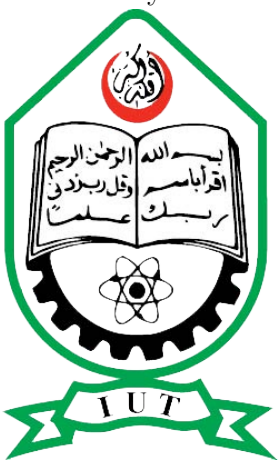
A Framework for Secured Communication Over the Untrusted Cloud

Authors
Tahmid Rashik Chowdhury- 144443
and
Rakib Jahan Siddique- 144445

Supervisor
Dr. Abu Raihan Mostofa Kamal
Professor
Department of CSE
and
Abdullah Al Tariq
Lecturer
Department of CSE
Islamic University of Technology



**A thesis submitted to the Department of CSE
in partial fulfillment of the requirements for the degree of B.Sc. Engineering in CSE
Academic Year: 2017-18**

November 7, 2018

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by Tahmid Rashik Chowdhury and Rakib Jahan Siddiqui under the supervision of Dr. Abu Raihan Mostofa Kamal, Professor, Department of Computer Science and Engineering (CSE), and Abdullah Al Tariq, Lecturer, Department of Computer Science and Engineering (CSE), Islamic University of Technology (IUT), Dhaka, Bangladesh. It is also declared that neither of this thesis nor any part of this thesis has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Authors:

Tahmid Rashik Chowdhury
Student ID- 144443

Rakib Jahan Siddiqui
Student ID- 144445

Supervisors:

Dr. Abu Raihan Mostofa Kamal
Professor
Department of CSE
Islamic University of Technology

Abdullah Al Tarik
Lecturer
Department of CSE
Islamic University of Technology

Acknowledgement

We would like to express our grateful appreciation for Dr. Abu Raihan Mostofa Kamal, Professor, Department of Computer Science and Engineering, IUT and Abdullah Al Tariq, Lecturer, Department of Computer Science and Engineering, IUT for being our advisor and mentor. Their motivation, suggestions and insights for this thesis have been invaluable. Without their support and proper guidance this research would not have been possible. Their valuable opinion, time and input provided throughout the thesis work, from first phase of thesis topics introduction, subject selection, proposing algorithm, modification till the project implementation and finalization which helped us to do our thesis work in proper way. We are really grateful to them.

Abstract

We are now part of such a system where there are many IOT devices. These devices connect to the internet, in other words “the cloud”. Two communicating parties communicate with each other through the cloud. To ensure secured communication between them a secured framework is introduced where heterogeneous IOT devices are used. This heterogeneity plays a major role in security since its very unlikely that all devices will leak information at the same time. Both parties access the sensor data from the sensors. These data vary from each other by a certain margin due to some added noise. Session keys are generated from this sensor data which is used by the end user for communication between itself and the cloud. Amount of data passing between the two communicating parties is a major issue to keep in mind. Data passing between the two communicating parties is significantly reduced to ensure more secured, faster and efficient communication with much less traffic. This reduced data passing is achieved using LCG algorithm. The accuracy with which we are decrypting the sensor data depends on the noise incorporated in the sensor data. So to ensure if both the users belong to the same environment we need to check if successful decryption of message with a slight margin of error is possible or not.

Contents

1 Introduction **9**
1.1 Overview 9
1.2 Problem Statement 16
1.3 Motivation and Scopes 17
1.4 Research Challenges 17
1.5 Thesis Outline 17

2 Literature Review **19**
2.1 A scalable framework for protecting user identity and access pattern in untrusted Web server using forward secrecy, public key encryption and bloom filter 19
2.2 A Framework for Secured Communication over the Untrusted Cloud 20

3 LCG Algorithm **27**
3.1 Pseudo Random Number Generator 27
3.2 Linear congruential generator 28
3.3 Advantages and Disadvantages 30

4 Implementation and Analysis **43**

5 Conclusion **49**

Chapter 1

Introduction

1.1 Overview

The Internet of things (IoT) is the network of physical devices, vehicles, home appliances, and other items embedded with electronics, software, sensors, actuators, and connectivity which enables these things to connect, collect and exchange data.
IoT involves extending Internet connectivity beyond standard devices, such as desktops, laptops, smartphones and tablets, to any range of traditionally dumb or non-internet-enabled physical devices and everyday objects. Embedded with technology, these devices can communicate and interact over the Internet, and they can be remotely monitored and controlled.
IoT security is the area of endeavor concerned with safeguarding connected devices and networks in the Internet of things (IoT).
The Internet of Things involves the increasing prevalence of objects and entities – known, in this context as things – provided with unique identifiers and the ability to automatically transfer data over a network. Much of the increase in IoT communication comes from computing devices and embedded sensor systems used in industrial machine-to-machine (M2M) communication, smart energy grids, home and building automation, vehicle to vehicle communication and wearable computing devices.
IoT also have some advantages. Over time, IoT evolved from devices being able to collect and communicate data to devices being able to process data and make decisions.
IoT wants to connect all potential objects to interact each other on the internet to provide secure, comfort life for human. Internet of Things (IoT) makes our world as possible as connected together. Embedded computing devices would be exposed to internet influence.
IoT is a network of all physical, tangible items such as devices, cars, buildings and anything that has software or sensors in them which allow us to automate everything.

IoT encourages the communication between devices, also famously known as Machine-to-Machine (M2M) communication. Because of this, the physical devices are able to stay connected and hence the total transparency is available with lesser inefficiencies and greater quality.
Due to physical objects getting connected and controlled digitally and centrally with wireless infrastructure, there is a large amount of automation and control in the workings. Without human intervention, the machines are able to communicate with each other leading to faster and timely output. Another obvious advantage of IoT is monitoring. Knowing the exact quantity of supplies or the air quality in your home, can further provide more information that could not have previously been collected easily.
IoT fundamentally proves to be very helpful to people in their daily routines by making the appliances communicate to each other in an effective manner thereby saving and conserving energy and cost. Vulnerabilities of IOT devices:

1. Insecure Web Interface- The first point concerns security related issues with the web interfaces built into IoT devices that allows a user to interact with the device, but at the same time could allow an attacker to gain unauthorised access to the device.

2. Insufficient Authentication/Authorisation- This area deals with ineffective mechanisms being in place to authenticate to the IoT user interface and/or poor authorisation mechanisms whereby a user can gain higher levels of access then allowed.

3. Insecure Network Services-This point relates to vulnerabilities in the network services that are used to access the IoT device that might allow an intruder to gain unauthorised access to the device or associated data.

4. Lack of Transport Encryption- This deals with data being exchanged with the IoT device in an unencrypted format. This could easily lead to an intruder sniffing the data and either capturing this data for later use or compromising the device itself.

5. Privacy Concerns- Privacy concerns are generated by the collection of personal data in addition to the lack of proper protection of that data. Privacy concerns are easy to discover by simply reviewing the data that is being collected as the user sets up and activates the device. Automated tools can also look for specific patterns of data that may indicate collection of personal data or other sensitive data.

6. Insecure Cloud Interface- This point concerns security issues related to the cloud interface used to interact with the IoT device. Typically this would imply poor authentication controls or data traveling in an unencrypted format allowing an attacker access to the device or the underlying data.

7. Insecure Mobile Interface- Similar to the point above, weak authentication or unencrypted data channels can allow an attacker access to the device or

underlying data of an IoT device that uses a vulnerable mobile interface for user interaction.

8. Insufficient Security Configurability- Insufficient security configurability is present when users of the device have limited or no ability to alter its security controls. Insufficient security configurability is apparent when the web interface of the device has no options for creating granular user permissions or for example, forcing the use of strong passwords. The risk with this is that the IoT device could be easier to attack allowing unauthorised access to the device or the data.

9. Insecure Software/Firmware: The lack of ability for a device to be updated presents a security weakness on its own. Devices should have the ability to be updated when vulnerabilities are discovered and software/firmware updates can be insecure when the updated files themselves and the network connection they are delivered on are not protected. Software/Firmware can also be insecure if they contain hardcoded sensitive data such as credentials. The inability of software/firmware being updated means that the devices remain vulnerable indefinitely to the security issue that the update is meant to address. Further, if the devices have hardcoded sensitive credentials, if these credentials get exposed, then they remain so for an indefinite period of time.

10. Poor Physical Security: Physical security weaknesses are present when an attacker can disassemble a device to easily access the storage medium and any data stored on that medium. Weaknesses are also present when USB ports or other external ports can be used to access the device using features intended for configuration or maintenance. This could lead to easy unauthorised access to the device or the data.

