Implementing IoT Lottery on Data Encryption Standard

Mohammed M. Alani[1], Muath Alrammal[2], and Munir Naveed[2]
[1] Senior Member of ACM, Abu Dhabi, UAE
[2] Higher Colleges of Technology, Abu Dhabi, 41012, UAE
Email: m@alani.me; malrammal@hct.ac.ae; mnaveed@hct.ac.ae

Abstract—As the number of IoT devices grow rapidly, and soon to exceed 40 billion, security challenges grow rapidly as well. One challenge proven to wreak havoc in the past few years is the use of IoT devices as attacking tools. This paper presents the results of implementing a brute-force attack on Data Encryption Standard using clusters of IoT devices. The implementation presented was successful. Results have shown that a cluster size of 200 IoT devices was able on average to find the key within 350 seconds. Another experiment of a cluster of 2000 IoT devices succeeded in finding the key within 0.015 seconds.

Index Terms—IoT, security, IoT security, DES, data encryption standard, brute-force attack

I. Introduction

With the number of connected Internet-of-Things (IoT) devices jumping from 13.4 billion in 2015 to 38.5 billion in 2020, IoT became a ubiquitous part of our daily lives [1]. This rapid increase in the number and "smartness" of these devices makes them vulnerable to many security issues. In addition to security challenges arising from protecting IoT devices themselves, another challenge is becoming a reality [2]. That challenge is that IoT devices are being used as a tool of attack by malicious attackers. In 2016, a malicious software called Mirai botnet was used to conduct the largest Distributed Denial of Service (DDoS) attack at its time with 1.2 Tbps magnitude. The botnet used an estimated 100,000 vulnerable IoT devices to attack the Managed DNS service structure of Dyn; one of the largest DNS service providers in the world. This attack, that managed to bring down the service for some time, in addition to many other attacks of smaller scale raised many questions about the security of IoT devices, and the Internet as a whole [3].

The most basic method of attack on any block cipher is simply trying all the possible key, *i.e*. brute-force attack. Most other types of attacks try to reduce the search space from all-possible keys to a smaller space for faster cracking.

In 1991, Quisquater and Desmedt published a paper that suggests an attack named the Chinese Lotto attack [4]. The simple idea of the attack was based on the notion of replacing the use of supercomputers to launch an exhaustive key-search attack with use a much larger number of not-super computers [5]. The concept introduced relies on using a massive distributed computing farm to do the key space in a relatively short time.

The Chinese lotto attack suggests that a massive number of television devices with a decryption capability added to them can be used to decrypt an encrypted message. The massively large key search space in this brute-force attack is divided into smaller spaces. Each one of these small search spaces is then assigned to a specific television set. This way, the search space will be exhausted in a much faster way because it will be searched by all the televisions in China in parallel. The television that finds the right key, would signal a message to the television owner that he/she has won the lotto and they should deliver the key code to the authorities to receive the prize.

IoT applications are growing rapidly with time. These devices have been used to solve several kinds of problems in real-world e.g. Inventory control systems [6], Indoor Automation [7], clinical support systems [8]. These applications have been optimized using Artificial Intelligence where techniques have been used to find answers from the unknow worlds e.g. [9], [10].

IoT devices can solve search problems which required extensive computing resources. These devices can run light weight solutions which can solve the problems using the parallel processing strategies. Such solutions have been successful to solve the complex problem e.g. train the system predict demand variation [6], [8], [10]. The strategies are combination of optimal modular design and the fast communication between the modules. The load balancing is added to maximize the use of resources. Such solutions are cross platform and could take benefit of low-level message passing based network communication between processing units.

The use of remote procedure calls (rpc) has been explored in [6] to develop light weight machine learning solutions. The key advantage of such a distributed computing architecture is to delegate sub tasks to each participating IoT device. The device the perform such tasks on anytime basis. To search for an answer for each task, the IoT devices perform a fixed look ahead search and develop a learning structure. The use of such structures is shared via mapping functions which are implemented as remote procedure calls. The use of RPC is also independent of the underlying implementation. The calls can be made from any platform. Such solutions are also intuitive as rpc implementation are available in all operating systems available for IoT devices.

