DESIGN OF AUTHENTICATION PROTOCOLS AND MESSAGE SECURITY ALGORITHMS IN CELLULAR NETWORKS

A THESIS

submitted in partial fulfillment of the requirements for the award of the degree

of

DOCTOR OF PHILOSOPHY

by

NEETESH SAXENA

DISCIPLINE OF COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

APRIL 2014

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work, which is being presented in the thesis, entitled **DESIGN OF AUTHENTICATION PROTOCOLS AND MESSAGE SECURITY ALGORITHMS IN CELLULAR NETWORKS**, in fulfillment of the requirement for the award of the degree of **DOCTOR OF PHILOSOPHY** and submitted in the **DISCIPLINE OF COMPUTER SCIENCE & ENGINEERING, Indian Institute of Technology Indore**, is an authentic record of my own work carried out during the period July, 2011 to April, 2014 under the supervision of Dr. Narendra S. Chaudhari, Professor, Discipline of Computer Science & Engineering, IIT Indore.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

Dated: Signature of the Student with date
. (NEETESH SAXENA)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Dated: Signature of the Thesis Supervisor
. (NARENDRA S. CHAUDHARI)

NEETESH SAXENA has successfully given his Ph.D. Oral Examination held on ....

Signature of Thesis Supervisor	Convener, DPGC
Date:	Date:

Signature of PSPC . Member #1	Signature of PSPC Member #2	Signature of External Examiner
Date:	Date:	Date:

Acknowledgements

The efforts taken in this PhD thesis would not have been possible without kind support and help of many individuals. I would like to extend my sincere thanks to all of them.

First and foremost, I am highly indebted and grateful to my thesis supervisor, Prof. Narendra S. Chaudhari, for his constant encouragement throughout my PhD course. The warm blessings, excellent comments, constructive suggestions, and thoughtful guidance given by him time to time shall carry me a long way in the journey of life, on which I am about to embark. Prof. Chaudhari gave me the freedom to pursue my PhD objectives, at the same time continuing to contribute valuable feedback, insightful discussions, and encouragement.

I express my profound sense of gratitude to Dr. Abhishek Srivastava (Assistant Professor, Discipline of Computer Science & Engineering) for his valuable time at the institute for the administrative purposes.

My PhD Student Progress Committee (PSPC) guided me through all these years. I would like to express my sincere thanks to the committee members, Dr. Aruna Tiwari (Assistant Professor, Discipline of Computer Science & Engineering) and Dr. Abhinav Kranti (Associate Professor, Discipline of Electrical Engineering), for their time, interest, and helpful comments and suggestions.

I am extremely thankful to Prof. Joachim von zur Gathen (b-it, cosec, Bonn, Germany) and his postdoctoral researcher Dr. Daniel Loebenberger (b-it, cosec, Bonn, Germany) for their valuable suggestions, constant supervision as well as for providing necessary information and support for the work I pursued at b-it.

I also take this opportunity to express my humble gratitude towards the members of IIT Indore including the office staff, laboratory technicians, and library staff for their kind co-operation and extended support offered to me by providing the required resources for my work. I wish to personally thank

Shailendra Verma, Prahlad Singh Panwar, Jaya Vakde, and Lalit Jain for their support.

I gratefully acknowledge the funding sources, which financially supported me during this PhD thesis work. I was supported with the research scholarship by Ministry of Human Resource and Development (MHRD) under teaching assistance-ship during my PhD work. I was honored to be a TCS Research Scholar and wish to thank Tata Consultancy Services Limited (TCS) for inviting me to discuss and present my research work in the TCS conference. I wish to acknowledge Dr. C. Anathram, Rahul Pandey, and Sachin Parkhi for their support during TCS Research Scholar Programme. My research work was also supported by the Department of Science and Technology (DST), Council of Scientific and Industrial Research (CSIR), Association for Computing Machinery (ACM), Applied Computer Security Associates (ACSA), Deutscher Akademischer Austausch Dienst (DAAD), and IIT Indore.

I would like to thank my friends at IIT Indore for all the great times that we shared. I truly thank to my friends for sticking by my side, even when I was irritable and depressed. I feel that I learned a lot about life, and strengthened the commitment and determination for living life to the fullest. I am particularly thankful to Jaya Thomas not only for her constant help, support, and valuable discussions but also for listening to me. I wish to thank Pooja Jain, Rajkumar Jain, Prakash Chandra Sharma, Ajay Pratap Singh, and Sambhaji Kadam for their helpful discussions. I also thank to Mahesh, Pramod, Vipul, Tanveer for their moral support.

