Afr. J. Comp. & ICT
Vol. 1, No. 2, 2008, pp 27 – 33
ISSN 2006-1781

COMPUTATIONAL ANALYSIS OF GSM SECURITY ALGORITHM

O.C. Abikoye
Department of Computer Science,
University of Ilorin,
Ilorin, Kwara State,
Nigeria
Email:kemi_adeoye@yahoo.com
&
'Dele Oluwade*
P.O.Box 20253, U.I.P.O.,
Ibadan, OYO 200005,
Nigeria
Email:deleoluwade@yahoo.com
(*Present institutional address: Department of Library and Information Technology, Federal University of Technology, Minna, Niger State, Nigeria)

Received 10 May 2008

Thesis Communication

ABSTRACT

The security and authentication mechanisms incorporated in the Global System of Mobile Network (GSM) make it the most secure mobile communication standard currently available, particularly in comparison to the analog systems. GSM security and encryption algorithm are used to provide authentication and radio link privacy to users on GSM network. Security in GSM consists of the following aspects: subscriber identity authentication, subscriber identity confidentiality, signaling data confidentiality, and user data confidentiality. These various security aspects are addressed by the user authentication algorithm(called A3), the signal encryption algorithm(A5) and the encryption key generation algorithm(A8). Both A3 and A8 algorithms are similar in functionality and are commonly implemented as a single algorithm called COMP128. In this paper, a computational time complexity analysis of COMP 128 is performed. It is shown that the running time of user authentication algorithm COMP128 is constant, meaning that the number of executions of basic operations in COMP128 algorithm is fixed and so the total time is bounded by a constant.

Keywords and Phrases: Computational complexity, GSM, Security, Algorithm, A3, A8, COMP128

I INTRODUCTION

GSM (Global System of Mobile Network) is one of the most widely used mobile telephone system. As communication with a mobile phone occurs over a radio link it is susceptible to attacks that passively monitor the airways (radio paths). The security and authentication mechanisms incorporated in the GSM make it the most secure mobile communication standard currently available, particularly in comparison to the analog systems. GSM security and encryption algorithm are used to provide authentication and radio link privacy to users on GSM network. Security in GSM consists of the following aspects: subscriber identity authentication, subscriber identity confidentiality, signaling data confidentiality, and user data confidentiality. Several security functions were built into GSM to safeguard subscriber privacy. The GSM specification addresses three key security requirements: authentication, confidentiality and anonymity. There are three proprietary algorithms used to achieve authentication and confidentiality. These are known as A3, A5 and A8 [4]. While A3 is used to authenticate the SIM (Subscriber Identity Module) for access to the network (user authentication algorithm), both A5 and A8 achieve confidentiality by scrambling the data sent across the airways; A5 is the signal encryption algorithm and A8 is the encryption key generation algorithm. Anonymity is achieved by use of temporary identities (TMSI). The A3 and A8 algorithms of GSM are key- dependent one-way hash functions. The GSM A3 and A8 algorithms are similar in functionality and are commonly implemented as a single algorithm called COMP128. To the best of the authors' knowledge, even though these algorithms are very important, no computational analysis has hitherto been performed on them. In this paper, a computational time complexity analysis of the combined algorithm, COMP 128, is performed. This is a GSM standard authentication algorithm which may be used by all networks which do not wish to develop a proprietary algorithm.

Analyzing an algorithm has come to mean predicting the resources that the algorithm requires. Occasionally resources such as memory, communication bandwidth or logic gates are of primary concern but most often it is computational time that is always measured. Generally by analyzing several candidates' algorithms for a problem, a most efficient one can be easily identified. Such analysis may indicate more than one viable candidate, but several inferior algorithms are usually discarded. The performance of any program/algorithm can be analyzed. By the performance of a program/algorithm we mean the amount of computer memory and time needed to run a program. There are two approaches to determine the performance of a program. One is analytical and the other is experimental. In computational analysis analytical methods are used, while in performance analysis experiments are conducted. The space complexity of a program is the amount of memory it needs to run to completion while the time complexity of a program is the amount of computer time it needs to run to completion [3]. Time complexity is always needed to provide an upper limit on the amount of time the program will run, a satisfactory real –time response and also to decide on which solution to use if there are many alternatives based primarily on the performance difference among these solutions.

In general, the time taken by an algorithm grows with the size of the input, so it is traditional to describe the running time of a program as a function of the size of its input. The best notion for input size depends on the problem being studied for many problems such as sorting or computing Discrete Fourier Transform, the most natural measure is the number of items in the input. For example, the array size *n* for sorting. For many other problems, such as multiplying two integers, the best measure of input size is the total number of bits needed to represent the input in ordinary binary notation. The running time of an algorithm on a particular input is the number of primitive operations or "steps" executed. It is convenient to define the notion of step so that it is as machine –independently as possible.