Manuscript received March 25, 2020; revised September 4, 2020.
doi:10.12720/jcm.15.10.735-740

In this paper, we present the first implementation of the conceptual work presented in [2] and validating the suggested design. This paper builds a small-scale IoT brute-force system by combining a number of IoT devices and harnessing their processing power to try possible decryption keys to decrypt a text that is encrypted with Data Encryption Standard (DES).

II. IOT LOTTO

In the design presented in [2], a concept similar to Chinese lottery is presented. However, the design employs a massive number of IoT devices instead of computers. As brute-force attacks rely on exhaustive search of the whole key space to find the correct decryption key, the presented system divides the key search space into smaller spaces that can be searched, in parallel, in a reasonable amount of time by a large group of IoT devices.

To assure proper key distribution, the proposed design employs a central arbitration unit named the Key Distribution Arbiter (KDA). This arbiter is responsible for assigning keys to IoT devices to assure that each device is assigned an exclusive key space. Fig. 1 shows an overview of the proposed design. This arbiter can be hosted on a single server or can be hosted on a cloud-based system. Cloud hosting of the arbiter is highly recommended for implementations that include a large number of IoT devices to assure seamless operation of the system.

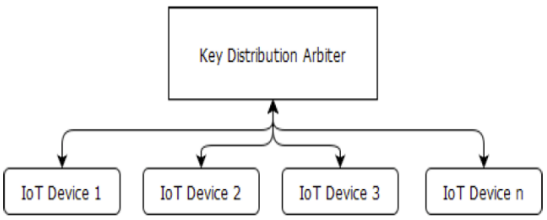

Fig. 1. Overview of IoT Lotto [2]

The IoT Lotto system uses n IoT devices. This arrangement will divide the key search space to n smaller search spaces. The main search space has a size of 2^u where u is the key length of the original symmetric encryption key, measured in bits.

Using n IoT devices would results in having n smaller search spaces each of the size $2^u/n$. This means that the time required to exhaust all of the search space, t, will also be divided by n to be t/n. The higher n goes, the less time it takes to find the correct key.

Inside each IoT device, three software components are required, as shown in Fig. 2. The Key Arbitration Agent (KAA) is the component that coordinates with the KDA to assure that the key search space assigned to the particular device is exclusive. The second component is the decryption algorithm. This component depends on the algorithm that was used in encryption as selected earlier by the attacker. In our setup, we're using DES. The last component is the one that stops the processing when the correct key is found, or all the search space has been exhausted with no success.

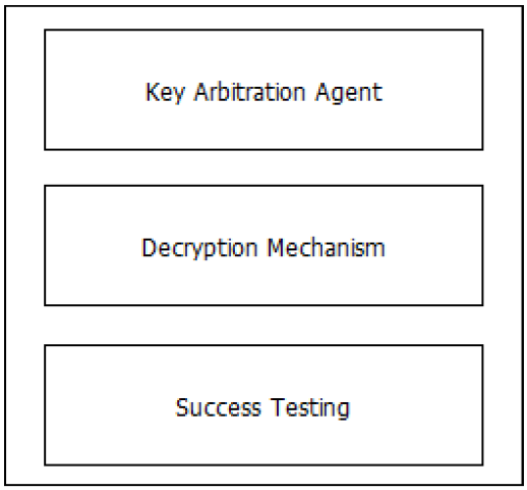

Fig. 2. Software components in IoT devices [2]

A. Key Arbitration Process

As mentioned earlier, each IoT device is assigned an exclusive key search space. As the search process is exhaustive and aims at trying to decrypt the given ciphertext block with all possible key combinations, each device will be assigned a range of keys instead of a random key sequence.

For example, if u = 8, then there are 2^8 possible combinations. Assuming that we have 16 IoT devices, n = 16, each device is expected to try $2^8/16$ = 16 keys. Instead of transferring the keys and storing them in the memory of the IoT devices, each device is given the first key and last key in its search space. Hence, for the first IoT device, the first key is 0000 0000, and the last key is 0000 1111.

For the second IoT device, the first key is 0001 0000 and the last key is 0001 1111. The keys in-between these ranges, can easily be generated by adding 1 until reaching the last key. Fig. 3 shows a flowchart of the key generation process along with the decryption.