Lastly, I would like to thank my family for all their love and encouragement. Their support has been unconditional during all these years and they have given up many things for me to be at IIT Indore.

Dated: (Neetesh Saxena)

Dedicated to my parents

Abstract

Various security and performance issues of authentication and key agreement (AKA) protocols exist in the cellular networks. We examine the security and performance issues of the authentication protocols present in second-generation (2G) Global System for Mobile Communications (GSM), 3G Universal Mobile Telecommunication System (UMTS), and 4G Long Term Evolution (LTE) cellular networks.

The efficient and secure AKA protocols for the cellular networks are proposed and analyzed. We propose SAKA protocol for the GSM network that overcomes the basic limitation of GSM-AKA protocol by providing bidirectional authentication. It reduces the storage space, overheads, and the required bandwidth by adopting a scheme of generating a delegation key. Similarly, the delegation key concept is used in the proposed ES-AKA and Secure-AKA protocols for the UMTS network, which lowers the bandwidth consumption and overheads during the mutual authentication. A puzzle-based solution scheme is introduced in the proposed Secure-AKA protocol that prevents the UMTS network from denial of service (DoS) attack. The proposed protocol for LTE network computes more functions at user and the server in order to protect the actual identity of user and key set identifier over the network without increasing the total number of bits to be transmitted as compared to the original LTE authentication protocol. Moreover, improved secure algorithms namely NewA3, NewA8, and NewA5 for the GSM network and MAES-128 algorithm for the UMTS network are proposed and implemented to enhance the security system of GSM and UMTS network architecture. In order to develop secure Short Message Service (SMS)-based applications in the cellular networks, the short messages are provided with end-to-end secure delivery of message information from the sender to the recipient. A secure and efficient SecureSMS protocol for secure delivery of value added services using SMS are proposed and analyzed, which provides the security services like authentication, confidentiality, integrity, and non-repudiation. Another, the EasySMS protocol is proposed that enables end-to-end secure transmission of SMS between the mobile users. The ciphering process of EasySMS is implemented by a modified AES algorithm (MAES with 256 bits block size and key size), where SubByte and ShiftRow of AES are swapped, and an alternative matrix is used (MixColumns) that is same as its inverse.

Furthermore, in various mobile services, it is always a challenge for the server to handle maximum number of authentication requests simultaneously based on its capacity to handle requests. The AKA protocols for SMS-based applications are extended to provide secure delivery of value added services and end-to-end SMS security to multiple recipients simultaneously, where the authentication server is able to handle multiple requests in a batch. These protocols maintain integrity during the transmission of messages and propose one time activation code in order to protect the actual identity of mobile user over the network. We have used a probability model to determine the number of malicious authentication requests in a batch. Further, an algorithm is also used to identify such users that are removed from the batch in order to perform re-batch authentication.

List of Publications

(A) In International Journals

(i) Published

[1] Saxena N., Chaudhari N.S. (2014), Secure-AKA: an efficient AKA protocol for UMTS networks, Wireless Personal Communication (WPC), Springer, ISSN: 0929-6212, 78(2), 1345-1373.

[2] Saxena N., Chaudhari N.S. (2014), EasySMS: a protocol for end-to-end secure transmission of SMS, IEEE Transactions on Information Forensics and Security (TIFS), ISSN: 1556-6013, 9(7), 1157-1168.

[3] Saxena N., Chaudhari N.S. (2014), SecureSMS: a secure SMS protocol for VAS and other applications, Journal of Systems and Software (JSS), Elsevier, ISSN: 0164-1212, 90, 138-150.

[4] Saxena N., Chaudhari N.S. (2013), SAKA: a secure authentication and key agreement protocol for GSM networks, CSI Transactions on ICT (CSIT), Springer, ISSN: 2277-9078, 1(4), 331-341.

[5] Saxena N., Chaudhari N.S. (2013), An enhanced NPA protocol for secure communications in GSM network, International Journal of Security and Networks (IJSN), Inderscience, ISSN: 1747-8413, 8(1), 13-28.

[6] Saxena N., Chaudhari N.S. (2013), Prevention of SMS against repudiation attack over the GSM network, Journal of Information Assurance and Security (JIAS), MIR Labs, ISSN: 1554-1010, 8(3), 156-166.