Many types of attacks have been around for a very long time. What's new is the scale and relative simplicity of attacks in the Internet of Things (IoT) – the millions of devices that are a potential victim to traditional style cyber attacks. 5 most common cyber attacks on IOT devices are:

1. Botnets: A botnet is a network of systems combined together with the purpose of remotely taking control and distributing malware. Controlled by botnet operators via Command-and-Control-Servers (C&C Server), they are used by criminals on a grand scale scale for many things: stealing private information, exploiting online-banking data, DDos-attacks or for spam and phishing emails.
   With the rise of the IoT, many objects and devices are in danger of, or are already being part of, so called thingbots – a botnet that incorporates independent connected objects. Botnets as well as thingbots consist of many different devices, all connected to each other – from computers, laptops, smartphones and tablets to now also those "smart" devices. These things have two main characteristics in common: they are internet enabled

and they are able to transfer data automatically via a network. Anti-spam technology can spot pretty reliably if one machine sends thousands of similar emails, but it's a lot harder to spot if those emails are being sent from various devices that are part of a botnet. They all have one goal: sending thousands of email requests to a target in hopes that the platform crashes while struggling to cope with the enormous amount of requests.

2. Man-In-The-Middle Concept: The man-in-the-middle concept is where an attacker or hacker is looking to interrupt and breach communications between two separate systems. It can be a dangerous attack because it is one where the attacker secretly intercepts and transmits messages between two parties when they are under the belief that they are communicating directly with each other. As the attacker has the original communication, they can trick the recipient into thinking they are still getting a legitimate message. Many cases have already been reported within this threat area, cases of hacked vehicles and hacked "smart refrigerators".

3. Data & Identity Theft: While the news is full of scary and unpredictable hackers accessing data and money with all types of impressive hacks, we are often also our own biggest security enemy. Careless safekeeping of internet connected devices (e.g. mobile phone, iPad, Kindle, smartwatch, etc.) are playing into the hands of malicious thieves and opportunistic finders. The main strategy of identity theft is to amass data – and with a little bit of patience, there is a lot to find. General data available on the internet, combined with social media information, plus data from smart watches, fitness trackers and, if available, smart meters, smart fridges and many more give a great all-round idea of your personal identity. The more details can be found about a user, the easier and the more sophisticated a targeted attack aimed at identity theft can be.

4. Social Engineering: Social engineering is the act of manipulating people so they give up confidential information. The types of information that criminals are seeking can vary, but when individuals are targeted, the criminals are usually trying to deceive the user into giving them passwords or bank information. Or they could be trying to access a computer in order to secretly install malicious software that will then give them access to personal information, as well as giving them control over the computer. Typically, social engineering hacks are done in the form of phishing emails, which seek to have you divulge your information, or redirects to websites like banking or shopping sites that look legitimate, enticing you to enter your details.

5. Denial of Service: A denial of service (DoS) attack happens when a service that would usually work is unavailable. There can be many reasons for unavailability, but it usually refers to infrastructure that cannot cope due to capacity overload. In a Distributed Denial of Service (DDoS) attack, a large number of systems maliciously attack one target. This is

often done through a botnet, where many devices are programmed (often unbeknownst to the owner) to request a service at the same time. In comparison to hacking attacks like phishing or brute-force attacks, DoS doesn't usually try to steal information or leads to security loss, but the loss of reputation for the affected company can still cost a lot of time and money. Often customers also decide to switch to a competitor, as they fear security issues or simply can't afford to have an unavailable service. Often a DoS attack lends itself to activists and blackmailers.

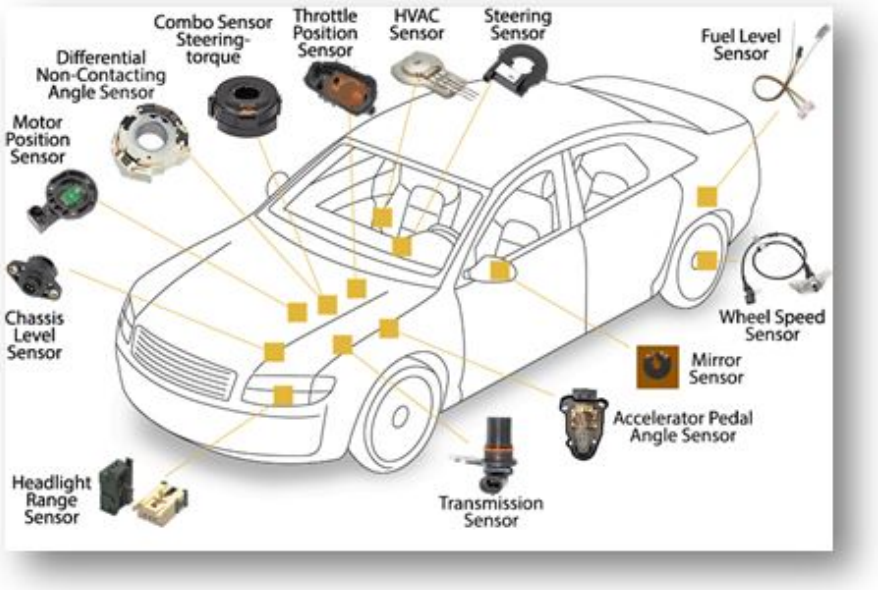
To prevent these attacks we need security as we cant trust any user. Reasons why we can't implement security features of phone or desktop computer, like windows firewall is that we need lightweight yet strong security features.

Everyone, from consumers to corporates, is embracing the changes brought by the revolution called the Internet of Things (IoT). It has changed the world in more ways than we could imagine until a few years back. And the changes and advancements will continue in future as well, in fact, Internet of Things (IoT) will shape our future. Already the numbers are staggering; billions of sensors connected with billions of devices are redefining almost everything under the sun. It is estimated that around 75 Billion devices will be interconnected by 2025. Around $6 Trillion is estimated to be spent on the Internet of Things (IoT) solutions in next five years. No price for guessing that the reason this amount is expected to be spent on IoT is that IoT has already shown a lot of potential in a very short time and it has just begun. There is not a single industry that is not impacted by IoT by now. Some have already had a major influence of IoT while some are just beginning to realise its importance. Retail companies are investing heavily in IoT as they understand the importance of data-driven analytics and also to further improve customer experience. Customers, on the other hand, are enjoying the new experiences made possible because of IoT. Data-driven analytics based on the data gathered from billions of sensors to reach the potential customers and for better marketing will be very common in 2018. Although in very early stages, Amazon's drone delivery system is completely powered by IoT and is expected to be a ground-breaking innovation. IoT is reshaping healthcare as well. Wearable technology to monitor your condition at any time and anywhere is very common now. Sensors are collecting data and at the same time, the data can be visible to doctors. This is helping doctors closely monitor crucial patients from far away.
The manufacturing industry is making use of smart machines to improve the overall manufacturing process and to produce better goods. IoT has something to offer to everyone, it has completely changed the way we used to do business, socialize and have fun. One of the main reasons that make IOT devices more safe and secured is its heterogeneous characteristics.
What do we mean by heterogeneity?
By heterogeneity of IOT devices we mean the diversity of IOT devices in terms of the protocols and standards of the IOT devices. The quality or state of being diverse in character or content is called heterogeneity. IoT devices are hetero-

geneous in nature.
Today, however, service developers looking to add value on top of existing IoT systems are faced with very heterogeneous devices and systems. These systems implement a wide variety of network connectivity options, protocols (proprietary or standards-based), and communication methods all of which are unknown to a service developer that is new to the IoT. Even within one IoT standard, a device typically has multiple options for communicating with others. Heterogeneity plays a major role in ensuring a strong, sustainable and robust security. Different IOT devices are considered as a separate sensor. Each IOT device are heterogeneous meaning that each device communicate using a separate and different protocol and standard. So even if any intruder manage to hack any one of the sensors he wont be able to access the data of other sensors as all the sensors don't use the same protocol for communication. Thus security is ensured in this case.

Lets go through an example to explain how heterogeneity ensures such security.
Assume an example of a car. A car has many sensors attached to it like weight sensor, pressure sensor, temperature sensor, accelerometer sensor, brake sensor, fingerprint sensor etc. Now lets assume if the intruder manage to access/hack all 4 of the weight, pressure, temperature and fingerprint sensor only then he will be able to start the car. And its very unlikely that he will be able to hack all 4 sensors at the same time as all 4 of the sensors use different protocols. And unless the data of all 4 of the sensors matches the data of the actual sensor data of the verified user, the car wont start. And thus no unauthorized user can gain access to the car and sabotage its other sensors like accelerometer and brake sensors. In this way we can prevent the user in the car from getting harmed in any way by the intruder.
If any intruder fails to access even a single IOT device(sensor) then he wont be as an authorized user and wont be given access to the object to which the sensors are connected.

Let's see another example:



Here is an example of a smart car. It has the above mentioned sensors. Suppose Steering sensor is compromised. We can imagine the damage the attacker can do using just the steering sensor. But as all the sensors are diverse in nature we can build up one security instead of a single security for each sensor. If we can do that a single security feature can be implemented using all of the sensors which means if an attacker can hack one sensor it won't be of use. Instead he has a simultaneously hack all the sensors to bypass the security which gives us an huge edge over the attacker.

