encrypted somewhere else such as a proxy server. In this scheme, the mobile device performs a small part of the cryptographic operations as the right access tree has less attributes than the left access tree to generate CT_{DO}. The proxy server performs CP-ABE encryption on left sub-tree to generate CT_{EPS} and combine it with CT_{DO}. The final CT will be CT = $CT_{DO} \wedge CT_{EPS}$. Zhou et al assumed that the root node is usually an AND gate so two ciphertext cab be merged by this AND gate. In fact, the root node could be an OR or an AND which means this scheme is restricted to limited applications. To fix this problem, Jin et al [17] proposed a new scheme that dealt with the restriction of Zhou's scheme. Jin et al proposed the idea of dummy attributes as the right sub-tree, in this case the real access tree will be on the left sub-tree only and the dummy attributes in the right sub-tree only. The constrained device will encrypt the data with the dummy attribute and the proxy will perform the cryptography operations of the left sub-tree. The data owner will not be revealed since it is encrypted with dummy attributes. The authors assume that every user has dummy attributes. Several schemes were proposed based on dummy attributes idea such as [18] [19] [20].

3. Preliminary

3.1 Ciphertext Attribute Based Encryption (CPABE)

CP-ABE is a form of ABE, the user encrypts her data with an access policy. A client interested to decrypt this data must have a secret key that satisfy the policy in CT. Figure 2 show visual representation of CP-ABE. CP-ABE performs encryption and decryption in four steps as follows: **Setup** → (PK,MSK): Setup algorithm takes security parameters to generate Public Key (PK) and Master Secret Key (MSK). PK is available for any user and is used as an input for the encryption algorithm. MSK is used to generate Secret Key (SK) through the key Generation algorithm. KeyGeneration

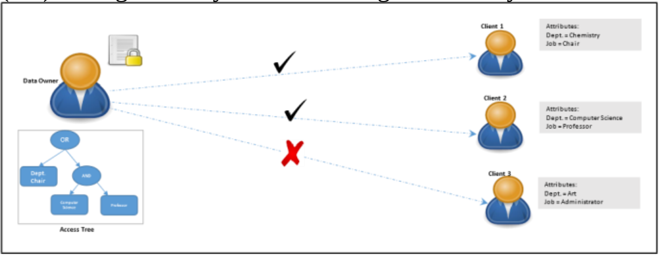


Figure 2. CP-ABE

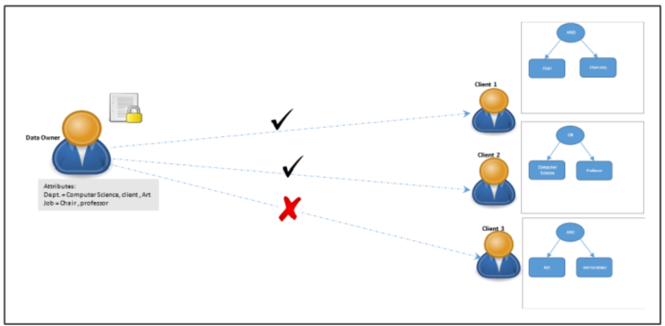


Figure 3. KP-ABE

(PK, MSK, ω)→ SK: KeyGeneration take the PK, MSK, and ω as input. ω is a list of attributes of the user. The output of this algorithm is the secret key SK. SK is unique for each user.

Encryption (PK, M, α)→ CT: In this algorithm the user encrypts his/her data with α where α is the access policy. The output of this algorithm is the ciphertext(CT). **Decryption (CT, SK)→ M**: In this algorithm the client uses her secret key to recover the message.

3.2 Key Policy Attribute Based Encryption (KPABE)

KP-ABE is the second form of ABE; the user encrypts her data with a list of attributes. The secret key associated with the access policy thus the trust authority who generate the SK will decide who encrypt the data. Figure 3 shows visual representation of KP-ABE. The following steps explain the main four algorithms of KP-ABE.

Setup → (PK,MSK): Setup algorithm uses security parameters to generate PK and SK.

KeyGeneration (PK, MSK, α)→ SK: KeyGeneration algorithm used to generate SK. The input of this algorithm PK, MSK, and the α. The algorithm generate SK.

Encryption (PK, M, ω)→ CT: In this algorithm the user encrypts the data using ω to generate CT.

Decryption (CT, SK)→ M: In this algorithm the client uses his/her SK to decrypt CT. If the attributes that the CT is associated with satisfy the policy that the SK is associated with then the client can decrypt CT and recover the message, otherwise the client will not be able to recover the message.

3.3 Access Tree

Several access trees were proposed for ABE schemes. As was mentioned previously, in CP-ABE the user encrypts the data with access policy where the secret key of the client is associated with a list of attributes. In our scheme we use the access structure proposed in [17]. To briefly review the construction of an access tree let's assume T is the access tree. T consists of multiple nodes at different levels, each non-leave node is either an AND or an OR gate, the leave nodes are the attributes. If the client's secret key has attributes that satisfy the access tree then the client will be able to decrypt the data. To prevent collisions, the Trusted Authority (TA) generates a random value to blend with each attribute in the secret key. Figure 4 shows the construction of access tree that is proposed in [17].