[7] Saxena N., Chaudhari N.S. (2012), An approach for SMS security using authentication functions, International Journal of Computer Application (IJCA), IJCA Special Issue on Communication Security Comnetcs(1),

Foundation of Computer Science, New York, USA, ISBN: 973-93-80864-65-2, 6-8.

[8] Saxena N., Chaudhari N.S. (2011), A secure digital signature approach for SMS security, International Journal of Computer Application (IJCA), Special Issues on IP Multimedia Communications, Foundation of Computer Science, New York, USA, ISBN: 978-93-80864-99-3, 98-102.

(ii) Communicated (Under Review)

[1] Saxena N., Chaudhari N.S., ES-AKA: an efficient and secure authentication and key agreement protocol for UMTS networks, Wireless Personal Communication, Springer.

[2] Saxena N., Chaudhari N.S., An improved authentication and key agreement protocol for 4G LTE networks, Journal of Information Security and Applications, Elsevier.

[3] Saxena N., Chaudhari N.S., An extended batch verification protocol for delivering value added services to mobile users, Computer Communications, Elsevier.

[4] Saxena N., Chaudhari N.S., Secure algorithms for SAKA protocol in the GSM networks, CSI Transactions on ICT, Springer.

[5] Saxena N., Chaudhari N.S., BOPSMS: a batch verification protocol for end-to-end secure SMS for mobile users, IEEE Transactions on Mobile Computing.

(B) In International Conferences

(i) Published as Full Papers in Proceedings

[1] Saxena N., Chaudhari N.S. (2014), NS-AKA: an improved and efficient AKA protocol for 3G (UMTS) networks, International Conference on Advances in Computer Science and Electronics Engineering (CSEE′14), Kuala Lampur, Malaysia, pp. 220-224.

[2] Saxena N., Chaudhari N.S. (2013), VAS-AKA: an efficient batch verification protocol for value added services, IEEE International Conference on System, Man and Cybernetics (SMC′13), Manchester, UK, pp. 1560-1565.

[3] Saxena N., Chaudhari N.S., Thomas J. (2013), Solution to an attack on digital signature in SMS security, 5th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO′13), Hammamet, Tunisia, pp. 1-6.

[4] Saxena N., Chaudhari N.S. (2013), NPA: protocol for secure communications in GSM cellular network, 10th Annual IEEE Consumer Communications & Networking Conference (CCNC′13), Wireless Communication Track, Las Vagas, USA, pp. 408-413.

[5] Saxena N., Chaudhari N.S. (2012), Secure encryption with digital signature approach for short message service, World Congress on Information and Communication Technologies (WICT′12), Trivandrum, India, pp. 803-806.

[6] Saxena N., Chaudhari N.S. (2012), A secure approach for SMS in GSM network, ACM CUBE International IT Conference & Exhibition (CUBE′12), Pune, India, pp. 59-64.

[7] Saxena N., Chaudhari N.S., Prajapati G. L. (2012), An extended approach for SMS security using authentication functions, 7th IEEE International Conference on Industrial Electronics and Applications (ICIEA′12), Singapore, pp. 663-668.

[8] Saxena N., Chaudhari N.S. (2012), Analysis of el-gamal with diffie-hellman scheme using hash-function for message security, IEEE International Students Conference on Engineering and Systems (SCES′12), MNNIT Allahabad, India, pp. 55-60.

[9] Saxena N., Chaudhari N.S. (2012), Message security in wireless networks: infrastructure based vs. infrastructureless networks, IEEE International Students Conference on Engineering and Systems (SCES′12), MNNIT Allahabad, India, pp. 915-920.

(ii) Published as Abstracts

[1] Saxena N., Chaudhari N.S. (2013), An integrated approach for SMS based secure mobile banking in India, 29th Annual Computer Security Applications Conference (ACSAC′13), New Orleans, USA. (http://www.acsac.org/2013/program/wips/Saxena.pdf)

[2] Saxena N., Chaudhari N.S. (2013), First symmetric batch verification protocol for securely deliver value added services to multiple mobile users, 29th Annual Computer Security Applications Conference (ACSAC′13), New Orleans, USA. (https://www.acsac.org/2013/program/posters/Saxena.pdf)

[3] Saxena N., Chaudhari N.S. (2013), Security in mobile banking, Mobile Telephony in Developing World Conference, University of Jyvaskyla, Finland. (http://mobiletelephony-developingworld.blogspot.in/p/abstracts.html)