1.2 Problem Statement

In this framework there are some drawbacks. One of them is that huge amount of data is being passed between the two parties which requires longer time for data transferring and creates congestion in the network. For example the index matrix that is passed from Party A to Party B is very large in size creates huge traffic in the network.
Another drawback in this framework is that due to this large number of data passing it becomes easier for the intruder to figure out about the session keys of both the parties. The less information we pass between the two parties the less information the intruder has and less chances for the intruder to figure out the session keys.

1.3 Motivation and Scopes

We intend to improve the efficiency of the protocol. Huge data passing between the two communicating parties is a major issue. So reducing the data passing even further is also one of our future motives. Introducing a more secured protocol for encryption and decryption is also something that we want to achieve in the future. Making sure our protocol works smoothly even on a low computational power device can also be achieved later.

1.4 Research Challenges

- Usually high computation powered device needed for this protocol to work smoothly.
- Huge energy is consumed during this secured data passing process.
- It was quite tough to reduce the data passing from a large matrix to a single number

1.5 Thesis Outline

In chapter 1 we had a brief overview of our study in a concise manner. Chapter 2 contains the literature review related to our paper. Chapter 3 consists of our proposed protocol where we have elaborately discussed about the LCG encryption algorithm. In Chapter 4 we discussed about our implementation and analysis of the LCG algorithm. And in Chapter 5 we mentioned our conclusion.

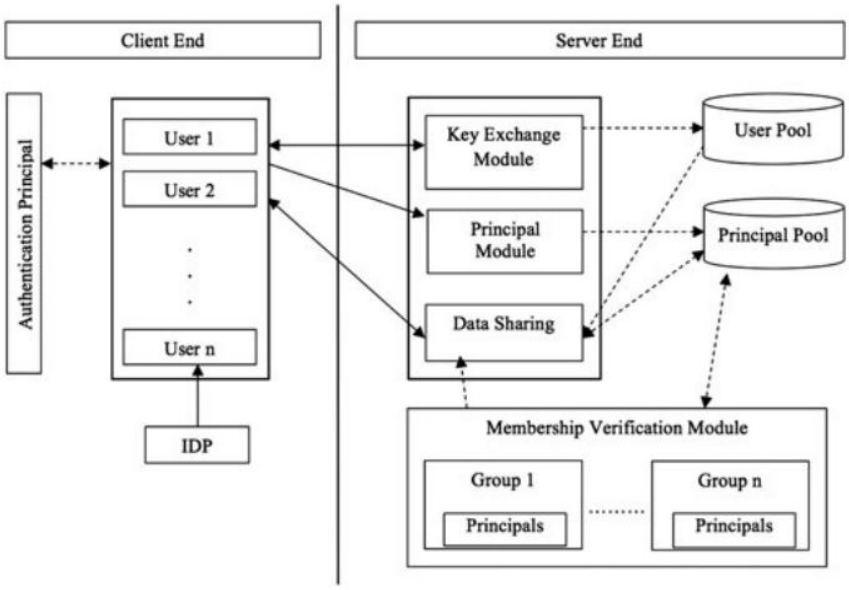
Chapter 2

Literature Review

We have gone through some research works related to Secured communication over the cloud. Some of these protocols were mainly focused on hiding user identity and protecting user data and privacy, while some were more focused on robust security. But almost none of the papers focused on reducing data passing. So we intend to overcome and solve this issue in this paper.

2.1 A scalable framework for protecting user identity and access pattern in untrusted Web server using forward secrecy, public key encryption and bloom filter

This paper consists of a scalable distributed framework which mainly focuses on securing user identity and hides user identity using forward secrecy[1] which uses Deffie Hellman key exchange mechanism[4]. Data sharing strategy is also hidden using "Principals". Bloom Filter[5] was introduced in this paper. Here User access pattern is abstracted using bloom filter and scalability is improved using extended Bloom filter.
However there is huge potential risks from attacks like Selective DoS attack[2][3]. User identity is very much unsafe in case of cloud data breach. This paper is usually focused on Client server architecture.



There is also not much implementation for distributed pervasive networks and IoT.

2.2 A Framework for Secured Communication over the Untrusted Cloud

This framework provides more secured communication for data exchange between two users by using matrices like session key seed and index matrices. Using these matrices Final session key matrix was formed which was used for encryption of data. Utilizes interaction among heterogeneous IoT devices to improve the security. Checks if both parties have similar session keys with a slight margin of error.
Initially both parties take sensor data from the sensors. But since both parties(Party A and Party B) are in different environment their environment data matrix are not exactly same. It varies a little.
Environment matrix consists of different features(temperature, pressure, humidity etc) on the column and observations on the row. So to reduce that difference a bit moving average of the observations of each feature was used by both parties from which both parties got intermediate key matrices.
Once both parties got the intermediate matrices now Party B selects some unique random elements and using those random elements Party B creates a matrix called Initial session key matrix by placing those unique elements randomly in different positions of the matrix. Now party B forms the Session key seed matrix by element wise multiplication of the Initial session key matrix and

the Intermediate key matrix. Now this session key seed matrix, the unique elements selected by Party B and W_{otp} (window size for calculating moving average) is sent over the cloud to Party A and this session key seed is also received by any intruder. But using this session key seed matrix the intruder won't be able to interpret anything. Once the Party A gets the session key seed matrix it does element wise division of the session key seed matrix by its intermediate key matrix to get its own initial session key matrix which varies slightly from the initial session key matrix of Party B due to different environmental data of both parties. Once both parties get the initial session key now Party A selects some random indices from the initial session key matrix and using those index numbers it creates a huge matrix called session key Index matrix. Using the indices mentioned in the session key index matrix Party A selects the elements of those indices from the initial session key matrix and creates a new matrix called the Final session key matrix. This session key index matrix is sent to Party B where this index matrix is used by Party B to access the elements of the mentioned indices from its initial session key to form its own Final session key matrix. The public key obtained by both the parties from the public database are same. Using that public key and the final session key Party A obtains its Public session key matrix. Party B also uses its public key and its final session key to obtain its Public session key matrix. As we have seen in [6], Short Integer Solution can be used to generate One-way Hash Functions, in our work we applied the same process to generate the hash values of the Primary Public Key K_{pubA} of PartyA. Once both the parties have obtained their session public keys now comes the next step. Party A multiplies the transpose of its session public matrix with its private key matrix and adds an error matrix which is obtained by sampling from a fixed Gaussian. This new matrix is actually a column vector of M_{pk} dimensions. This column vector is named k_A.

$$\mathbb{k}_A = K_{SP_A}^T \times \mathbb{k}_{prvA} + \epsilon_A$$

$$or,$$

$$\begin{bmatrix} k_{A_1} \\ k_{A_2} \\ . \\ . \\ . \\ k_{Am_{pk}} \end{bmatrix} = K_{SP_A}^T \times \begin{bmatrix} k_{p_1} \\ k_{p_2} \\ . \\ . \\ . \\ k_{pn_{env}} \end{bmatrix} + \begin{bmatrix} \epsilon_{A_1} \\ \epsilon_{A_2} \\ . \\ . \\ . \\ \epsilon_{Am_{pk}} \end{bmatrix}$$

This column vector is now sent to Party B. Party B also generates an error vector, E_b. After generating the error vector Party B calculates two row vectors u_1 and u_2 using the following equations:

$$\mathbb{u}_1 = \epsilon_B^T \times K_{SP_B}^T$$

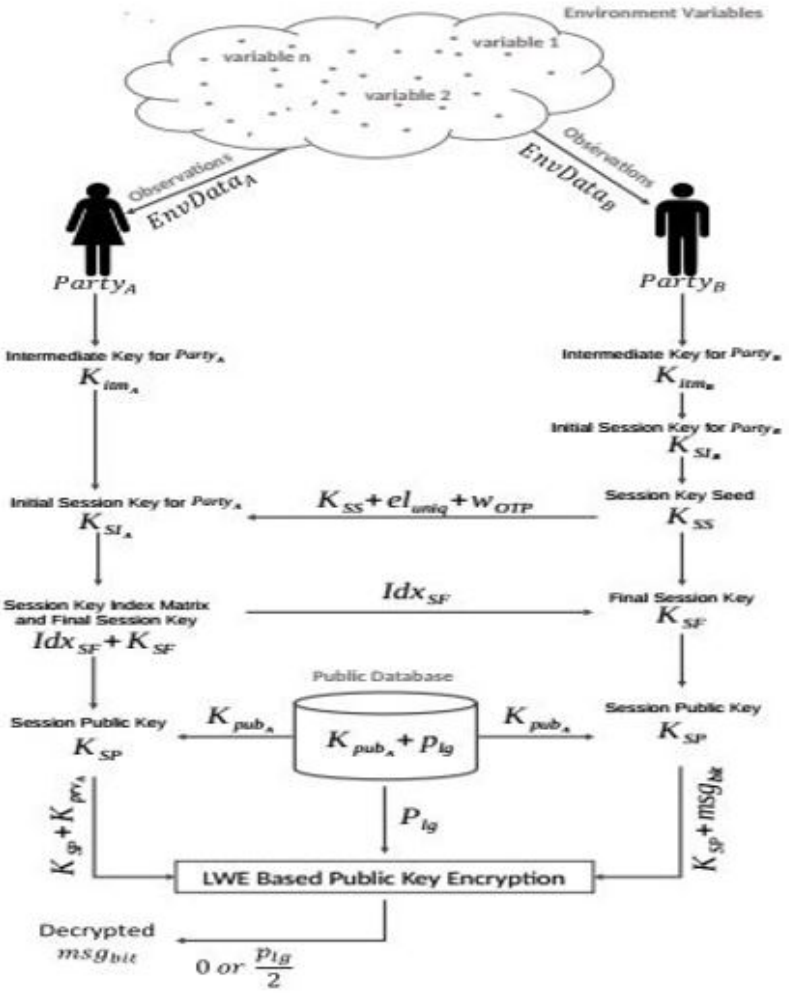
$$u_2 = \epsilon_B^T \times \mathbb{k}_A + msg_{bit} * \frac{p_{lg}}{2}$$

