ANDROID BASED MESSAGE ENCRYPTION/DECRYPTION USING MATRIX

Dinesh P. Baviskar[1], Sidhhant N. Patil[2], Onkar K. Pawar[3]

[1]Technical Assistant, Computer Department, Ahinsa Polytechnic Dondaicha, Dhule (425408)
[2]HOD, Electrical Department, Ahinsa Polytechnic Dondaicha, Dhule (425408)
[3]Computer Department, Ahinsa Polytechnic Dondaicha, Dhule (425408)

Abstract

In this paper we are going to encrypt plain text to encrypted text. Message encryption is character value based encryption in which 3D message character matrix is replaced with 2D encryption value matrix. It can also know by some kind of message masking which work on upper level to provide maximum security. Password protection increases encrypted message file security.

***Keywords** – Matrix, ASCII Offset Value, DES, 3D Message Masking, and 2D.*

--***--

1. INTRODUCTION

Now a days Android mobiles are becoming popular everywhere. Anyone with few bucks can brought android mobile, also Akash Tablet for students work on Android OS.

Android is Open Source Platform so anyone can develop apps for Android platform also changes can made in Android OS. Android provide lot of programming power with Linux kernel.

Various apps are present in Android Market for text communication. Communication is required to complete your work efficiently. But security is must in communication. Due to open source security of android platform is less. So it is possible to find vulnerability of Android OS to use it for hacking purpose.

This android application is developed to provide security to text communication between Android devices using MATRIX encryption.

We developed MATRIX encryption technique to provide maximum security on ASCII level with less computing to work on low resource configuration android mobiles also.

1.1 Overview

Today data theft becoming very dangerous. We know words are dangerous than any other weapon. But weapon in wrong person's hands is most hazardous.

If sensitive information like atomic bomb is hacked by terrorists then they can use that info to create atomic bomb against us. Android is powerful but less secure cause of Open Source. So we need to provide security to your data in android device like SMS, files, Emails.

Everyone wants to keep his messages and files secret and if you find your messages? It will very bad for you. This can happen with your Android device. Many applications found in Android market which theft your data, messages, emails and lot more from your android mobile.

If we can't save data from steal then we need to convert it as it will no longer use by hacker. This can achieve by encryption. Current encryption techniques are good but they are developed for computers which have large processing power, memory but there is no special technique for android device. Android devices low processing power and memory so we need to optimize encryption technique for android like Delvik Virtual Machine.

This MATRIX encryption technique is fully developed and optimized for android devices. So it can use on android mobile efficiently.

2. EXITING SYSTEM & PROBLEM DEFINITION

DES algorithm and Six ways to break DES

DES (Data Encryption Standard) is a symmetric cryptographic algorithm which was adopted in January 1977 as a standard (see [1]) for protecting non-classified information in the United States by the former National Bureau of Standards (now known as National Institute of Standards and Technology). It is widely used for protecting sensitive information and for the authentication of banking transactions, for example. We propose here to present six different ways to break DES, the last one being currently analysed at the LASEC.

2.1 Exhaustive Key Search

DES is an algorithm which encrypts 64 bits blocks of data using a 56 bits secret key. A common scenario is the following: we have an encrypted block at disposal, we have some information about the plaintext (we know that it is an ASCII text, or a JPEG image, for example) and we would like to recover the secret key.

The simpler method is to try to decrypt the block with all the possible keys. The information we have on the clear text will allow us to recognize the right key and to stop the search. In average, we will have to try 36'028'797'018'963'968 (36 millions of billions) of keys. Knowing that a common modern PC can check about one to two millions keys each second, this represents a work time of about 600 to 1200 years for a single machine.

2.2 A Dedicated Machine

An exhaustive search is quite time consuming for a single PC, but it is possible to do better. In 1998, the EFF (Electronic Frontier Foundation has built a dedicated machine ([10]) in order to show to the world that DES is not (or no more) a secure algorithm. Deep Crack, that's the name of the machine, costs $200'000 and is built with 1536 dedicated chips. Deep Crack is able to recover a key with the help of an exhaustive search in 4 days in average, checking 92 billion of keys each second. Knowing the budget of electronic intelligence agencies (for example, the National Security Agency in the USA), it is easy to be pessimistic on the security of DES against such organizations!