[4] Saxena N., Chaudhari N.S. (2013), Mobile banking through secure SMS in India, 9th Global TCS Technical Architects′ Conference (TACTiCS), TCS Siruseri, Chennai, India.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbrevations

AKA	Authentication and Key Agreement
AS	Authentication Server
AuC	Authentication Center
AV	Authentication Vector
BSC	Base Station Controller
BTS	Base Transceiver Station
DoS	Denial of Service
EPS	Evolved Packet System
GSM	Global System for Mobile Communication
HLR	Home Location Register
HSS	Home Subscriber Server
IMSI	International Mobile Subscriber Identity
LTE	Long Term Evolution
MITM	Man-in-the-middle
MME	Mobile Management Entity
MS	Mobile User/ Mobile Station
MSC	Mobile Switching Center
SK	Secret Key
SMS	Short Message Services
TMSI	Temporary Mobile Subscriber Identity
UE	User Equipment
UMTS	Universal Mobile Telecommunications System
VLR	Visitor Location Register
API	Application Programming Interface
PKI	Public Key Infrastructure

Chapter 1

Introduction

Current AKA protocols for cellular networks are used to authenticate users. The existing security algorithms provide privacy to the transmitted information over the network. However, these protocols have various security and performance limitations. The security algorithms used with these protocols have been proved vulnerable or provide weaker encryption. We propose secure and efficient AKA protocols for 2G GSM, 3G UMTS, 4G LTE networks, which improve the limitations of original authentication protocols.

1.1 Motivation and Scope

The security measures of first-generation (1G) were taken into account during the design of 2G digital cellular system, i.e., GSM. In particular, the GSM system was implemented in more than 70 countries around the world till 2002. The existing security issues and limitations of GSM-AKA protocol are overcome by 3G UMTS-AKA protocol, which supports mutual authentication between the network and the mobile user. Unfortunately, the UMTS-AKA protocol also suffers from various security and performance issues including the transmission of actual cipher and integrity keys (CK and IK) over the network, possibilities of various attacks, and generation of huge communication and computation overheads. Further, the 4G LTE network has been developed in order to provide high data speed and to connect and enable communication among various heterogeneous networks. But, the identity protection of mobile user and the clear text transmission of key set identifier are not covered by 4G LTE Evolved Packet System Authentication and Key Agreement (EPS-AKA) protocol. Thus, all these existing loopholes motivated us to develop secure and efficient AKA protocols for 2G GSM, 3G UMTS, and 4G LTE networks.

Although we are running towards 4G/5G technology, the very basic cellular network technology, i.e., GSM, is still not completely secure (GSM protocol is vulnerable to various attacks, does not provide mutual authentication, and GSM security algorithms have been proved vulnerable), which allows an attacker to perform denial of service and session hijacking (impersonation attack) over the GSM network. GSM/UMTS/LTE chipset developers never released any hardware documentation to understand their core functionalities. These cellular network companies buy the operating system kernel and the protocol stack from third parties. Only few people working as operator have knowledge of these protocols. In fact, no one has detailed technical knowledge outside the protocol stack. There was no open source protocol implementation available, which could provide the platform for more people to learn about the protocols and enable quick prototyping/testing by modifying existing code in order to make it more resistance. Nowadays, the open source implementation is available for the GSM network. However, there is no such implementation available for UMTS and LTE networks. We found that many cellular operators do not provide encryption for GSM networks, and hence, making it difficult to know whether the information transmitted over the network is secure or not.

Additionally, two mobile applications of the cellular networks that are focused in this thesis are as follows: (i) secure delivery of value added services (VAS) using SMS, and (ii) end-to-end secure transmission of SMS for mobile users. In both the problems, an authentication server (AS) is involved to authenticate and verify whether the requesting mobile station or mobile user (MS) is valid or not. Hence, the extension of both the problems are also addressed, where the AS receives multiple authentication requests simultaneously or within a very short time period and verifies all MS in a batch. The challenges of developing secure AKA protocols with efficient and reliable AS encouraged us to analyze and develop batch verification-based AKA protocols for both problems.

1.2 Objectives

The main objectives of the thesis are as follows:

1. To propose and analyze a secure and efficient AKA protocol for
   - ▷ 2G GSM network,
   - ▷ 3G UMTS network,
   - ▷ 4G LTE network,
   - ▷ secure delivery of value added services using SMS, and
   - ▷ end-to-end secure transmission of SMS for mobile users.
2. To propose and analyze a secure and efficient batch verification-based AKA protocol for
   - ▷ secure delivery of value added services using SMS, and
   - ▷ end-to-end secure transmission of SMS for mobile users.