u_1 is obtained by multiplying the generated error vector E_b with the transpose of session public key of Party B. And u_2 is obtained by multiplying the transpose of error vector of Party B with the column vector k_A obtained from party A and adding the message bit multiplied by $P_{lg}/2$. In simpler words, Party B is encrypting the message bit and then sending this encrypted message to Party A. Party A upon receiving the two row vectors u_1 and u_2 calculates a row vector u using the following equation:

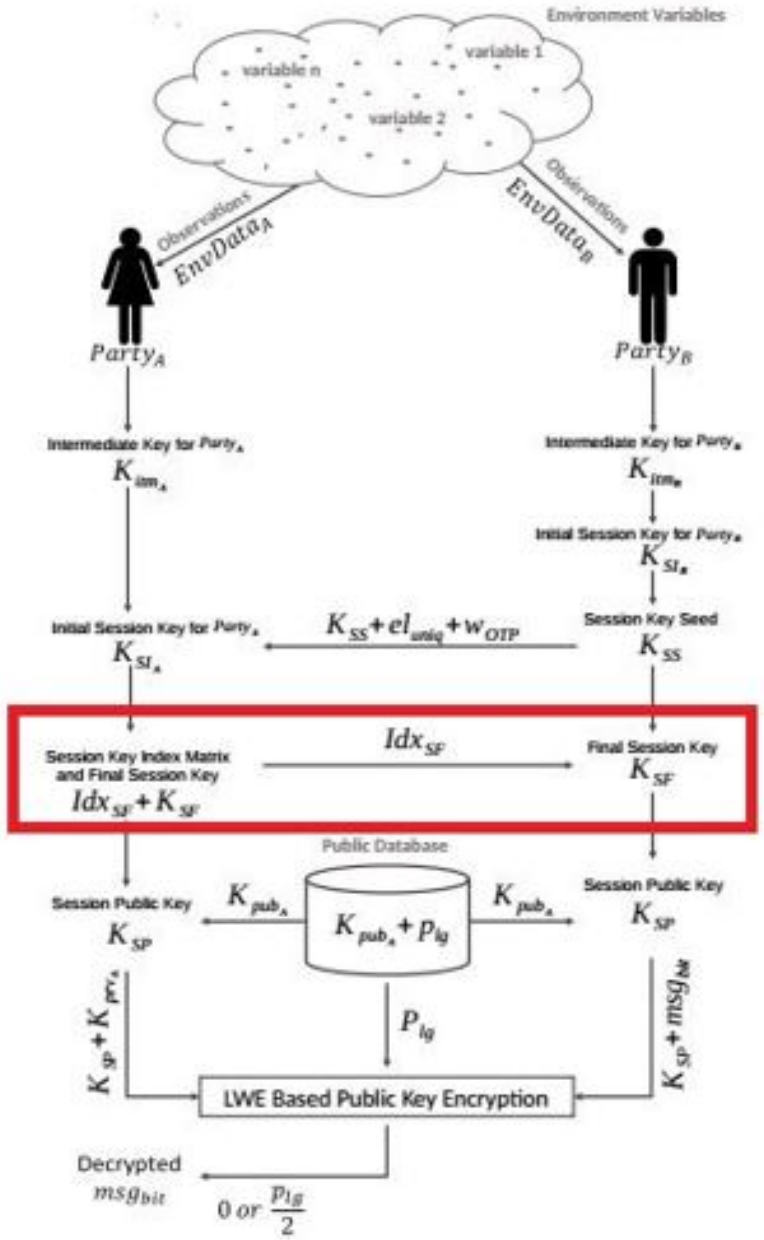
$$u = (u_2 - \mathbb{u}_1 \times \mathbb{k}_{prv_A}) \ mod \ p_{lg}$$

The row vector u is obtained by multiplying the row vector u_1 with the private key of A and subtracting it from u_2. And finally doing modulus of that result by P_{lg}. The value of u will be close to $P_{lg}/2$ if message bit sent to Party A is 1 and the value of u will be very far from $P_{lg}/2$ if the message bit sent to Party B is 0.

Below we have shown the framework for secured communication between the two parties.



However this architecture also has some flaws. Attacker might launch a D-DoS attack if large number of similar devices are connected on the internet. So to prevent this from happening heterogeneous IoT devices are used. But this heterogeneity of IoT devices prevents standardization of security protocols which is another drawback of this architecture. And the most vital drawback of this protocol is that huge amount of data is being passed between the two parties which requires longer time for data transmission and creates congestion in the network. For example the index matrix that is passed from Party A to Party B is very large in size creates huge traffic in the network.
However in our proposed framework we made some changes in this marked region.



Instead of passing this huge index matrix we will be passing only a single number which reduces data transmission to a large extent.

Chapter 3

LCG Algorithm

In the framework we just discussed about Party A creates a Final session key while passing session key index matrix. This matrix is randomly created from the initial session key of Party A. After receiving this session key index matrix from party A, party B also creates a final session key by taking the same indices from it's initial session key.
We propose that instead of passing this huge Session key Index Matrix of Party A to Party B, Party A sends a seed value which is a single integer to Party B and both generates a session key index matrix.
This generation is done by getting pseudo random numbers using the Linear Congruential Generator(LCG) algorithm which starts with a random seed which A passes to B initially by generating it randomly.

3.1 Pseudo Random Number Generator

A pseudo-random number generator (PRNG) is a program written for, and used in, probability and statistics applications when large quantities of random digits are needed. Most of these programs produce endless strings of single-digit numbers, usually in base 10, known as the decimal system. When large samples of pseudo-random numbers are taken, each of the 10 digits in the set 0,1,2,3,4,5,6,7,8,9 occurs with equal frequency, even though they are not evenly distributed in the sequence. [7]
Many algorithms have been developed in an attempt to produce truly random sequences of numbers, endless strings of digits in which it is theoretically impossible to predict the next digit in the sequence based on the digits up to a given point. But the very existence of the algorithm, no matter how sophisticated, means that the next digit can be predicted! This has given rise to the term pseudo-random for such machine-generated strings of digits. They are equivalent to random-number sequences for most applications, but they are not truly random according to the rigorous definition. The digits in the decimal

expansions of irrational numbers such as pi (the ratio of a circle's circumference to its diameter in a Euclidean plane), e (the natural- logarithm base), or the square roots of numbers that are not perfect squares (such as 2 1/2 or 10 1/2) are believed by some mathematicians to be truly random. But computers can be programmed to expand such numbers to thousands, millions, billions, or trillions of decimal places; sequences can be selected that begin with digits far to the right of the decimal (radix) point, or that use every second, third, fourth, or n th digit. However, again, the existence of an algorithm to determine the digits in such numbers is used by some theoreticians to argue that even these single-digit number sequences are pseudo-random, and not truly random. The question then becomes, Is the algorithm accurate (that is, random) to infinity, or not? – and because no one can answer such a question definitively because it is impossible to travel to infinity and find out, the matter becomes philosophical.

3.2 Linear congruential generator

Randomness has always been a crucial concept in computing. Pseudorandom number generators are designed to produce numbers that look random according to statistical tests. Good PRNGs generate numbers that are essentially "random" for any practical purpose. However, the degree to which the generated numbers approximate true randomness depends on the PRNG algorithm– resultant sequences will never be truly random since they are generated using relatively small sets of initial values.

The simplest and most common kind of PRNG is the linear congruential generator (LCG), perhaps best explained through a manual example: The linear congruential method produces a sequence of integers X_1, X_2, X_3,... between zero and m-1 according to the following recursive relationship [8]:

$$X_{i+1} = (aX_i + c) \bmod m, \qquad i = 0, 1, 2, ...$$

- The initial value X_0 is called the seed
- a is called the constant multiplier
- c is the increment
- m is the modulus
- The selection of a, c, m and seed (X_0) drastically affects the statistical properties such as mean and variance, and the cycle length
- When c is not equal to 0, the form is called the mixed congruential method; When c = 0, the form is known as the multiplicative congruential method

Issues to consider:

- The numbers generated from the example can only assume values from the set I = 0, 1/m, 2/m, ..., (m-1)/m. If m is very large, it is of less problem. Values of $m = 2^{31} - 1$ and $m = 2^{48}$ are in common use.