2.3 A Huge Cluster of Computers

One needs not even a lot of money to break DES. Volunteers who are ready to donate their machine's idle time and the Internet are sufficient. In January 1999, Distributed.Net, an organization specialized in collecting and managing computer's idle time, broke a DES key in 23 hours! More than 100'000 computers (from the slowest PC to the most powerful multiprocessors machines) have received and done a little part of the work; this allowed a rate of 250'000'000'000 keys being checked every second.

2.4 A Time-Memory Tradeoff

An exhaustive search needs a lot of time, but negligible memory at all. It is now possible to imagine an scenario: we have a lot of available memory, and we are ready to precompute for all the possible keys **k** the encrypted block **y** corresponding to a given block **x** of data and storing the pairs **(y, k)** . So we will be able to find very quickly the right key if we have at disposal an encrypted version **x'** of our known block with an unknown key **k'** by searching in this kind of dictionary. This method becomes to be interesting in the case where we have more than one key to find and we have enough memory at disposal.

In 1980, Martin Hellman proposed in [2] a time-memory tradeoff algorithm, which needs less time than an exhaustive search and less memory than the storing method. His algorithm needs in the order of 1000 GB of storage possibilities and about 5 days of computations for a single PC.

2.5 Differential Cryptanalysis

In 1990, Biham and Shamir, two Israeli cryptographers working at the Weitzmann Institute, have invented (see [3]) a new generic technique to break symmetric algorithms called the differential cryptanalysis. It was the first time that a method could break DES in less time than an exhaustive search.

Imagine that we have a device which encrypts data with a hard-wired secret key, and imagine furthermore that we don't have the tools needed to "read" the key in the chip. What we can do is to choose some blocks of data and to encrypt them with the device.

The data analysis phase computes the key by analyzing about 2^{47} chosen plaintexts. A big advantage of this attack is that its probability of success increases linearly with the number of available chosen plaintexts and can thus be conducted even with fewer chosen plaintexts. More precisely, the attack analyses about 2^{14} chosen plaintexts and succeeds with a probability of 2^{-33}.

2.6 Linear Cryptanalysis

Another very important generic method to break ciphers is the linear cryptanalysis ([4]), which was invented in 1994 by a Japanese researcher working at Mitsubishi Electronics, Mitsuru Matsui. If we have 2^{43} = 8'796'093'022'208 known plaintext-cipher text pairs at disposal, which represents 64'000 GB of data, it is possible to recover the corresponding key in a few days using linear cryptanalysis. It is the most powerful attack on DES known at this time.

A current research project at the LASEC is the cost analysis of this attack. We have first implemented a very fast DES encryption routine using advanced techniques on a common Intel Pentium III architecture; this routine is able to encrypt at a rate of 192 Mbps on a PIII 666MHz processor. We have then implemented the attack; it is currently running on 18 CPU's, breaking a DES key in 4 days.

The goal of this project is to do a better statistical analysis on its complexity and on its success probability. First experimental and theorical results have shown that a linear cryptanalysis needs in reality less time as estimated by Matsui in 1994.

2.7 Conclusion from Existing System

There is no special encryption technique for android platform. Existing some encryption techniques are good in security but very costly in resource requirement and some encryption techniques are not costly in resource requirement but they are not secure

If we use costly encryption technique then application may freeze in low configuration android mobiles this will make negative impact on users.

But we can’t use other encryption technique for security reason so there should be an optimized technique which provide maximum security and requires less resource.

2.8 Project Plans

Our Plan is To Develop an encryption technique for android devices. These Themes We Will Be Implementing in project

- Encrypt text files and text message on android device using MATRIX technique.
- Internet communication
- Encrypt GTalk text communication using MATRIX.