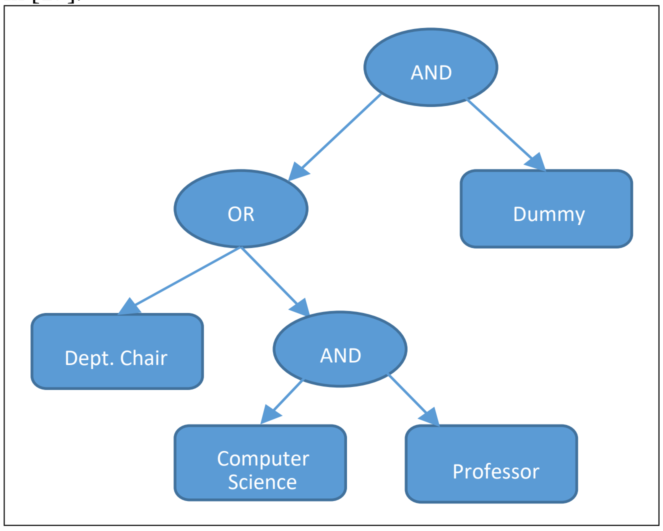


Figure 4. Access Tree

3.4 Cryptography curves

Wang et al [21] experimented with different types of curves with ABE and evaluated their performance. In this article, two kinds of curves are used. The first one is super singular curve which is a symmetric curve and the second one is asymmetric one called MNT.

4. The Proposed Architecture

Figure 5 shows our architecture for smart home systems. The architecture is divided into six modules as follows:

Data Collector Module (DCM): this module is used for receiving data from the sensors. DCM acts as a middleware in the smart home gateway. The module runs multiple threads for accepting connections from the sensors, then it verifies and authenticates the sensors' information and prepares the data into a specific format then forwards to Gateway Encryption module of the home gateway.

Gateway Encryption Module (GEM): this module is responsible for encrypting the data coming from the DCM. When some data is received from the DCM, it checks a database to get the dummy attribute associated with sensor that generated the data. Then the module encrypts it with the dummy attribute and sends it to the Proxy Encryption Module (PEM).

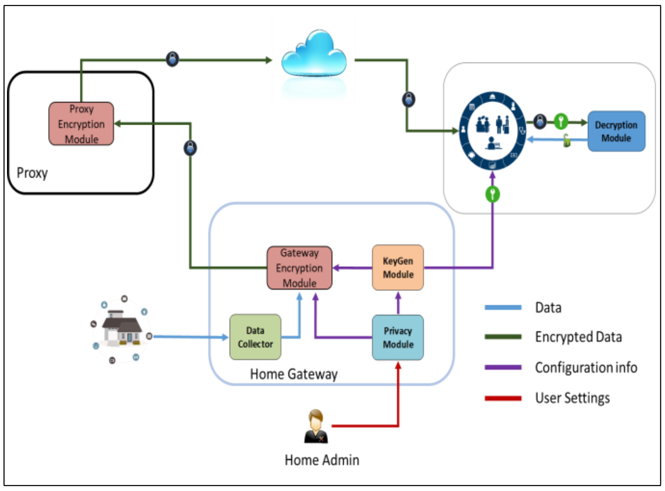


Figure 5. Proposed Architecture

Proxy Encryption Module (PEM): this module is responsible for encrypting the data from GEM using the actual policy/attributes that were set by the owner. PEM checks the message for the sensors' policy and uses that policy to encrypts the data and send it to the Cloud.

KeyGen Module (KM): this module is responsible for generating the PK, MK which are required by the GEM for encrypting the data and SKs for the services for decrypting. During generating the SK the KM incorporates the dummy attributes as well.

Privacy Module (PM): this module interacts with the admin for setting up the primitives required by the GEM and KM.

Decryption Module (DM): this module decrypts the ciphertext using the secret key.

Python was used for the implementation of the modules charm [22] [23] was also utilised for its crypto modules. The data from the sensors are converted to bytes before it is fed into the GEM. Serialization and de-serialization were used when passing CT around to make it harder for an attacker to break it. In order to evaluate the performance of this framework, a custom data collector (DCM) which serves as a middleware was utilised. this way, this framework can be used as plug and play with any other smart home middleware. Transmission Control Protocol (TCP) was used for its reliable data transfer. Different types of sensors placed in different locations of the house were used to simulate a smart home. All the sensors send data to the gateway, which is in our case, a Raspberry Pi which acts as the DCM and GEM and, where, a desktop PC is used to act as the PEM. The configurations of the gateway and desktop is displayed in table 1 and 2. Table 3 shows the type of data generated by the sensors and their identifications. this architecture was evaluated with different types of ABE. One dummy attribute is used for GEM which is a unique attribute assigned to each sensor. Based on access, those dummy attributes are assigned for the key generation. In addition, three services will be used that have access to the sensor's data based on the ABE policy. The configuration for CP-ABE settings CP-ABE is shown in table 5 and in table 4 is for KP-ABE.