- To achieve maximum density for a given range, proper choice of a, c, m and is very important. Maximal period can be achieved by some proven selection of these values.

  - For m a power of 2, i.e. $m = 2^b$, and c $\neq$ 0 , the longest possible period is $P = m = 2^b$, when c is relatively prime to m and a = 1 + 4 k where k is an integer.
  - For m a power of 2, i.e. $m = 2^b$, and c=0 , the longest possible period is $P = m/4 = 2^{b-2}$, when X_0 is odd and the multiplier, a is given by a=3+8k or a=5+8k where k is an integer.
  - For m a prime number and c = 0, the longest possible period is P = m - 1 when a satisfies the property that the smallest k such that a^{k-1} is divisible by m is k = m - 1.

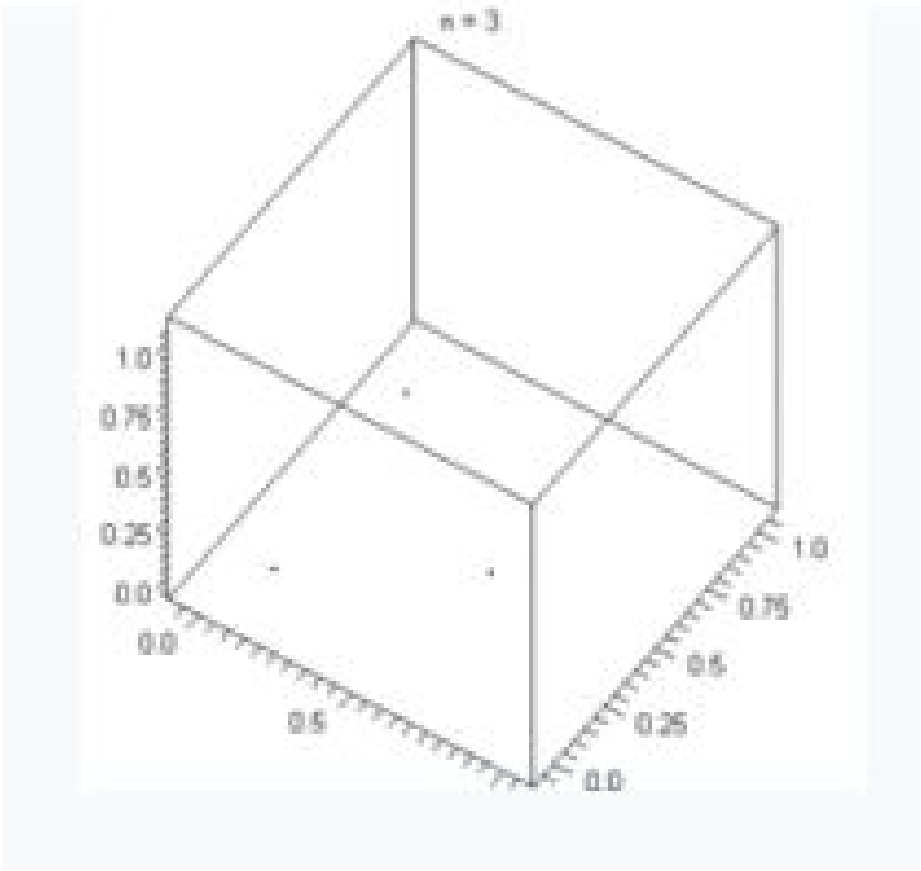
For example, we choose m = 7 and a = 3, the above conditions satisfy. Here k has to be 6.

- when k = 6, $a^{k-1} = 728$ which is divisible by m
- when k = 5, $a^{k-1} = 242$ which is not divisible by m
- when k = 4, $a^{k-1} = 80$ which is not divisible by m
- when k = 3, $a^{k-1} = 26$ which is not divisible by m

Of course, the longest possible period here is 6, which is of no practical use. But the example shows how the conditions can be checked.

3.3 Advantages and Disadvantages

LCGs are fast and require minimal memory (one modulo-m number, often 32 or 64 bits) to retain state. This makes them valuable for simulating multiple independent streams.



Hyperplanes of a linear congruential generator in three dimensions. This structure is what the spectral test measures.

Although LCGs have a few specific weaknesses, many of their flaws come from having too small a state. The fact that people have been lulled for so many years into using them with such small moduli can be seen as a testament to strength of the technique. A LCG with large enough state can pass even stringent statistical tests; a modulo-2 LCG which returns the high 32 bits passes TestU01's SmallCrush suite and a 96–bit LCG passes the most stringent BigCrush suite.

For a specific example, an ideal random number generator with 32 bits of output is expected (by the Birthday theorem) to begin duplicating earlier outputs after $\sqrt{m} = 2^{16}$ results. Any PRNG whose output is its full, untruncated state will not produce duplicates until its full period elapses, an easily detectable

statistical flaw. For related reasons, any PRNG should have a period longer than the square of the number of outputs required. Given modern computer speeds, this means a period of 2^{64} for all but the least demanding applications, and longer for demanding simulations.

One flaw specific to LCGs is that, if used to choose points in an n–dimensional space, the points will lie on, at most, $(n!.m)^{1/n}$ hyperplanes (Marsaglia's Theorem, developed by George Marsaglia).[24] This is due to serial correlation between successive values of the sequence X_n. Carelessly chosen multipliers will usually have far fewer, widely spaced planes, which can lead to problems. The spectral test, which is a simple test of an LCG's quality, measures this spacing and allows a good multiplier to be chosen.
The plane spacing depends both on the modulus and the multiplier, a large enough modulus can reduce this distance below the resolution of double precision numbers. The choice of the multiplier becomes less important when the modulus is large. It is still necessary to calculate the spectral index and make sure that the multiplier is not a bad one, but purely probabilistically it becomes extremely unlikely to encounter a bad multiplier when the modulus is larger than about 2^{64}.

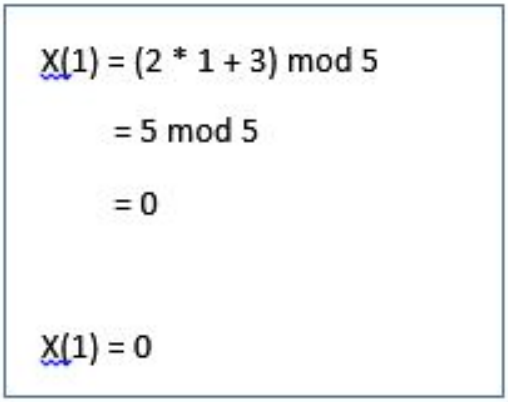
Another flaw specific to LCGs is the short period of the low–order bits when m is chosen to be a power of 2. This can be mitigated by using a modulus larger than the required output, and using the most significant bits of the state.

LCGs should be chosen very carefully for applications where high–quality randomness is critical. Any Monte–Carlo simulation should use an LCG with a modulus greater and preferably much greater than the cube root of the number of random samples which are required. This means, for example, that a (good) 32–bit LCG can be used to obtain about a thousand random numbers; a 64–bit LCG is good for about 2^{21} random samples which is a little over two million, etc. For this reason, LCGs are in practice not suitable for large scale Monte–Carlo simulations.

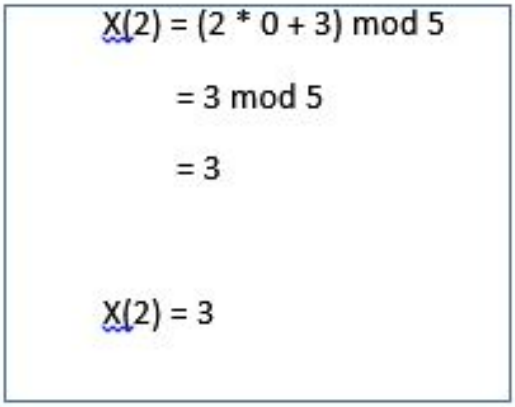
LCGs are not intended, and must not be used, for cryptographic applications; use a cryptographically secure pseudorandom number generator for such applications. Nevertheless, for some applications LCGs may be a good option. For instance, in an embedded system, the amount of memory available is often severely limited. Similarly, in an environment such as a video game console taking a small number of high–order bits of an LCG may well suffice. The low–order bits of LCGs when m is a power of 2 should never be relied on for any degree of randomness whatsoever. Indeed, simply substituting 2^n for the modulus term reveals that the low order bits go through very short cycles. In particular, any full–cycle LCG when m is a power of 2 will produce alternately odd and even results.