Proposed System

This MATRIX encryption algorithm is developed specially for android devices. It can use with very low configuration android devices also. It is secure also; MATRIX is used to achieve dynamic encryption values with change in location of characters.

This algorithm uses only 8-10 % cpu for encryption and only 5% for decryption. These are very less numbers than other algorithm. Also it is very secure as third person doesn't know size of MATRIX then it became very difficult to crack message and total no of possibilities to generate MATRIX values are 3.2e+660

Above value is very big to crack message and it is useless if you don't know MATRIX size so this encryption algorithm is very hard to crack.

That means it can encrypt message using low computing without compromising with security.

3. SYSTEM IMPLEATATION

3.1 Matrix Encryption

Message Matrix

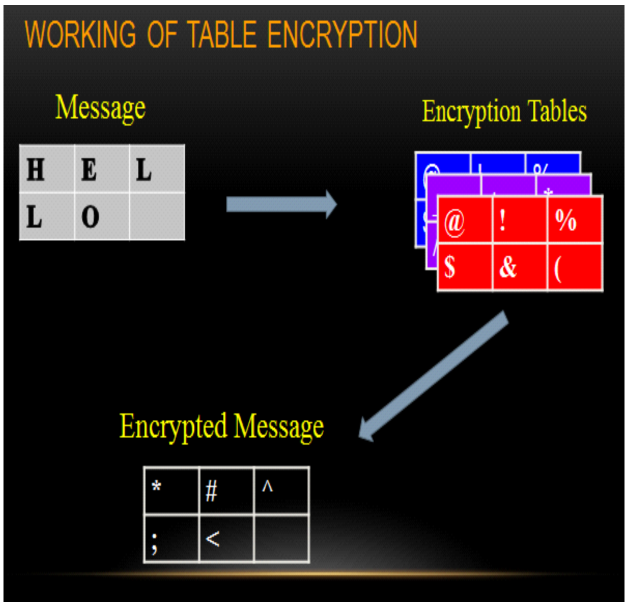
In this project we are going to encrypt plain text to encrypted text. Message encryption is character value based encryption in which 3D message character matrix is replaced with 2D encryption value matrix. It can also know by some kind of message masking which work on upper level to provide maximum security. Password protection increases encrypted message file security.

- **Encryption Values:-** These encryption matrixes contain encrypted values.
- **Unique Matrix:-** Every character has unique matrix.
- **Dynamic Values:-** Encryption value changes with change in location.

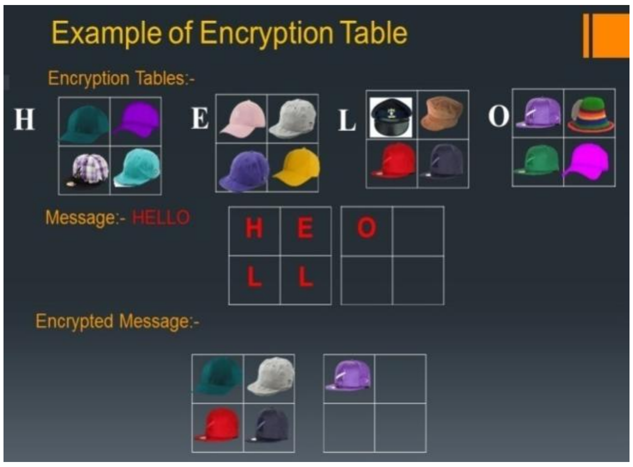
*	%	#	@
!	`	{	;
"	.	/	&

This MATRIX algorithm is developed to encrypt only text data. Currently encryption MATRIX is created for 99 most commonly used characters. We can increase number of characters.

Features of MATRIX:-
1. Protection from brute force attack and dictionary attack.
2. MATRIX size is automatically generated.
3. It can detect message is encrypted or not for GTalk or other chatting.



In above we can see working of MATRIX encryption.
It is as follows:
1. At first message is converted into 3D MATRIX.
2. Then MATRIX containing encryption values for each character are generated.
3. Characters from Message matrix are replaced using encryption values of proper characters and location.
4. This encrypted message is stored in .o2k file with other information which is required to decrypt message.