Table 1. Hardware and Software Specifications of Gateway

Processor	1.2GHz 64 bit quad-core ARMv8
RAM	1GB
Storage	16GB eMMC flash Storage
Operating System	Raspberian Debian OS

Table 2. Hardware and Software Specifications of Proxy

Processor	3.2GHz Ci7
RAM	16GB RAM
Storage	320GB
Operating System	Ubuntu

Table 3. Sensor Information and data types

Sensor type	Sensor ID	Data type
Light	Light1....Light8	String [Dim, Bright, Very Bright]
Temperature	Temp1....Temp7	Float [2 decimal places]
Contact [Door]	Cont1....Cont8	String [Open/close]
Smoke Detection	Smoke1....Smoke4	String [low, medium, heavy]
Water Flow	Water	Integer
Electricity consumption	Elec	Integer
Gas Flow	Gas	Integer

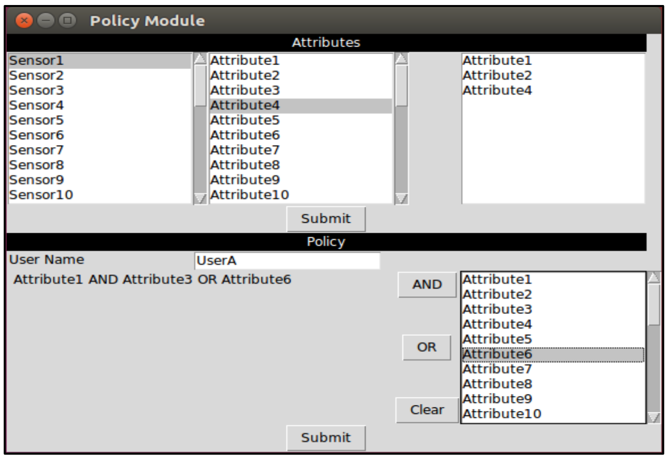
Table 4. KP-ABE settings

Sensor ID	Dummy Attribute	Actual Attribute
Light1	D1	Attribute1 .... Attribute10
Temp1	D9	Attribute1 .... Attribute10
Cont1	D16	Attribute1 .... Attribute10
Smoke1	D24	Attribute1 .... Attribute10
Water	D28	Attribute1 .... Attribute10
Elec	D29	Attribute1 .... Attribute10
Gas	D30	Attribute1 .... Attribute10

Home admin is responsible for setting up the attributes and policies required for the KM and PEM by using the PM. PM is a graphical user interface as shown in figure 6 for KP-ABE setup where the home admin can select the attributes for the sensors and write policies for the services for generating the secret key. For CP-ABE the interface is reversed; attributes for services and policy for the sensors. KG is responsible for setting up the environment by generating the PK and MK, and then transferring them to the GEM. KM is also responsible for generating the secret key of the services and transmitting to the designated service in a secured manner. DCM acts a smart home middleware and collects data from the sensors and forwards it to the GEM. GEM is responsible for encrypting the data using one dummy attribute/policy and then the data is transferred to the PEM. PEM does the encryption with actual attributes/policy which is set by the home admin and then the encrypted data is stored in the Cloud. The services can only decrypt the data from the Cloud if they have the decryption module running in their system and have a valid key, which satisfies the requirements of ABE. Figure 7 shows the view of different modules and services. On the left hand side of the Figure 7 is the visual representation of DCM and the rest are different services that have access to different sensors. If a service has access to a specific sensor it can view the value otherwise an error message is shown based on the access policy of the service.

Table 5. CP-ABE settings

Sensor ID	Dummy Policy	Actual Policy
Light1	D1	Attribute1 AND Attribute2 AND ... Attribute10
Temp1	D9	Attribute1 AND Attribute2 AND ... Attribute10
Cont1	D16	Attribute1 AND Attribute2 AND ... Attribute10
Smoke1	D24	Attribute1 AND Attribute2 AND ... Attribute10
Water	D28	Attribute1 AND Attribute2 AND ... Attribute10
Elec	D29	Attribute1 AND Attribute2 AND ... Attribute10
Gas	D30	Attribute1 AND Attribute2 AND ... Attribute10



Sensor1
Sensor2
Sensor3
Sensor4
Sensor5
Sensor6
Sensor7
Sensor8
Sensor9
Sensor10

Attribute1
Attribute2
Attribute3
Attribute4
Attribute5
Attribute6
Attribute7
Attribute8
Attribute9
Attribute10

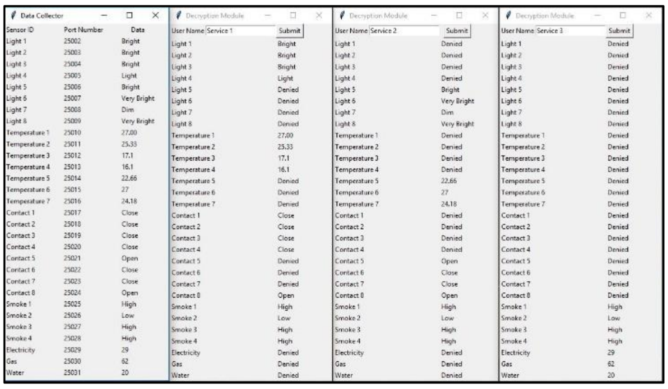
Attribute1
Attribute2
Attribute4

User Name UserA

Attribute1 AND Attribute3 OR Attribute6

Attribute1
Attribute2
Attribute3
Attribute4
Attribute5
Attribute6
Attribute7
Attribute8
Attribute9
Attribute10