1.3 Key Contributions of the Thesis

This section presents the major contributions of this research:

Chapter 3 proposes and analyzes a new and secure AKA protocol named SAKA (Secure Authentication and Key Agreement) for the GSM network. The highlights of the proposed SAKA protocol are as follows:

1. The SAKA protocol
   - ▷ eliminates various security and performance issues exist in the GSM network,
   - ▷ solves the need of synchronization between the MS and its Home Location Register (HLR),
   - ▷ generates less communication overhead as compared to all other existing and proposed GSM protocols, and
   - ▷ prevents the GSM network from replay attack, redirection attack, impersonation attack, and man-in-the-middle (MITM) attack.
2. On an average, it reduces 56% of the bandwidth consumption during the authentication in comparison to the original GSM authentication protocol, which is the maximum reduction of the bandwidth by any GSM protocol from the literature.

Further, the new and secure algorithms namely NewA3, NewA8, and NewA5 are proposed and implemented with respect to existing A3, A8, and A5 algorithms in the GSM network. The NewA5 algorithm is based on block cipher and its variants with cipher feedback, counter, and output feedback mode are proposed to convert block cipher to stream cipher. These new algorithms are used with the proposed SAKA protocol in the GSM network. Hence, the proposed GSM architecture is secure from partition attack, narrow pipe attack, collision attack, interleaving attack, and man-in-the-middle attack. Since, counter mode provides parallelism, therefore, the NewA5 algorithm with counter mode is chosen to encrypt and decrypt the message over the network.

Chapter 4 proposes two AKA protocols for 3G UMTS network: the ES-AKA protocol and the Secure-AKA protocol. Following points describe our significant technical contribution of the ES-AKA protocol:

1. The ES-AKA protocol

   - ▷ provides mutual authentication between the MS and the HLR, and between the MS and the Visitor Location Register (VLR),
   - ▷ guarantees the freshness of the keys used in the protocol,
   - ▷ is able to reduce the bandwidth consumption between the VLR and the HLR, reduces VLR storage, and solves the synchronization problem that exists in the UMTS-AKA protocol,
   - ▷ produces lesser communication overhead as compared to all existing AKA protocols from the literature, and
   - ▷ generates less computation overhead than UMTS-AKA, EXT-AKA, S-AKA, and COCKTAIL-AKA protocols.

2. It prevents the UMTS network from redirection attack, man-in-the-middle attack, replay attack, DoS attack, and active attacks in network corruption. It is found that no protocol other than S-AKA and ES-AKA provide the prevention from DoS attack. Further, it is observed that the S-AKA provides partial prevention, whereas the ES-AKA is free from flood of authentication requests-based DoS attack.

3. On an average, it reduces 62% of the bandwidth consumption during the

authentication process, which is the maximum reduction of the bandwidth by any UMTS protocol from the literature.

4. On an average, it reduces 6% message exchanged ratio (in terms of computations) during the authentication in comparison to the original UMTS-AKA protocol.

5. A modified AES algorithm namely MAES-128 is proposed to implement along with the ES-AKA protocol in the UMTS network.

Further, the Secure-AKA protocol (especially proposed to prevent DoS attack) for 3G UMTS network contains following main features:

1. The Secure-AKA protocol

   ▷ provides mutual authentication between the MS and the HLR, and between the MS and the VLR in the UMTS network,

   ▷ prevents the UMTS network from redirection and black-hole attack, man-in-the-middle attack, replay attack, active attacks in the corrupted network, denial of service attack, and dropping acknowledgement (ACK) signal,

   ▷ is able to reduce the bandwidth consumption between the VLR and the HLR, and reduces the VLR storage,

   ▷ solves the synchronization problem of UMTS-AKA as the mobile user and the roaming network node do not require to maintain any counter (this is possible with message authentication code (MAC_3) and DK key in the proposed protocol),

   ▷ hides the actual identity of each MS, *i.e.*, International Mobile Subscriber Identity (IMSI), and computes a Temporary Mobile Subscriber Identity (TMSI) during the authentication process. The other existing protocols do not provide identity protection over the network, and

   ▷ produces lesser communication and computation overheads as compared to all existing and recent AKA protocols from the literature.