In above example we can see a simple example of MATRIX encryption with feature of dynamic values.

Matrix encryption is works on overlapped matrix for every character.

Starting value of each matrix is greater than previous matrix by one value.

A: [ID=0]

1	2	3	4	5	6	7	8	Encryption Value
0	1	2	3	4	5	6	7	Location Index

B: [ID=1]

2	3	4	5	6	7	8	9	Encryption Value
0	1	2	3	4	5	6	7	Location Index

C: [ID=2]

3	4	5	6	7	8	9	10	Encryption Value
0	1	2	3	4	5	6	7	Location Index

Sample:
Take a message “ABBCAAABABB”
Length of message= 11

For this encryption we will consider above encryption matrix of character A,B and C.
Matrix size is 8

Formula:
1. For Encryption:
Encryption value=(ID+Base Encryption Value)+Location Index

2. for Decryption:
ID=(Encryption value- Location Index)-Base Encryption Value

For First Letter from message (Letter is ‘A’)
Encryption Value=(0 + 1) + 0 = 1 i.e. ID = 0, Base Encryption Value = 1 and Location Index = 0

For second Letter from message(Letter is ‘B’)
Encryption Value=(1 + 1) + 1 = 3 i.e. ID = 1, Base Encryption Value = 1 and Location Index = 1

For third Letter from message (Letter is ‘B’)
Encryption Value=(1 + 1) + 2 = 4 i.e. ID = 1, Base Encryption Value = 1 and Location Index = 2

For fourth Letter from message (Letter is ‘C’)
Encryption Value=(2 + 1) + 3 = 6 i.e. ID = 2, Base Encryption Value = 1 and Location Index = 3

For fifth Letter from message (Letter is ‘A’)
Encryption Value=(0 + 1) + 4 = 5 i.e. ID = 0, Base Encryption Value = 1 and Location Index = 4

For sixth Letter from message (Letter is ‘A’)
Encryption Value=(0 + 1) + 5 = 6 i.e. ID = 0, Base Encryption Value = 1 and Location Index = 5

For 7th Letter from message (Letter is ‘A’)
Encryption Value=(0 + 1) + 6 = 7 i.e. ID = 0, Base Encryption Value = 1 and Location Index = 6

For 8th Letter from message (Letter is ‘B’)
Encryption Value=(1 + 1) + 7 = 9 i.e. ID = 1, Base Encryption Value = 1 and Location Index = 7

For 9th Letter from message (Letter is ‘A’)
Encryption Value=(0 + 1) + ? = ?i.e. ID = 0, Base Encryption Value = 1 and Location Index = 8

As matrix size is of 8 values, the range of Location Index is from 0 to 7 only.

If Location Index becomes greater than 7 then it again get initialized to zero value.

So Encryption value becomes

Encryption Value=(0 + 1) + 0 = 1 i.e. ID = 0, Base Encryption Value = 1 and Location Index = 0

For 10th Letter from message (Letter is ‘B’)
Encryption Value=(1 + 1) + 1 = 3 i.e. ID = 1, Base Encryption Value = 1 and Location Index = 1

For 11th Letter from message (Letter is ‘B’)
Encryption Value=(1 + 1) + 2 = 4 i.e. ID = 1, Base Encryption Value = 1 and Location Index = 2

So Encrypted Message becomes.

Origial Message	A	B	B	C	A	A	A	B	A	B	B
Encrypted Message	1	3	4	6	5	6	7	9	1	3	4

For Decryption:

ID = (Encryption value- Location Index)-Base Encryption Value

For 1st Value: ‘1’

ID = (1 – 0) – 1 = 0 i.e. Encryption value= 1, Location Index= 0 and Base Encryption Value =1

For 2nd Value: ‘3’

ID = (3 – 1) – 1 = 1 i.e. Encryption value= 3, Location Index= 1 and Base Encryption Value =1

For 3rd Value: ‘4’