Example:

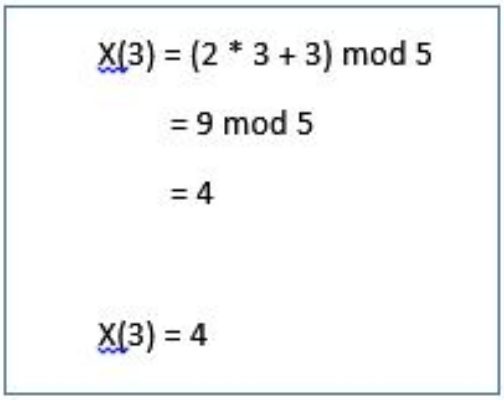
Let's see a few examples of how LCG will look with different constant values.
Let's consider the values,
Seed, X(0) = 1
a = 2
c = 3
m = 5
Now using the formula,
X(i+1) = (a * X(i) + c) mod m
We get,

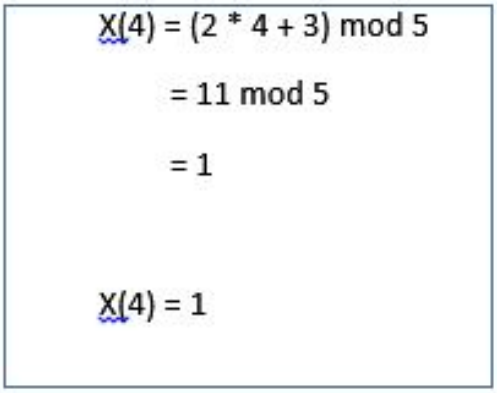


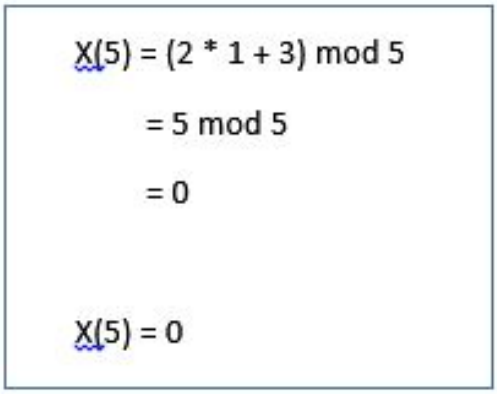
Now let's find X(2),



Continuing,

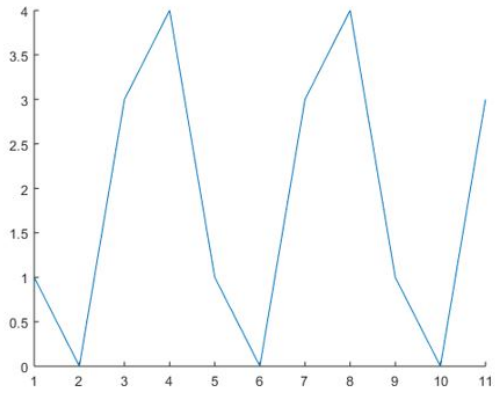






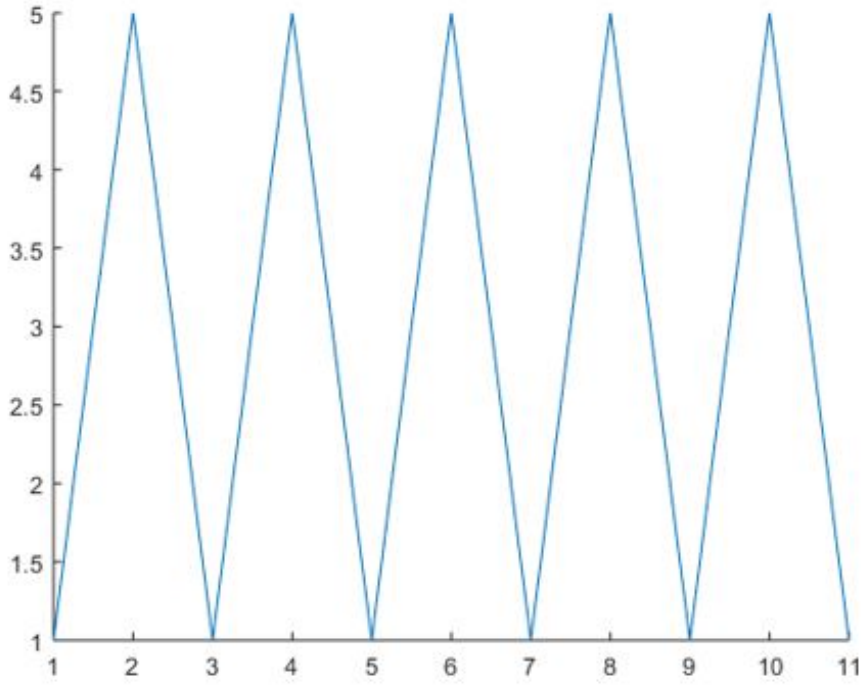
From here onwards the pattern, 0, 3, 4, 1 keeps repeating itself.
Now let's ask ourselves a simple question. Why don't we see values greater than 5? Yes, because modulus is 5. So the range of the random numbers depend on

the modulus.
Again, we are getting numbers from 0 to 5 but we don't see 2. Why? Because we won't see all the numbers in between. The more we see the better. Because that gives us more randomization. Let's see it graphically,



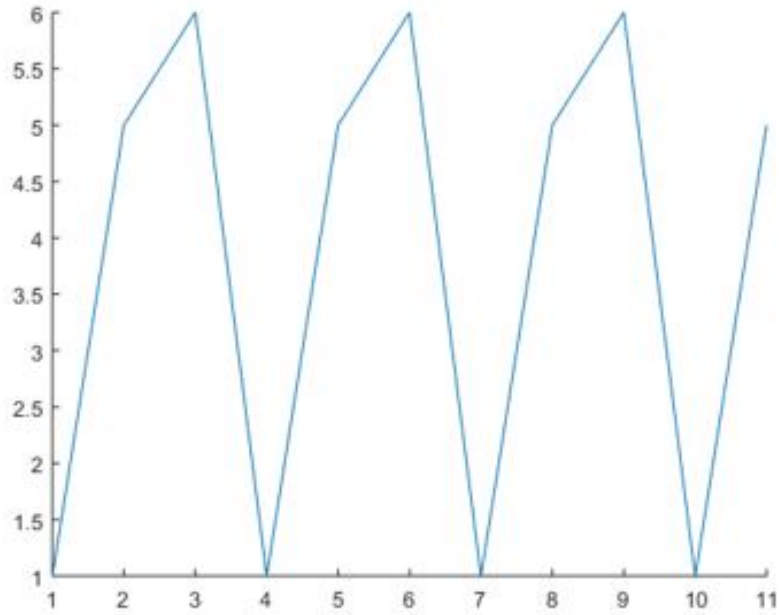
It is clear from the graph that the numbers start repeating itself after 5 and it seems a pretty selection if we want random numbers below 6.

Now, Let's change the mod value and see what happens, m = 6 gives us this graph,

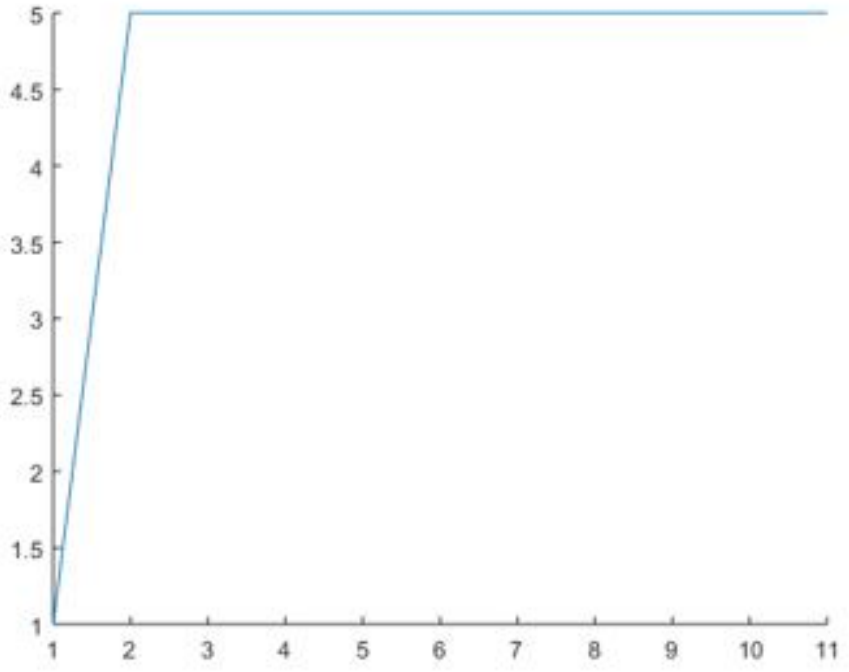


It gives us a really bad randomization. It is because 5 was prime where is 6 is not. Thus we can say that 5 was a better choice.