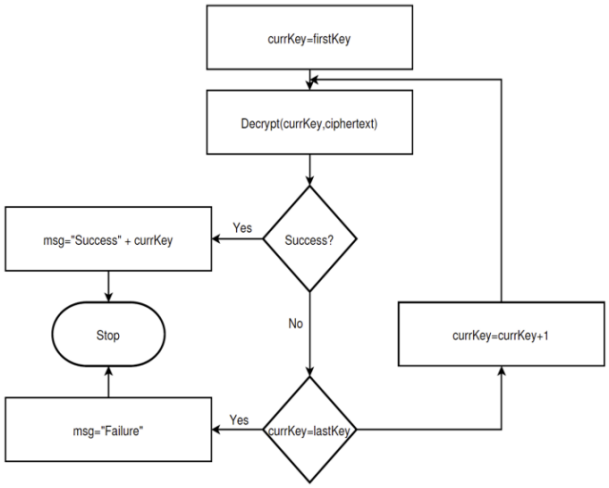

Fig. 3. Flowchart of key generation and decryption at the IoT device [2]

This type of key arbitration reduces the memory requirements at the IoT device side and reduces the amount of data to be transferred between the KDA and the IoT device, keeping in mind that the example explained earlier is a non-realistic one because key sizes can be 128, 256, 512, 1024 bits, or even higher.

III. Implementation Environment

Experiments were performed using Raspberry PI 3 devices with RAM of 1GB and a 64bit 1.2GHz ARM quad core CPU. All the devices are connected via a switch and are assigned standard static IP assigning scheme. Each device runs on Raspbian Operating system and has full support for Remote Procedure Calls (RPC).

The devices are programmed using asynchronous RPC calls to send and receive messages for communication. A device can send a remote call message using Internet Protocol (IP) address, function name and its parameters via RPC calls. Experimental setup is designed as a client-server architecture where a server (KDA) sends an encryption key range to a client (IoT device) to process. The receiving device which works as client searches through key space and finds the correct decryption key.

In an analogy to a master-slave architecture of node communication in a cluster, in our experiment setup, a master acts as server and makes RPC calls to clients which process the requests for decryption key search. However, due to asynchronous communication, the master can stop a slave from searching for a key or give it a new search depending on decisions by load balancing strategy. The load-balancing strategy is focused on maximum utilization of devices in key search process. If a device exhausts its search space earlier than its peers, it is given a new search space created by reducing the load of another device. The key search process is shown in Fig. 3. Each RPC call sends two parameters for search which are current key and cipher text. If it is successful in decrypting the ciphertext then statistical parameters are stored e.g. duration of search, number of iterations etc. A search can be stopped for pre-emptive reason and its status is kept failure due to the fact that the correct key was not found, yet. All RPC calls are made using standard xmlrpc library of Python.

The decryption is performed using python Crypto. Cipher package and applies Data Encryption Standard (DES) algorithm [11]. The standard key size of DES is 56 bits. Hence, the complete size of key search space is 2^{56}. Although DES is not currently used in significant real-life applications, it was chosen due to its limited key size as a proof-of-concept for the IoT lottery system.

The key space is divided into parts using a heuristic that divides the space according to the available number of IoT devices for the search. The search starts from an entry point in the key search space and ends at a fixed length of search process. If search does not find solution within the length, then IoT device returns failure flag.

IV. Results

Experiments are performed using different number of IoT devices. The perform of each strategy is measured using the average time to find solution and number keys to search. As a further measure of optimization, the key was transferred as an integer ignoring the least significant zeros. After the transfer, the key is converted to binary and then the remaining zeros are inserted to the left until it reaches the standard DES key size of 56-bits.

Fig. 4, where the key length is measured in integer digits, shows a profile of search process with 200 nodes in a cluster. The results show that IoT takes more time to find a key successful when the key size is large. In Fig. 5, a profile of time taken by a node in a cluster of 2000 nodes in shown with respect to key length. According to this profile, the time taken per node reduces if key size is larger than 9 integer digits. The reason for this optimization is due to a better load balancing strategy in a large network of IoT device. Fig. 6 supports the results shown in Fig. 5 as it shows the average time take per node in a cluster of large size. The time reduces significantly to find a key successfully if the number of IoT devices grow. The results also show the importance of a load balancing strategy to scale up the performance of IoT network.