ID = (4 – 2) – 1 = 1 i.e. Encryption value= 4, Location Index= 2 and Base Encryption Value =1
For 4th Value: ‘6’

ID = (6 – 3) – 1 = 2 i.e. Encryption value= 6, Location Index= 3 and Base Encryption Value =1

For 5th Value: ‘5’

ID = (5 – 4) – 1 = 0 i.e. Encryption value= 5, Location Index= 4 and Base Encryption Value =1

For 6th Value: ‘6’

ID = (6– 5) – 1 = 0 i.e. Encryption value= 6, Location Index= 5 and Base Encryption Value =1

For 7th Value: ‘7’

ID = (7 – 6) – 1 = 0 i.e. Encryption value= 7, Location Index= 6 and Base Encryption Value =1

For 8th Value: ‘9’

ID = (9 – 7) – 1 = 1 i.e. Encryption value= 9, Location Index= 7 and Base Encryption Value =1

For 9th Value: ‘1’

ID = (1 – 0) – 1 = 0 i.e. Encryption value= 1, Location Index= 0 and Base Encryption Value =1

Here Location Index was greater than 7, So it is again set to zero.

For 10th Value: ‘3’

ID = (3 – 1) – 1 = 1 i.e. Encryption value= 3, Location Index=1 and Base Encryption Value =1

For 11th Value: ‘4’
ID = (4 – 2) – 1 = 1 i.e. Encryption value= 4, Location Index=2 and Base Encryption Value =1

So Decrypted becomes.

Encrypted Message	1	3	4	6	5	6	7	9	1	3	4
Decrypted ID	0	1	1	2	0	0	0	1	0	1	1

Second Phase of Decryption:
Now convert Decrypted ID into Original Characters.

So Decrypted becomes.

Decrypted ID	0	1	1	2	0	0	0	1	0	1	1
Original Characters	A	B	B	C	A	A	A	B	A	B	B

So Message Becomes “ABBCAAABABB”
Which is exact same as original message.
This Second Phase of Decryption provides additional security to encrypted message.

We can reorder all alphabets and numbers also symbols in any order to generate ID for each character.

This ID is only known to sender and receiver so if anyone tries to decrypt message using formula, that person will stuck at second phase of decryption without knowledge of ID.

Also overlapped matrix size can be modified. It makes to crack message even more difficult, as the matrix is overlapped so is difficult to guess when first matrix ends and another starts.

That directly effects on ID and creates wrong decryption.

4. ANALYSIS

- **Feasibility Study:**
  Feasibility Study is conducted to see if the proposed system is a feasible one with all respects. Feasibility Study is lot of the system proposal according to its workability impact of the organization, ability to meet uses need and effective use of resources. There are three main aspects in the feasibility study. The feasibility of a project can be ascertained in terms of technical factors, economic factors, or both. A feasibility study is documented with a report showing all the ramifications of the project. In project finance, the pre-financing work is to make sure there is no "dry rot" in the project and to identify project risks ensuring they can be mitigated and managed in addition to ascertaining "debt service" capability.

- **Economic Feasibility:**
  In economic feasibility cost/benefit analysis is done. Here we determine the benefits and time savings that are expected from the system and compare them with cost. The proposed system is economically feasible. We developed java package having classes and methods for MATRIX encryption and decryption so it will very easy and less costly to implement in another application. Since the cost of the system is only the implementation cost of the system. There is no need spend any monthly thereafter. Since benefits outweigh the cost. It is economically feasible.

- **Operational Feasibility:**
  An operationally feasible system is one that will be used effectively after it has been developed. If users have difficulty with a new system, it will not produce the expected benefits. The proposed system is found to be operationally feasible because of the following reasons. It is very simple in use. There is no difficulty in using the front end which has been developed. Even the users who don’t have any knowledge in android mobile the user friendliness and help section provides them convenience and case. The system is designed, in such a way that not only the person currently handling this work can operate the system but a person who is new to the system with case. Hence this system is found to be operationally feasible.