Figure 6. Policy Module



Sensor ID	Port Number	Data
Light 1	25002	Bright
Light 2	25003	Bright
Light 3	25004	Bright
Light 4	25005	Light
Light 5	25006	Bright
Light 6	25007	Very Bright
Light 7	25008	Dim
Light 8	25009	Very Bright
Temperature 1	25010	27.00
Temperature 2	25011	25.33
Temperature 3	25012	17.1
Temperature 4	25013	16.1
Temperature 5	25014	22.66
Temperature 6	25015	27
Temperature 7	25016	24.18
Contact 1	25017	Close
Contact 2	25018	Close
Contact 3	25019	Close
Contact 4	25020	Close
Contact 5	25021	Open
Contact 6	25022	Close
Contact 7	25023	Close
Contact 8	25024	Open
Smoke 1	25025	High
Smoke 2	25026	Low
Smoke 3	25027	High
Smoke 4	25028	High
Electricity	25029	29
Gas	25030	62
Water	25031	20

User Name	Service 1	Service 2	Service 3
Light 1	Bright	Denied	Denied
Light 2	Bright	Denied	Denied
Light 3	Bright	Denied	Denied
Light 4	Light	Denied	Denied
Light 5	Denied	Bright	Denied
Light 6	Denied	Very Bright	Denied
Light 7	Denied	Dim	Denied
Light 8	Denied	Very Bright	Denied
Temperature 1	27.00	Denied	Denied
Temperature 2	25.33	Denied	Denied
Temperature 3	17.1	Denied	Denied
Temperature 4	16.1	Denied	Denied
Temperature 5	Denied	22.66	Denied
Temperature 6	Denied	27	Denied
Temperature 7	Denied	24.18	Denied
Contact 1	Close	Denied	Denied
Contact 2	Close	Denied	Denied
Contact 3	Close	Denied	Denied
Contact 4	Close	Denied	Denied
Contact 5	Denied	Open	Denied
Contact 6	Denied	Close	Denied
Contact 7	Denied	Close	Denied
Contact 8	Open	Open	Denied
Smoke 1	High	High	High
Smoke 2	Low	Low	Low
Smoke 3	High	High	High
Smoke 4	High	High	High
Electricity	Denied	Denied	29
Gas	Denied	Denied	62
Water	Denied	Denied	20

Figure 7. Screen shot of data collector and different services

5. Experimentation and Evaluation

In order to evaluate this framework, a number of experiments were carried out and the impact on CPU, Memory, and latency of the KP-ABE, CP-ABE and a revised version of KP-ABE was monitored. In these experiments the following values were used: 30 attributes/policy, 30 sensors and a max of 150 sample for the ABE settings. Different KP-ABE and CP-ABE schemes were tested using this architecture to check which scheme is more suitable in smart home scenario. Moreover, super singular curve and MNT curve were used. A python script was used to determine the resource utilization in all experiments. To calculate the latency, the time at which the data enters the DC and the time at which the encryption is fully completed were recorded and the difference provided the latency. Also, a USB tester was used to calculate the power consumption of the Raspberry Pi.

Figure 8 shows the resource utilization (CPU, Memory and gateway (Raspberry pi), part by the gateway and part by the Latency) when using CP-ABE with different offloading plans. proxy, and finally, all the encryption by the proxy server.

Although the least load is given when all the encryption is done by the proxy, this plan should be avoided as the proxy server can’t always be trusted. so the next best is to have part of the encryption done by the local gateway and part by the proxy server(PEM). Figure 9 shows the frequency of data sent to the middleware based on the number of sensors and their rate of sending data. In this experiment, 30 sensors were used and their data sample rate varied from 3 to 12 per minute. Figure 10 shows the size of the cipher text CT using a constant message of 10 characters with varying the attributes/policy. it is clear that YCT achieved the least size.