Let's try 7,

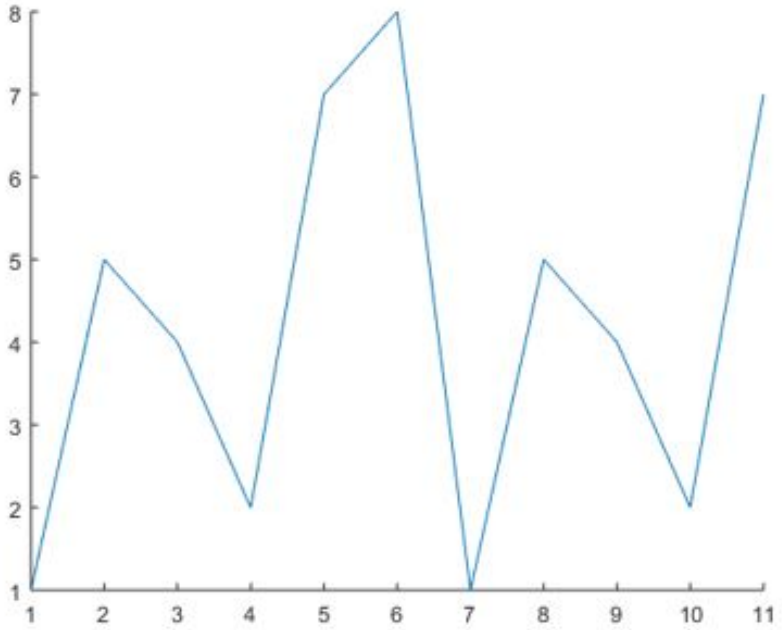


As we can see 7 is better choice than 6 obviously. It is because 7 is a prime. Let's try 8,



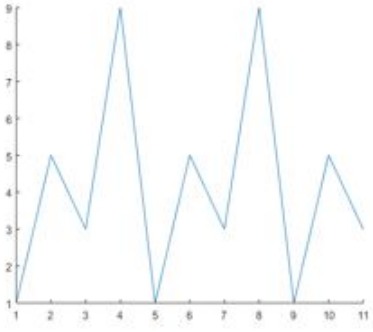
As we can clearly see 8 is the worst option until now. Because it will end up on the same random number always after the first try and will always return us 5. So, by this time we can clearly understand how important the choice of constants is for LCG.

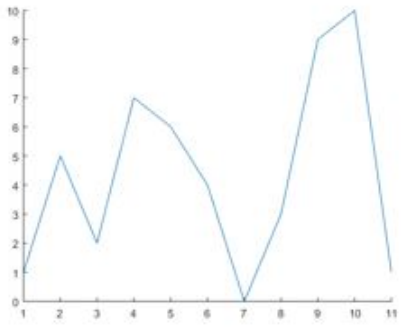
Moving on, setting mod value to 9 gives us,



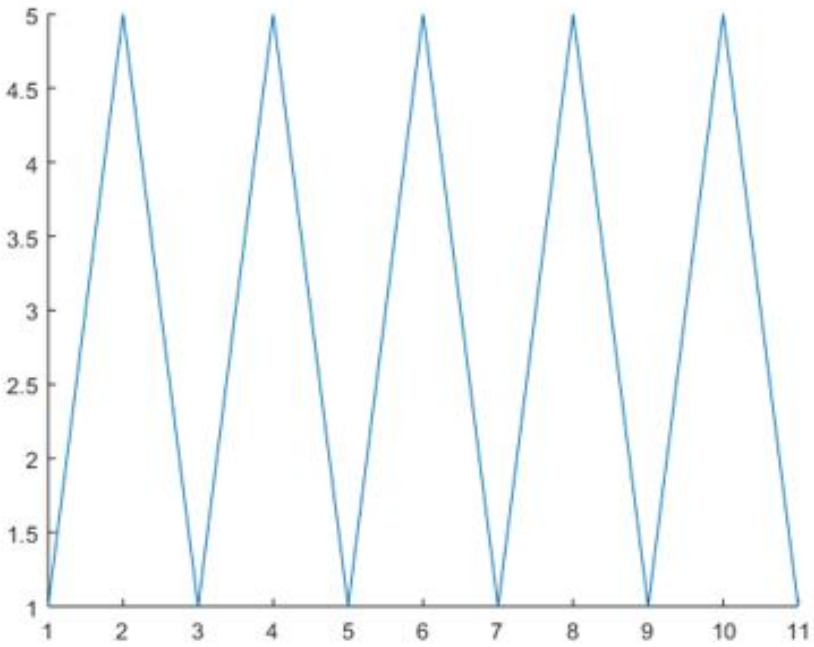
This is a really good choice. As we can see as the mod value is increasing we have more numbers to choose from thus giving us a really good randomization. But just increasing the value won't always give us better output, as we saw a value 6 which is higher than 5 gives us worse randomization. And a value 8 up until now gives us worse.

Thus we have to do trial and error to find the correct set of constants suitable for us. Let move on and compare the mod value of 10 and 11 now,



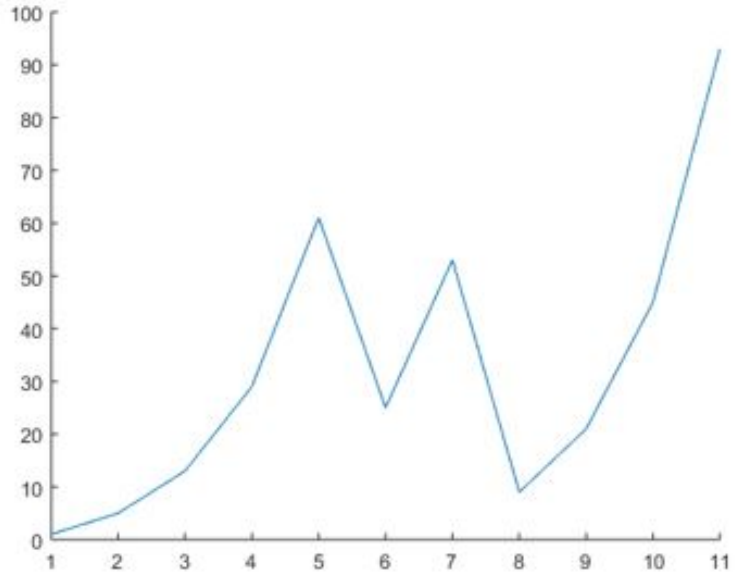


We see, 10 is okay, but 11 is much better. Because 11 is a prime and as of 12 it is not.



This is because 12 is not only not prime, it is divided by 2, 3, 4, 6, meaning it is as far from being prime as possible, giving us unwanted results.

Now let's go ahead and increase the mod value to 100.



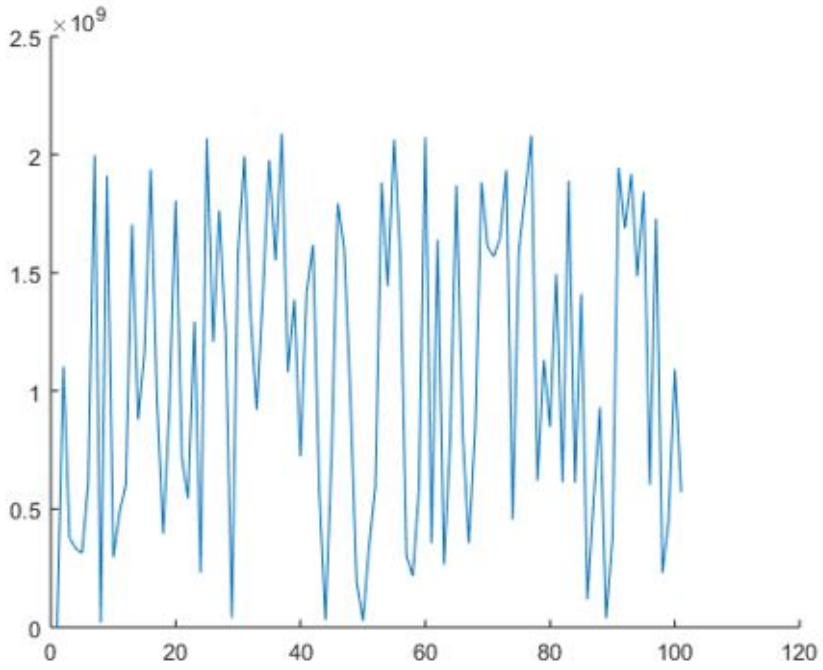
What's to be noted here is that 100 is not prime but it is a high enough number to get a randomized pattern. So to sum up, choosing a large enough number gives us better randomization than just choosing a prime one.