2. On an average, it reduces 65%, 64%, 66%, 51%, 22%, 22%, and 22% of the bandwidth consumption in comparison to UMTS-AKA, AP-AKA,

EURASIP-AKA, S-AKA, COCKTAIL-AKA, X-AKA, and EXT-AKA respectively. This 65% reduction in bandwidth consumption by Secure-AKA is the maximum reduction of the bandwidth by any UMTS protocol.

3. On an average, it is able to lower the ratio of the messages exchanged in terms of computations by 59%, 48%, 54%, 55%, 54%, 48%, and 59% during the authentication with respect to UMTS-AKA, AP-AKA, EURASIP-AKA, S-AKA, COCKTAIL-AKA, X-AKA, and EXT-AKA respectively.

Chapter 5 proposes an improved AKA protocol for 4G LTE network. The contribution of protocol in terms of addressing various security issues are as follows:

1. The proposed protocol
   - ▷ completely hides the actual identity of user, i.e., IMSI, during the authentication over the network,
   - ▷ avoids the key set identifier of access security management entity (KSI$_{ASME}$) to be transmitted over the network,
   - ▷ is able to overcome user-id theft attack, man-in-the-middle attack, and key-id theft attack over the LTE network, and
   - ▷ solves the synchronization problem that occurred in the original EPS-AKA protocol.
2. It reduces 6.1%, 11.8%, 11.7%, and 13.4% of the bandwidth consumption during the authentication between the Mobility Management Entity (MME) and the Home Subscriber Server (HSS) considering single authentication vector (AV) as compared to EPS-AKA, Kϕien′s AKA, Purkhiabani′s AKA, and Choudhury′s AKA protocols respectively.
3. It lowers 6%, 18.5%, and 12.9% of the communication overhead in comparison to Kϕien′s AKA, Purkhiabani′s AKA, and Choudhury′s AKA protocols respectively. In other words, the proposed protocol is able to solve the security issues without increasing the bandwidth requirement and communication overhead.

Chapter 6 proposes and analyzes a new, secure and efficient protocol called SecureSMS for the secure delivery of value added services to the mobile users. The highlights of SecureSMS protocol are as follows:

1. The SecureSMS protocol

    ▷ generates lesser communication and computation overheads,

    ▷ provides the justified prevention against SMS disclosure, SMS spoofing, replay attack, man-in-the-middle attack, over the air (OTA) modification in SMS transmission, security issue in SMS transmission through Signaling System Number-7 (SS7), signer and content verification attack, source substitution attack, time-memory tradeoff attack, codebook attack, key separation attack, and known key attack, and

    ▷ proposes a scheme to store and implement the cryptographic algorithms onto the SIM card.

2. On an average, it reduces 71% and 59% of the total bandwidth used during the authentication as compared to SMSSec and PK-SIM respectively.

3. It provides end-to-end SMS security with authentication (by the SecureSMS protocol), confidentiality (by encryption with AES/ Blowfish, preferred AES-counter), integrity (SHA1/ MD5, preferred SHA1) and non-repudiation (ECDSA/ DSA, preferred ECDSA).

Chapter 7 proposes an efficient batch verification-based protocol namely EXTVAS-AKA, which enables the verification of multiple authentication requests at one time for providing the secure value added services to multiple mobile users. The highlights of the proposed protocol are as follows:

1. The EXTVAS-AKA protocol

    ▷ handles multiple authentication requests at one time or in a fixed (very less) time duration,

    ▷ provides mutual authentication between each MS and the AS,

    ▷ protects the original identity of the MS during the transmission of information over the network, and

   - ▷ maintains integrity in the communication medium between the MS and the AS using MAC function to make it secure enough for transmitting the information.

2. In the authentication requests from the device to the server, the EXTVAS-AKA protocol reduces 14.29%, 50.69%, and 56.89% of the bandwidth consumption as compared to ABAKA, BLS, and ECDSA-AKA protocols respectively.

3. From the server to the device authentication requests, the EXTVAS-AKA lowers the communication bandwidth by 70% and 85.63% in comparison to ABAKA and ECDSA-AKA protocols.

Chapter 8 proposes EasySMS Protocol for end-to-end SMS security over the network. The proposed protocol is having following features:

1. The EasySMS protocol

   - ▷ provides mutual authentication between the MS and the AS,
   - ▷ prevents the SMS information from various attacks including SMS disclosure, over the air modification, replay attack, man-in-the-middle attack, and impersonation attack, and
   - ▷ transmits lesser number of transmitted bits and generates less computation overhead as compared to SMSSec and PK-SIM protocols.