The use of optimal load-balancing plays main role in solving the key search with IoT devices. The load-balancing strategies are implemented using the parallel processing strategies used in [9]. It applies a heuristic which depends on estimated time to complete and range of key values. The device that has highest heuristic value is added to a load queue in such a way that the least value is kept the end of the queue. The queue is managed by central system and update the queue after a fixed time interval which is tuned to have optimal updates. Each update includes the refined heuristic values and position in the load queue. As soon as a resource completes it searches, it takes a task from a device of highest heuristic value or the first value of the queue. The queue is updated, and the spared device gets the tasks of least heuristic value.

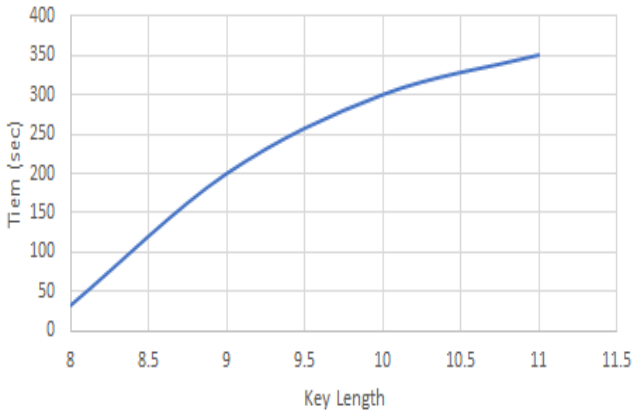

Fig. 4. Time taken by per node to decrypt a key in a cluster of 200 nodes

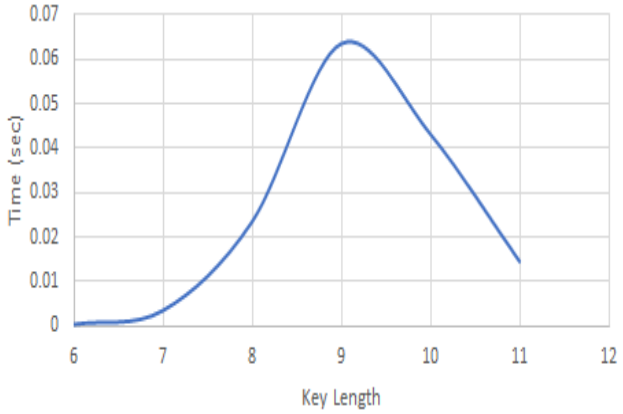

Fig. 5. Time take by an IoT to search a key successfully in a cluster of 2000 nodes