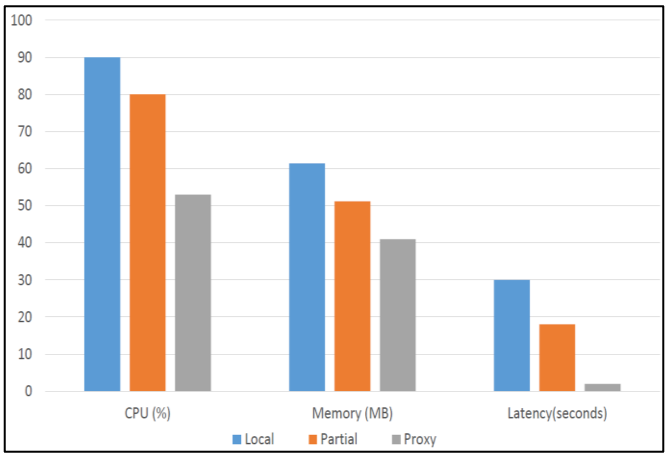


Figure 8. Comparison of offloading techniques

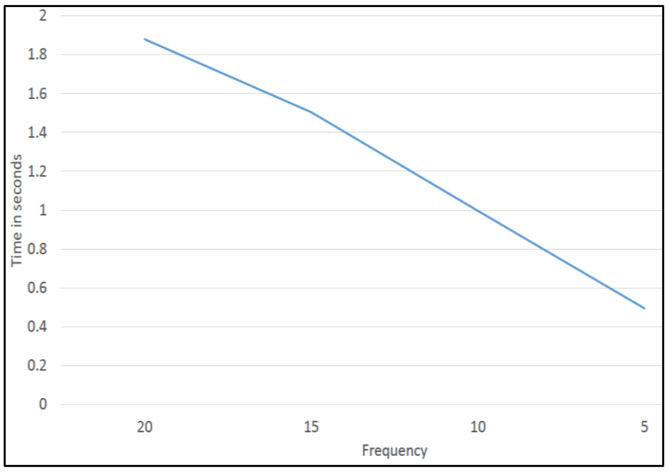


Figure 9. Data interval

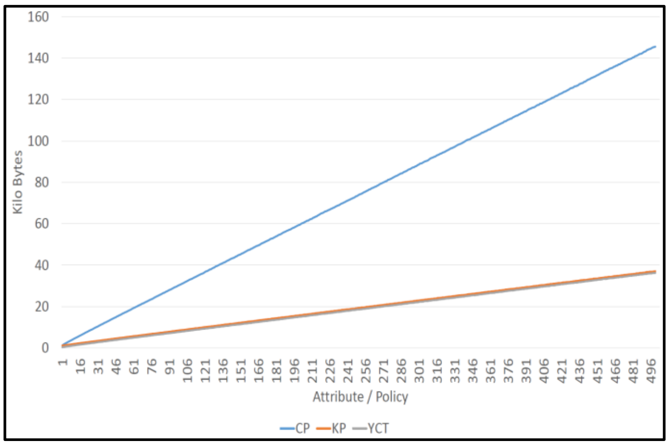


Figure 10. Ciphertext size

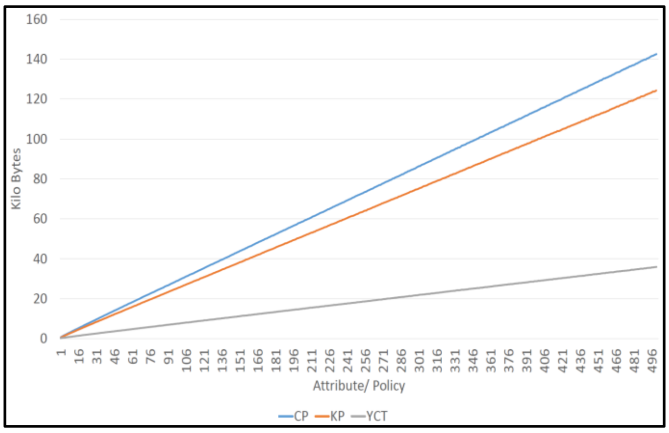


Figure 11. Secret Key size

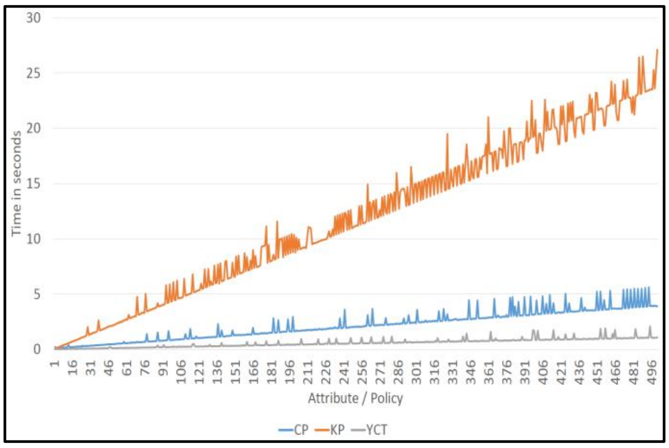


Figure 12. Key generation Time

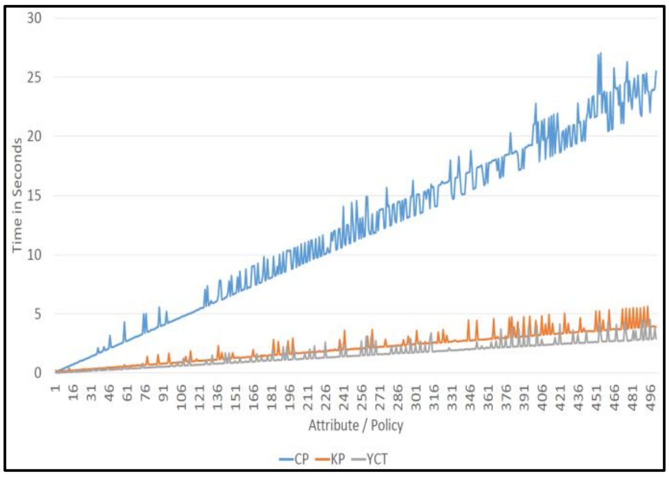


Figure 13. Full encryption

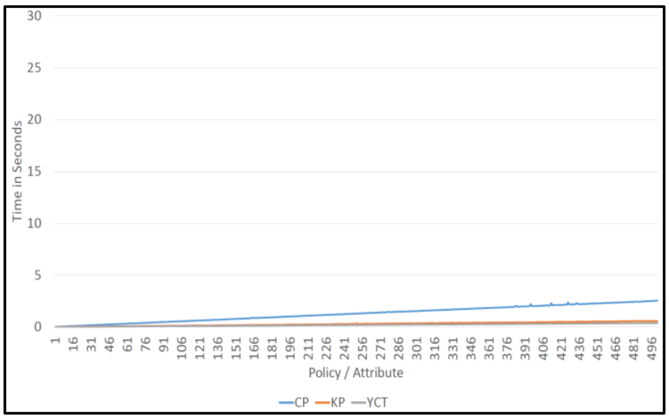


Figure 14. Offloaded encryption

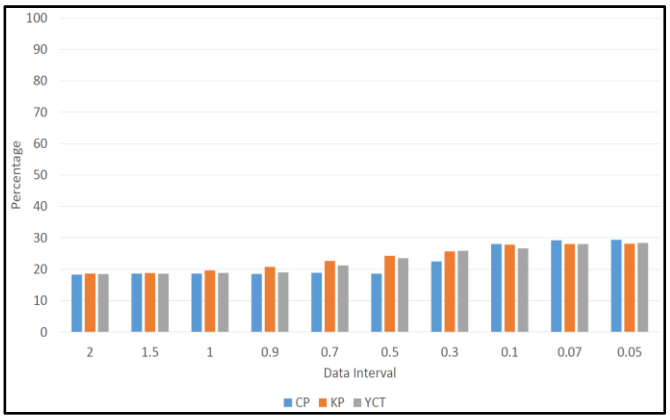


Figure 18. Offloaded encryption memory utilization

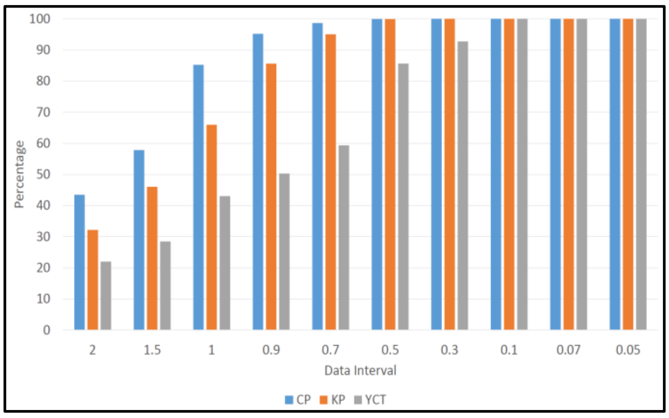


Figure 15. Full encryption CPU utilization

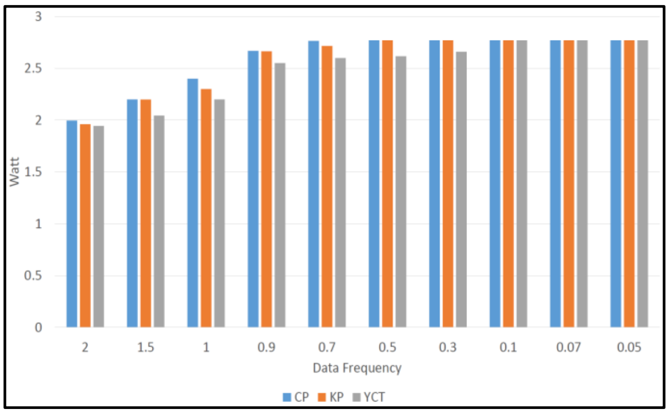


Figure 19. Full encryption power consumption

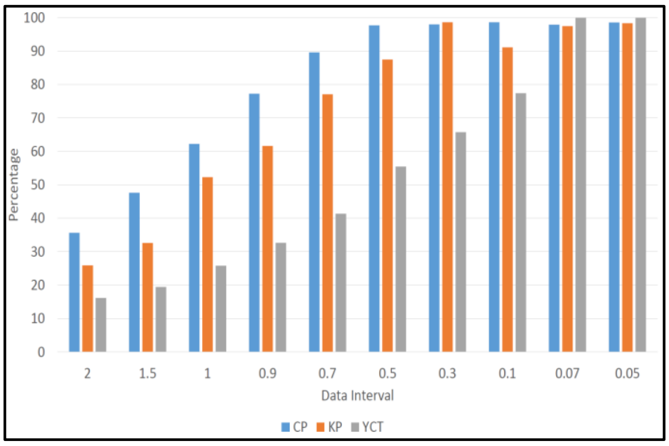


Figure 16. Offloaded encryption CPU utilization

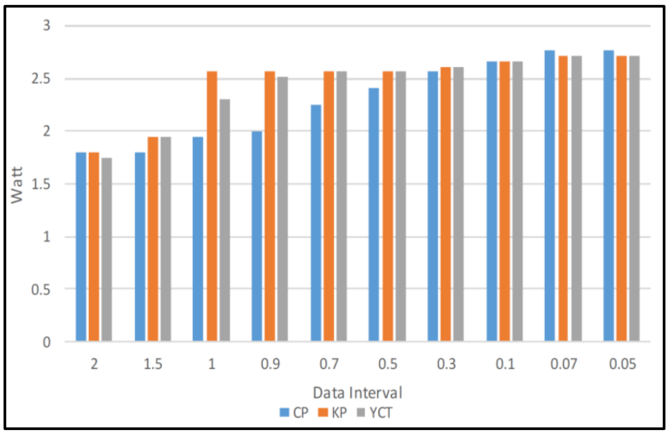


Figure 20. Offloaded encryption power consumption

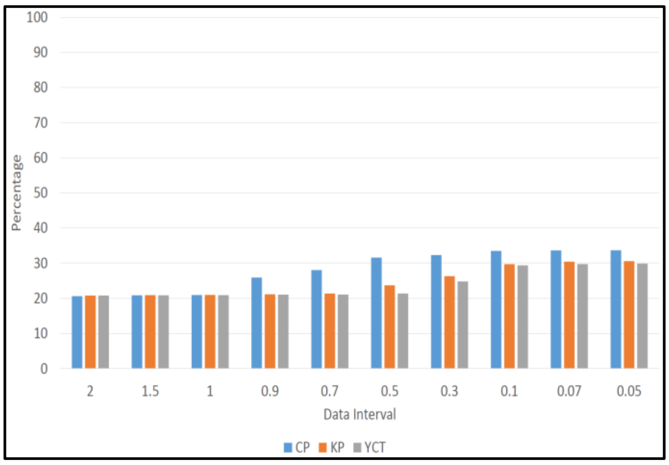


Figure 17. Full encryption memory utilization

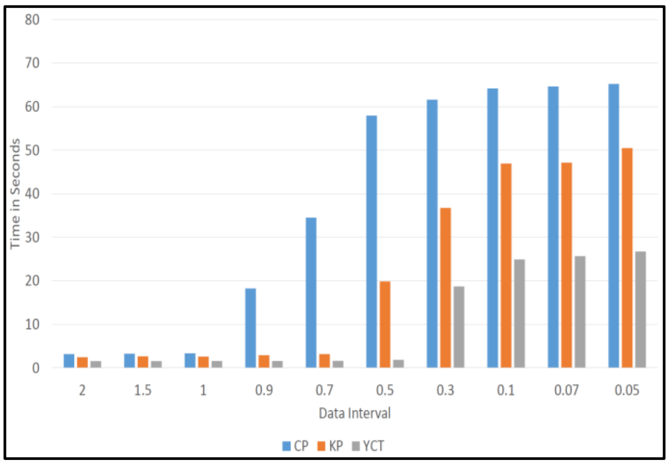


Figure 21. Full encryption latency

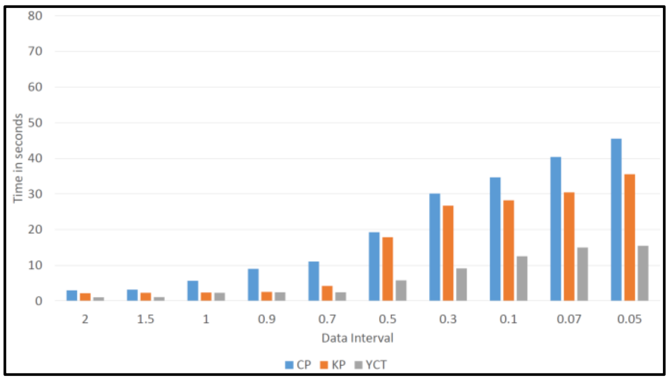


Figure 22. Offloaded encryption latency

Figure 11 displays the secret key size with different attributes/policy. Again YCT achieved the least size for the key. Figure 12 shows time required for key generation vs number of attributes (from 1 to 500) at the gateway. It is also clear here that YCT took the least amount of time in the majority of experiments. Figures 