In [7], the computational complexity of an algorithm useful in numerical analysis was studied and it was shown that the algorithm has a linear time complexity. In another paper [8], an algorithm for finding the error pattern of a uniform digital code was proposed and analyzed. It was shown that the (asymptotic) time complexity of the PASCAL-like algorithm is also of linear time. In the present paper, it is shown that the running time of COMP128 is constant, meaning that the number of executions of basic operations in COMP128 algorithm is fixed and so the total time is bounded by a constant.

II. COMP128 ALGORITHM

In this section, the COMP 128 algorithm is described. The algorithm has been extracted from [4].

```
(1)    (Load RAND into last 16 bytes of input)

       FOR i from 16 to 31
          X[i] = rand[i]
       END FOR

(2.1)  (Loop eight times)

       FOR i from 1 to 8
                     I=n

          (Load key into first 16 bytes of input)

          FOR j from 0 to 15
             X[j] = key[j]
          END FOR

(2.2)  (Perform substitutions)

          FOR j from 0 to 4
                FOR k from 0 to $2^j$ - 1
               FOR l from 0 to $2^{4-j}$ - 1
                     m = 1 + k * $2^{5-j}$
                 n = m +$2^{4-j}$
                y = (x[m] + 2 * x[n]) mod $2^{9-j}$
                z = (2 * x[m] + x[n]) mod $2^{9-j}$
                x[m] = table [j,y]
                x[n] = table [j,z]
                END FOR
            END FOR
        END FOR
```

```
(2.3)   (Form bits from bytes)
        FOR j from 0 to 31
            FOR k from 0 to 7
                bit [4*j+k] = the (8-k)th bit of byte j
            END FOR
        END FOR
```

```
(2.4)   (Permutation but not on the last loop)
        IF (i < 8) THEN
            FOR j from 0 to 15
                FOR k from 0 to 7
                    next bit = (8 x j + k) * 17 mod 128
                    Bit k of x[j + 16] = bit[next_bit]
                END FOR
            END FOR
        END IF
    END FOR
```

The vector x[ ] consists of 32 nibbles, the first 8 of which are taken as the output SRES.

III. COMPUTATIONAL TIME COMPLEXITY OF COMP 128 ALGORITHM

The time complexity of a program depends on the factors that the space complexity depends on. A program will run faster on a computer capable of executing 10^9 instructions per second than on the one that can execute only 10^7 instructions per second. Some compilers will take less time than others to generate the corresponding computer code. Smaller problem instances will generally take less time than larger instances.

The time taken by a program *p* is the sum of the compile time and the run (execution time). One way to estimate the time complexity of a program or method is to select one or more operations i.e through the use of operation counts. Such as add, multiply and compare and to determine how many of each is done. The success of this method depends on the ability to identify the operations that contribute most to all the time complexity.

Another way to estimate the time complexity of a method or program is through the use of step count. In step count method, we attempt to account for the time spent in all parts of the program/method. The step count is a function of the instance characteristics. Although any specific instance may have several characteristics (e.g the number of inputs, the number of outputs, the magnitude of the inputs and outputs), the number of step is computed as a function of some subset of these.

Let f(n) and g(n) be two non-negative functions. Then g(n) is said to be asymptotically bigger than f(n) iff $f(n)/g(n)\to 0$ as $n\to\infty$ [3]. The notation f(n) = O(g(n)) , where O is called 'big oh', is used when f(n) is asymptotically smaller than or equal to g(n)(which means g(n) is asymptotically greater than or equal to f(n)) i.e g(n) is an upper bound for f(n). More rigorously, f(n) = O(g(n)) iff there exists positive constants c and n_0 such that $f(n) \leq cg(n)$, for all $n \geq n_0$. Apart from the 'big oh' which is the most frequently used asymptotic notation, other notations include omega (Ω), theta (Θ), 'little oh'(o) and little omega(ω). Now, f(n) = Ω(g(n)) if f(n) is asymptotically bigger than or equal to g(n). This means g(n) is an asymptotic lower bound for f(n). f(n) = Θ(g(n)) iff there exist positive constants c_1 and c_2 , and n_0 , such that $c_1g(n) \leq f(n) \leq c_2g(n)$, for all $n > n_0$. i.e f(n) can be bounded both from above and below by the same function g(n), which implies that both of the following are valid:

f(n) = Ω(g(n)); f(n) = O(g(n))

The computational time complexity analysis of COMP128 algorithm is performed below.