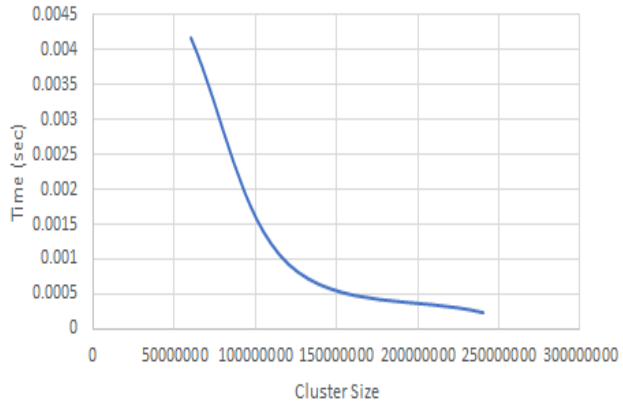

Fig. 6. Average time required to decrypt a key versus size of cluster

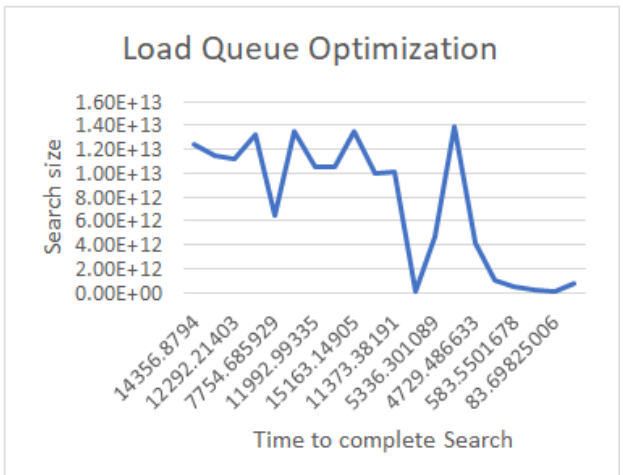

Fig. 7. Time taken by per node to complete search with fixed load queue size

A contextual model is used to transfer the search from one device to another. The latest search state is stored on a contextual model for each member of the queue. Once a device is given a search taken from another, the new device resumes the search from the state stored on the contextual model. The contextual model is also updated according to the fixed time interval. This interval is tuned for both queue update and state space stored for contextual model. The model remains updated for each member of the queue. The contextual model is kept minimal in this way and always have same number of entries as in the load queue.

To find the optimal queue size, the load balancing strategies is explored with different queue sizes and the profile of fixed queue size is shown in Fig. 7. The queue size is kept to a constant number depending on the size of the cluster. A very small size according to the cluster size will degrade the load balancing performance. A dynamic queue size requires a sophisticated heuristic which requires several learning trials. As shown in Fig. 7, the optimal performance of load balancing is achieved with much smaller time window using a fixed size load queue. Without using appropriate load queue size, the parallel processing strategies can consume more time search even less number of states as shown in Fig. 7. If the queue size is too large, the load balancing will consume more time in transfer the contextual information and increases the overload, hence less number of states are explored with large time. With a very small queue size, the time to complete the search is minimized but in such cases each device can only search very limited states.

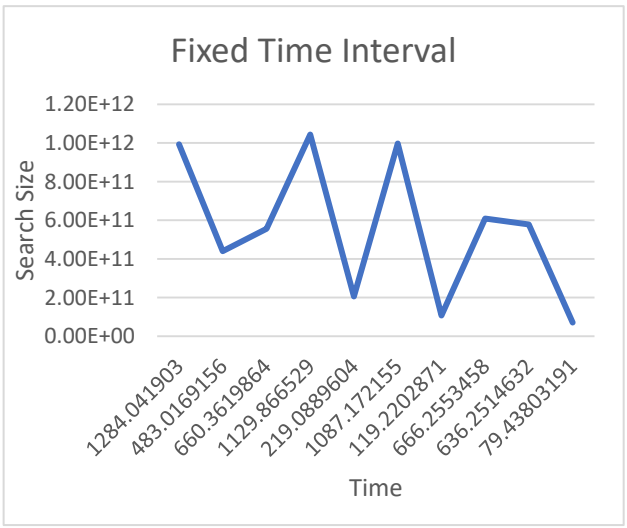

Fig. 8. Time taken by per node to complete search with fixed time interval to update load queue and contextual model

The load balancing strategies highly depends on the time interval to update load queue and contextual model as shown in Figure 8. The smaller time windows add overhead to update heuristic values for each queue member and store state of the latest search progress by each queue member device. Such overheads lead to very limited search of states by each device to find the key. The number of states visited by each device is optimized by using the appropriate window for contextual model and load queue as shown in Fig. 8.

V. Discussion

The implementation of the design system went smoothly and as planned. The results, as expected, show that the increase in cluster size does reduce the time rapidly. However, the relationship does not reflect linearity i.e. the factor by which we increased the number

of IoT devices was not the same factor of decrease in time required to find the correct key, as shown in Fig. 6. This steep decline in the required time with the increase in the cluster size can be seen clearly if we compare the points of 11-digit key length. In Fig. 4, with a cluster of 200 IoT devices, the time required was about 350 seconds. On the other hand, as shown in figure 6, a cluster size of 2000 devices resulted in a time of only 0.015 seconds for an 11-digit key length.

Results presented in the previous section prove the concept introduced in [2]. Results also show that the attack can be implemented at a large or small scale. The results reveal the importance of optimizing load balancing to exploit the use of IoT devices to search the keys for decrypting data. Load balancing strategy is implemented by using a heuristic based model to identify the best candidate for transferring the contextual information from one device to another.