13 and 14 display the execution time of full encryption and offloaded encryption. it is very clear that the time is minimal when offloading is used. Also, YCT takes less time in both cases. Figures 15, 16, 17 and 18 show the resource utilization vs sample rate. Data interval of 2 seconds to 0.05 seconds were used with both full and offloaded encryption. the Figures show that with full encryption data interval up to 0.7 can be achieved. beyond 0.7 both CP and KP had the CPU at full capacity. on the other hand, when offloading is used even when the data interval is 0.05 both CP and KP didn’t reach 100% however, YCT did at both 0.07 and 0.05 intervals. Figure 19 and 20 display the power utilization of the gateway vs sample rate. Less energy was consumed by the gateway when offloading was done as expected. Figure 21 and 22 show the latency of the system vs the data interval.

From the results of figure 9, we see that the interval of data coming increases from 2 seconds to 0.5 seconds when the frequency increased from 20 seconds to 5 seconds. In Figure 8 we can see that if the system performs all the computation at the remote server, the resource utilization and latency are decreased by 40%. Figure 12 shows that as the number of policy/ attributes get larger the execution time of Secret Key Generation increases. From Figure 11 and 10 we can see that with the increase of attribute/ policy the size of the message also increases and CP-ABE has the highest in size compared to YCT and KP-ABE. If we perform all encryption at the