Source	modulus *m*	multiplier *a*	increment *c*	output bits of seed in *rand()* or *Random(L)*
Numerical Recipes	2^{32}	1664525	1013904223	
Borland C/C++	2^{32}	22695477	1	bits 30..16 in *rand()*, 30..0 in *lrand()*
glibc (used by GCC)[9]	2^{31}	1103515245	12345	bits 30..0
ANSI C: Watcom, Digital Mars, CodeWarrior, IBM VisualAge C/C++ [10] C99, C11: Suggestion in the ISO/IEC 9899 [11]	2^{31}	1103515245	12345	bits 30..16
Borland Delphi, Virtual Pascal	2^{32}	134775813	1	bits 63..32 of (*seed* * *L*)
Turbo Pascal	2^{32}	134775813 ($0x8088405_{16}$)	1	
Microsoft Visual/Quick C/C++	2^{32}	214013 ($343FD_{16}$)	2531011 ($269EC3_{16}$)	bits 30..16
Microsoft Visual Basic (6 and earlier)[12]	2^{24}	1140671485 ($43FD43FD_{16}$)	12820163 ($C39EC3_{16}$)	
RtlUniform from Native API[13]	$2^{31} - 1$	2147483629 ($7FFFFFED_{16}$)	2147483587 ($7FFFFFC3_{16}$)	
Apple CarbonLib, C++11's `minstd_rand0` [14]	$2^{31} - 1$	16807	0	see MINSTD
C++11's `minstd_rand` [14]	$2^{31} - 1$	48271	0	see MINSTD
MMIX by Donald Knuth	2^{64}	6364136223846793005	1442695040888963407	
Newlib, Musl	2^{64}	6364136223846793005	1	bits 63...32

Here we can see some common parameters in use for LCG. These are widely used to generate pseudo random numbers. These values are thoroughly tested and made sure that it gives us good randomization.

Let's consider glibc which is used by GCC. Now let's put on the parameters and see the graph,
m = pow(2, 31)
a = 1103515245
c = 12345

Putting the values in the equation we get the graph,



As we can see from the graph we took the first 100 values and see no repetition and the values are quite randomly distributed.

Chapter 4

Implementation and Analysis

Here is the implementation of the basic LCG algorithm implemented and tested using MATLAB2015.

```
seed_x = 1;

x = zeros(10);

y = zeros(100);

x(1) = seed_x;

a = 2;

c = 3;
```

```
m = 101;
j = 1;
for j = 1 : 10
   t = a * x(j) + c;
   x(j+1) = mod(t, m);
end
```

```
figure;
hold on
for ii = 1:10
 plot(x(:, ii))
end
```

Here's the algorithm which uses LCG that fits into the current framework implemented using MATLAB2015.
length_of_final_y represents the number of columns in the final key and length_of_final_x represents the number of rows.
length_of_session_key_x represents the matrix from which the random indices will be selected from.

length_of_final_x = 100;
length_of_final_y = 100;
length_of_session_key_y = 50;

At first random seed value is generated and saved in seed_x which gives a random value between 1 and length_of_the_final_y.

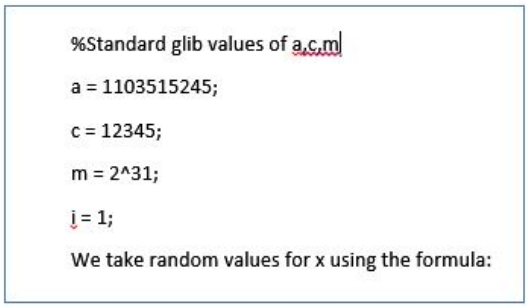
```
%generating random seed
%it will passed to the other party
seed_x = round(1 + (rand(1) * length_of_final_x));
```

seed_x = round(1 + (rand(1) * length_of_final_x));

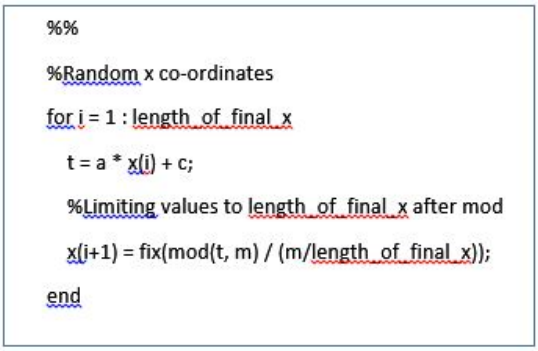
x and y will represent out final index matrices where for example y(1), x(1) represents the first index value. We initialize both as 0. We also initialize the first value of x(1) as seed.
Here, we will generate random values for x matrix only and y will increment by 1. This is because x represents the different values of the same sensor on a definite y. As we are using the heterogeneous nature of IoT devices for this, we want to take values from all the sensors where each sensor represents a column and the rows represent different values for the same sensor.
Thus, randomizing x will give us a random value from a sensor and incrementing y will ensure that we are using the heterogeneity by taking values from every sensor.

```
%declaring zero matrix initially
x = zeros(length_of_final_x,1);
y = zeros(length_of_final_y,1);
%Initializing with seed values
x(1) = seed_x;
```

We take standard glib values which is used is GCC. These are standard tested values which have been used for a long time.



X(i+1) = (a * X(i) + c) mod m
Then we divide it using m/length_of_final_x and then fix it to integer to constraint the values from 1 to length_of_final_x.



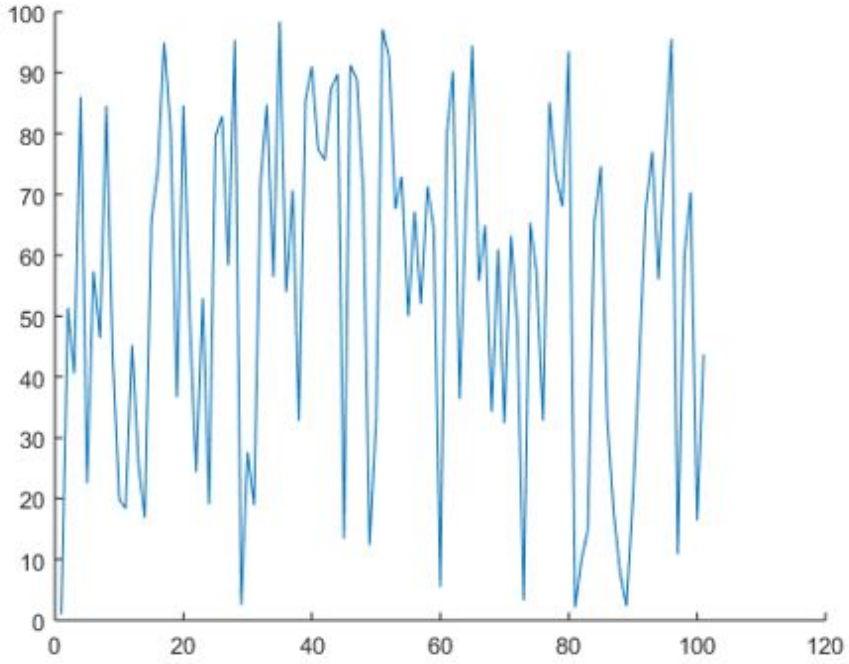
We increment values in y matrix by 1 until we reach the length_of_final_y. If we still need to get going we start from 1 again.

```
%Incrementing y co-ordinates
j = 1;
for i = 1 : length_of_final_y
  if j > 50
    j = 1;
  end
  y(i) = j;
  j = j+1;
end
```

We plot the x values to see our randomization.

```
%
%plotting x values
figure(1);
for ii = 1:length_of_final_x
 plot(x(:, ii));
end
```

This produce a graph of randomized values which look something like this(taking a seed value as random).



Thus, out final session index matrix becomes the matrix made using y and x. Every y, x pairs is one index. For example the first index is y(1), x(1).
So, we just optimized the mentioned framework by reducing the passing of a big matrix to just passing of a single number.
The pseudo random generator can be more optimized using even more lightweight generator than LCG. Perhaps a twofold security measure can be implemented too.

Chapter 5

Conclusion

In this paper we proposed a modified version of the related paper "A Framework for secured communication over the untrusted cloud" where instead of passing a huge index matrix we are passing only a single number using LCG Algorithm which uses pseudo random number generator. Using this number both communicating parties create their final session key. This reduces data passing by a large extent and prevents congestion in the network. Our framework is thus more efficient and lightweight than the previous framework.

References

[1] Sepp Hochreiter; Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:17351780, Nov. 1997.

[2] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou. Understanding the mirai botnet. In *26th USENIX Security Symposium*,Vancouver, BC, Canada, August 2017.

[3] Michele De Donno, Nicola Dragoni, Alberto Giaretta,and Angelo Spognardi. Analysis of ddos-capable iot malwares. In *Proceedings of 1st International Conference on Security, Privacy, and Trust (INSERT)*, 2017.

[4] Diffie W, Hellman ME. New directions in cryptography. *IEEE Transactions on Information Theory*1976; 22(6):644–654 November.

[5] Broder A, Mitzenmacher M. Network applications of bloom filters: a survey. *Internet Mathematics* 2005; 1:485–509.

[6] Daniele Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*,Vancouver, BC, Canada, February 2002.

[7] https://whatis.techtarget.com/definition/pseudo-random-number-generator-PRNG

[8] https://en.wikipedia.org/wiki/Linear_congruential_generator