1. Assignment takes a constant time Θ (k);

we have $n\Theta(k) = \Theta(nk)$ where n =16

$T_1 = \Theta(nk)$

2.1 assignment takes a constant time Θ (k)

$n\ j\ \Theta(k)$ where n=8 and j = 16

which give at most $\Theta(n^2k)$

$T_2 = \Theta(n^2k)$,

2.2 assignment , addition , multiplication and modulo each take constant times Θ (k),

where $n\ j\,k\,l\ \Theta(k) = \Theta(n^4k)$,

j = 4, k= 2^4 =16, l= 2^4 =16

$T_3 = \Theta(n^4k)$

2.3 assignment takes a constant time Θ (k)

where $n\ j\,k\ \Theta(k) = \Theta(n^3k)$ where n = 8, k = 8, j = 32

$T_4 = \Theta(n^3k)$

2.4 assignment , addition , multiplication modulo take constant times Θ (k)

where $n\ j\,k\ \Theta(k) = \Theta(n^3k)$ where j=16, n = 8, k = 8

$T_5 = \Theta(n^3k)$

Finally, T (n) = $T_1 + T_2 + T_3 + T_4 + T_5$

$$= \Theta(nk) + \Theta(n^2k) + \Theta(n^4k) + \Theta(n^3k) + \Theta(n^3k)$$

$$= \Theta(n^4k) \text{ where } n \le 32$$

$$= O(n^4k) \quad n = 32$$

since n has an upper bound which is 32 i.e we cannot have more than 32 loop and so the running time is constant.

T (n) = $\Theta(k)$

IV CONCLUSION AND DISCUSSION

The security mechanisms specified in the GSM standard make it the most secure cellular telecommunications system available. The use of authentication, encryption, and temporary identification numbers ensures the privacy and anonymity of the system's users, as well as safeguarding the system against fraudulent use. Even GSM systems with the A5/2 encryption algorithm or even with no encryption are inherently more secure than analog systems due to their use of speech coding, digital modulation, and TDMA channel access.

The running time of user authentication COMP128 algorithm is constant, T (n) = $\Theta(k)$, meaning that the number of executions of basic operations in COMP128 algorithm is fixed, hence, the total time is bounded by a constant.

The total memory requirement for the data and program storage of COMP128 algorithm depends on the total bytes occupied by the number of microprocessor instructions codes, the bytes of external memory, the bytes occupied by the variables used in the instruction codes.

In conclusion, GSM has experienced a lot of security breaches, despite all the breaches GSM is by far more secure than previous analog cellular systems and continues to be the most secure public wireless standard in the world.

POSTSCRIPT

This paper is part of the M.Sc (Computer Science) project of the first author [1] written under the supervision of the second author.

REFERENCES

[1] O.C. Abikoye, O.C, "Computational Analysis and Implementation of GSM Security Algorithm," *M.Sc Project* , Department of Computer Science, University of Ibadan, 2006
[2] Business Wire Press release "GSM Alliance Clarifies False & Misleading Reports of Digital Phone Cloning" ; http://jya.com/gsm042098.txt , 1998
[3] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, *Introduction to Algorithms*, The MIT Press Cambridge, 1992.
[4] "GSM System Security Study", Technical Information by Racal Research Ltd, Worton Drive, Worton Grange Industrial Estate, Reading, Berks. RG2 0SB, England, 10-1617-01 10th June 1988; http://jya.com/ /gsm061088.htm <accessed in 2005>
[5] Ellis Horowitz and Sartrj Sahni, *Fundamentals of Computer Algorithms*, Computer Science Press Inc, 1978.
[6] David Margrave, George Mason University, "GSM Security and Encryption";

www.hackcanada.com/blackcrawl/cell/gsm/gsm-secur/gsm-secur.html, <accessed in 2005>

[7] Dele Oluwade and O.A.Taiwo, "Asymptotic Time Complexity of an Algorithm for Generating The Coefficients of the Chebyshev Polynomials For The Tau Numerical Method," *Nigerian Journal of Mathematics and Application* (Journal of the Mathematical Association of Nigeria, Kwara State Branch), vol. 13, pp 131-141, 2000

[8] Dele Oluwade, C.O.Uwadia and J.O.A.Ayeni,"Asymptotic Time Complexity of an Algorithm for Finding The Error Pattern of a Uniform Digital Code," *Journal of Scientific Research and Development* (Journal of the Faculty of Science, University of Lagos), vol.6, pp 127-134, 2001