Gateway (see Figure 13) the execution time increases gradually with the number of attributes where if we do partial encryption at the gateway and offload the rest to the proxy (see Figure 14) the execution time is decreased ten times at the gateway. From the Figure 15 and 16, we can see that the cpu utilization of CP-ABE 20% higher than other ABE schemes at data interval 2 to 0.5 and then gradually increases in the GEM for both cases. Also we can notice that the CPU consumption is lower when we do partial encryption rather than full encryption. In Figure 18 and 17 the memory utilization is almost similar for all the schemes and it is below 40% of the total memory available. From Figure 20 and 19 we can say that the power consumption does not go above 2.7 watt when the data interval is at 0.05 seconds. Further evaluation can be drawn from figure 21 and 22 which show the latency which is required for the whole process. We can see that the CP-ABE has the highest latency among the schemes and YCT has the lowest for both cases. If a use case has a threshold of 10 second latency then:

For full encryption minimum data interval for

- CP-ABE is 1 second
- KP-ABE is 0.7 second
- YCT-ABE is 0.5 second

For partial encryption minimum data interval for

- CP-ABE is 0.7 second
- KP-ABE is 0.7 second
- YCT-ABE is 0.3 second

In conclusion, doing partial encryption at the gateway and offloading the rest to the proxy reduces the resource consumption and mainly latency of the data by 30% than doing full encryption process in the gateway.