- **Technical Feasibility:**
  Technical feasibility centers on the existing system and to the extent it can support the proposed system. This encryption package and application is built in java language so they are platform independent. This encryption can done on computer also using this package. Hence this system is found to be technical feasible.

- **Market Feasibility:**
  This is a generalized project so that it can be used in any application and service like SMS, Emails, GTalk, Facebook, Twitter etc. The existing traditional system is not optimized for android devices. Proposed system use less computing resources efficiently and do not compromise with security. So new system is not costly and provides better security.

4. SYSTEM REQUIRMENT AND SPECIATION

4.1 Requirements to Develop/Run Android Application on PC:-

Hardware Requirements
- Minimum 1Gb RAM,
- Minimum 20 GB HDD
- Dual Core processor or above

Software requirements
- Eclipse
- JDK compiler 1.5 and above
- SDK manager Revision 21
- ADT 21.0.0
- At least one Android Platform Package (prefer API 10)

4.2 Requirements to Run Android Application on Mobile:-

- 600MHz processor
- 128 MB RAM
- Android-2.3 and above OS on Mobile/Simulator
- Minimum API Level 10
- 525 KB memory on SD card or phone memory.
- At least Medium Screen Density Mobile

4.3 Connectivity Requirements:-

- In order to use encrypted message service over internet your mobile/emulator (PC) must connected to internet.
- To use GTalk chat service you must have your own account on that service.

5. CONCLUSION

The Security of conversion on social website and data sending becomes major issue especially in case of Google. This paper present to protect conversion on social website and data sending to provide security to generate key for this purpose. In the proposed scheme data encryption algorithm is used and also 3 D MATRIXES are used.This technique is based on dynamic values for a character so it becomes very difficult to identify actual message.

6. FUTURE DEVELOPMENT

- Emails, Facebook communication, Gtalk messages and for all text messaging services this encryption technique can provide private protection to your messages.
- We can increase security by arranging blocks of message in different ways.
- Word document, PDF files encryption support can provide in future.

REFERENCES

[1]. Information Security, http://www.infosec.gov.hk/english/technical/ files/ short.pdf.
[2]. Micro System Center, http://technet.microsoft.com/en-us/library/ cc181234.aspx.
[3]. M. Toorani and A. A. BeheshtiShirzai, SSMS - A Secure SMS Messaging Protocol for the M-Payment Systems, IEEE Symposium on
Computers and Communications, 2012, 700-705.
[4]. Marko Hassinen, SafeSMS - End-to-End Encryption for SMS Messages, IEEE International Conference on Telecommunications, 2008,
359-365.
[5]. S. Jahan, M. M. Hussain,M. R.Amin and S. H. Shah Newaz, A Proposal for Enhancing the Security System of Short Message Service in
GSM, IEEE International Conference on Anti-counterfeiting Security and Identification, 2008, 235-240.
[6]. Mary Agoyi and DevrimSeral, SMS Security: An Asymmetric Encryption Approach, IEEE International Conference on Wireless and
Mobile Communications, 2010, 448-452.
[7]. Na Qi Jing Pan Qun Ding, The Implementation of FPGA-based RSA Public-Key Algorithm and Its Application in Mobile-Phone SMS
Encryption System, IEEE International Conference on Instrumentation, Measurement, Computer, Communication and Control, 2011,
700-703.
[8]. Ch. Rupa and P.S. Avadhani, Message Encryption Scheme Using Cheating Text, IEEE International Conference on Information
Technology, 2009, 470-474.
[9]. Rishav Ray, JeeyanSanyal, Tripti Das, KaushikGoswami, Sankar Das and AsokeNath, A new Randomized Data Hiding Algorithm with Encrypted Secret Message using Modified Generalized Vernam Cipher Method: RAN-SEC algorithm, IEEE Information and Communication Technologies World Congress, 2011, 1211-1216.
[10]. Hongbo Zhou, Mutka and Lionel M. Ni, Multiple-key Cryptography-based Distributed Certificate Authority in Mobile Ad-hoc Networks, IEEE proceedings of GLOBECOM, 2005, 1681-1685.