VI. Conclsion and Future Work

In this paper the IoT Lotto attack was implemented successfully on DES-encrypted data. Results have shown that a small cluster of IoT devices was able to break the encryption in around 350 seconds which is very good in comparison to older brute-force attacks on DES. Experiments also shown that the increase in the cluster size would cause a non-linear drop in the time required to break the encryption. In general, the results presented here prove the concept of IoT Lotto introduced in [2].

Future research can be directed towards other more commonly used ciphers like AES or 3DES. In addition, some work can be done in terms of optimization of the process and the code. Another research direction can be implementing the attack on much larger clusters.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Prof. Alani introduced the idea of IoT lotto and supervised the preparation for the experiments and discussion of results.

Dr. Alrammal developed the proposed solution, he conducted the experimentation, and help in finalizing the paper.

Dr. Munir Naveed designed the IoT architecture and analyze the results.

References

[1] Iot connected devices to triple to over 38bn units. https://www.juniperresearch.com/press/press-releases/iot-connected-devices-to-triple-to-38-bn-by-2020

[2] M. M. Alani, "Iot lotto: Utilizing IoT devices in brute-force attacks," in *Proc. 6th International Conference on Information Technology*, New York, NY, USA, 2018.

[3] S. Hilton. (2016). Dyn analysis summary of friday october 21 attack | dyn blog. [Online]. Available: https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/

[4] J. J. Quisquater, Y. G. Desmedt, "Chinese lotto as an exhaustive code-breaking machine," *Computer*, vol. 24, no. 11, 1991.

[5] M. Leech, "Chinese lottery cryptanalysis revisited: The internet as a codebreaking tool," *Tech. Rep.*, 2003.

[6] M. Naveed, M. Adnan, I. Ahmed, and Y. Javed, "Smart IoT based demand variation prediction model," in *Proc. Sixth HCT Information Technology Trends*, Ras Al Khaimah, United Arab Emirates, 2019, pp. 296-299.

[7] M. Naveed, Y. Javed, G. M. Bhatti, and S. Asif, "Smart indoor Positioning Model for Deterministic Environment," in *Proc. Sixth HCT Information Technology Trends (ITT)*, Ras Al Khaimah, United Arab Emirates, 2019, pp. 288-291.

[8] M. Naveed, D. Kitchin, A. Crampton, L. Chrpa, and P. Gregory, "A monte-carlo path planner for dynamic and partially observable environments," in *Proc. IEEE Conference on Computational Intelligence and Games (CIG)*, Granada, 2012, pp. 211-218.

[9] M. Naveed, A. Crampton, D. Kitchin, and L. McCluskey, "Real-Time path planning using a simulation-based markov decision process," in *Research and Development in Intelligent Systems*, Bramer M., Petridis M., Nolle L. eds., 2011.

[10] K. Samara, Y. Javed, and M. Naveed, "Designing common ontologies to support clinical practice guidelines using OWL-based ontologies," in *Proc. Fifth HCT Information Technology Trends (ITT)*, Dubai, United Arab Emirates, 2018, pp. 7-11.

[11] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C. John Wiley & Sons*, 2007.

Prof. Mohammed M. Alani is a Professor of Networking and Cybersecurity. He has authored many books and paper that were published in reputable international venues. His areas of interest include security in cloud computing, and mobile computing, in addition to applications of machine learning and data analytics in cybersecurity

Dr. Muath Alrammal received his MSc in Information Technology from Telecom SudParis, Evry, France, in 2007. He received his Ph.D. degrees in Computer Sciences from the University of Paris Est, Paris, in 2011.

In 2011, he joined LACL, University of Paris Est, as post-doctoral researcher. In 2012 he joined LIFO, University of Orleans, France, as a post-doctoral researcher. Between 2013-2017 he joined IT Department in KIC, Abu Dhabi, UAE, where he was an assistant professor. Since 2017 he works for HCT, Abu Dhabi, UAE as assistant professor in CIS department. His current research interests include processing big data in streaming, performance models, selectivity estimation techniques, and Machine learning. Dr. Alrammal is a member of LaMHA research group.

Dr. Munir Naveed obtained his first degree in Software Engineering and a PhD in Automated planning for real-time-strategy games. He has been exploring AI in different domains e.g. computer games, big data and network security since 2014. The main focus of his research is designing new algorithms to solve problems for real-time applications.