6. Conclusions

In this article, an optimizing mechanism for ABE using partial encryption at the gateway and delegating/ offloading the rest of the heavy encryption process to the proxy using smarthome as a case study is presented. The proposed solution shows that further optimization of ABE schemes on resource constrained devices is required. the experiments carried out show that offloading is a must when using ABE schemes on constrained devices. As for future work, we will investigate different techniques and mechanisms to use IoT devices available at smart home to optimize further in terms of latency and ultimately not to be dependant on the proxy server for its computation power.

References

[1] M. Abdelhaq, R. Alsaqour, N. Albrahim, M. Alshehri, M. Alshehri, S. Alserayee, E. Almutairi, and F. Alnajjar, “The impact of selfishness attack on mobile ad hoc network,” *International Journal of Communication Networks and Information Security*, vol. 12, no. 1, pp. 42–46, 2020.

[2] L. Wilbanks, “The impact of personally identifiable information,” *IT Professional*, vol. 9, pp. 62–64, July 2007.

[3] “Iot standards and protocols.” [online] https://www.postscapes.com/internet-of-thingsprotocols/. Accessed: 01-29-2018.

[4] C. J. Barnes, “Smart home alone: The worlds gateway to more efficient use of energy and mayhem,” 2017.

[5] M. Ambrosin, A. Anzanpour, M. Conti, T. Dargahi, S. R. Moosavi, A. Rahmani, and P. Liljeberg, “On the feasibility of attribute-based encryption on internet of things devices,” *CoRR*, vol. abs/1611.08098, 2016.

[6] W. Ali, G. Dustgeer, M. Awais, and M. A. Shah, “Iot based smart home: Security challenges, security requirements and solutions,” in *2017 23rd International Conference on Automation and Computing (ICAC)*, pp. 1–6, Sept 2017.

[7] B. L. R. Stojkoska and K. V. Trivodaliev, “A review of internet of things for smart home: Challenges and solutions,” *Journal of Cleaner Production*, vol. 140, pp. 1454 – 1464, 2017.

[8] C. Wilson, T. Hargreaves, and R. Hauxwell-Baldwin, “Benefits and risks of smart home technologies,” *Energy Policy*, vol. 103, pp. 72 – 83, 2017.

[9] J. Singh, T. Pasquier, J. Bacon, H. Ko, and D. Eyers, “Twenty security considerations for cloud-supported internet of things,” *IEEE Internet of Things Journal*, vol. 3, pp. 269–284, June 2016.

[10] H. Lin and N. W. Bergmann, "Iot privacy and security challenges for smart home environments," *Information*, vol. 7, no. 3, 2016.

[11] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, (New York, NY, USA), pp. 89–98, ACM, 2006.

[12] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertextpolicy attribute-based encryption," in *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, SP '07, (Washington, DC, USA), pp. 321–334, IEEE Computer Society, 2007.

[13] M. Ambrosin, M. Conti, and T. Dargahi, "On the feasibility of attribute-based encryption on smartphone devices," in *Proceedings of the 2015 Workshop on IoT Challenges in Mobile and Industrial Systems*, IoT-Sys '15, (New York, NY, USA), pp. 49–54, ACM, 2015.

[14] X. Yao, Z. Chen, and Y. Tian, "A lightweight attributebased encryption scheme for the internet of things," *Future Generation Computer Systems*, vol. 49, pp. 104 – 112, 2015.

[15] L. Touati, Y. Challal, and A. Bouabdallah, "C-cp-abe: Cooperative ciphertext policy attribute-based encryption for the internet of things," in *2014 International Conference on Advanced Networking Distributed Systems and Applications*, pp. 64–69, June 2014.

[16] R. Chowdhury, H. Ould-Slimane, C. Talhi, and M. Cheriet, *Attribute-Based Encryption for Preserving Smart Home Data Privacy*, pp. 185–197. Cham: Springer International Publishing, 2017.

[17] Y. Jin, C. Tian, H. He, and F. Wang, "A secure and lightweight data access control scheme for mobile cloud computing," in *2015 IEEE Fifth International Conference on Big Data and Cloud Computing*, pp. 172–179, Aug 2015.

[18] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Infocom, 2010 proceedings IEEE*, pp. 1– 9, Ieee, 2010.

[19] W. Liu, J. Liu, Q. Wu, B. Qin, and Y. Zhou, "Practical direct chosen ciphertext secure key-policy attributebased encryption with public ciphertext test," in *European Symposium on Research in Computer Security*, pp. 91–108, Springer, 2014.

[20] S. Roy and M. Chuah, "Secure data retrieval based on ciphertext policy attribute-based encryption (cp-abe) system for the dtns," tech. rep., Citeseer, 2009.

[21] X. Wang, J. Zhang, E. M. Schooler, and M. Ion, "Performance evaluation of attribute-based encryption: Toward data privacy in the iot," in *Communications (ICC), 2014 IEEE International Conference on*, pp. 725–730, IEEE, 2014.

[22] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, "Charm: a framework for rapidly prototyping cryptosystems," *Journal of Cryptographic Engineering*, vol. 3, no. 2, pp. 111– 128, 2013.

[23] "Charm: A tool for rapid cryptographic prototyping." [online] https://github.com/JHUISI/charm. Accessed: 01-29-2017.