2. On an average, the EasySMS protocol reduces 51% and 31% of the bandwidth consumption and reduces 62% and 45% of the message exchanged ratio during the authentication process as compared to SMSSec and PK-SIM protocols respectively.

3. To the best of our knowledge, this is the first protocol, which is completely based on symmetric key cryptography and provides end-to-end SMS security to the mobile users.

Chapter 9 proposes an efficient batch oriented authentication and key agreement protocol namely BOPSMS, which provides end-to-end message security during the communication between the mobile users over the network. The advantages of the BOPSMS protocol are as follows:

1. The BOPSMS protocol

    ▷ securely transmits the SMS messages from one MS to multiple MS simultaneously,

    ▷ provides mutual authentication between the AS and the MS (sender), and the AS and all recipient MS, and

    ▷ prevents the SMS message from various attacks like SMS disclosure, SMS spoofing, replay attack, man-in-the-middle attack, OTA protection, and security protection over the SS7 channel.

2. The privacy of each MS (sender as well as all recipients) is well protected during the authentication process.

3. From the device to the server batch authentication, the proposed approach for the AS reduces 23.81%, 56.17%, and 61.68% of the transmission bandwidth as compared to ABAKA, BLS, and ECDSA-AKA protocols respectively, while from the server to the device, the proposed AS scheme lowers the communication bandwidth by 80% and 90.5% in comparison to ABAKA and ECDSA-AKA protocols respectively.

4. To the best of our knowledge, this is the first batch oriented AKA protocol, which provides end-to-end security to the SMS, completely based on symmetric key cryptography.

1.4 Organization of the Thesis

The entire thesis is organized in 10 chapters. Next chapter discusses the literature review of various AKA protocols for GSM, UMTS, and LTE networks, for secure delivery of value added service, and for end-to-end SMS security.

Foremost, chapter 3 briefly describes the original GSM architecture, GSM-AKA protocol, and existing encryption schemes of GSM network with their weaknesses. The chapter proposes a new SAKA protocol for the GSM network. The security and performance analysis of the proposed protocol is presented and the behavior of protocol is discussed under an attack model. Further, new and secure algorithms for the GSM network are proposed, analyzed, and implemented. The security of these algorithms is evaluated in terms of cryptanalysis, brute force analysis, and operational analysis along with the justification against various attacks. A brief analysis of attacks on real GSM phone with open source software: Open Source Mobile Communications Base Band (OsmocomBB), OpenBTS, and OpenBSC, is also discussed that focuses on possibilities of various attacks in the current cellular network with some hands on experience.

Chapter 4 reviews original UMTS AKA protocol and its limitations. In order to overcome various limitations of UMTS-AKA, a secure and efficient ES-AKA protcol with an improved MAES-128 algorithm is proposed. The chapter, further discusses the security and performance analysis of ES-AKA under the communication, trust, and attack models. Next, various scenarios with black-hole attack, dropping ACK signal issue, and DoS attack are presented with an assumption that the adversary is more powerful than a normal user in terms of resource consumption. In order to resolve these loopholes, a tour puzzle-based solution namely Secure-AKA protocol is proposed with the discussion of security requirements, drawbacks in the existing scheme, and improvements to the existing scheme. The security analysis, performance evaluation, and simulation of both the protocols are presented.

Chapter 5 proposes a secure and efficient AKA protocol for 4G LTE network. This protocol improves limitations of the original EPS-AKA protocol in the LTE network. The security and performance evaluation of the proposed protocol are discussed along with implementation results. Finally, a formal security proof of proposed AKA Protocol is derived in order to justify the security of the authentication protocol.

Chapter 6 presents an efficient SecureSMS protocol, which provides the secure delivery of VAS to the mobile user. The basic security requirements and system model are considered for the effective design of the proposed protocol. Its security check and performance efficiency is verified under an attack model. Further, the chapter discusses the implementation aspects and reports simulation results of the SecureSMS protocol with its formal proof.

A batch verification-based EXTVAS-AKA protocol for the secure delivery of VAS to multiple mobile users is presented in chapter 7. The protocol proves its security and efficiency under an attack model with the consideration of a system and communication model for the required service. The reliability analysis of the proposed scheme and an algorithm to detect malicious request(s) in a batch, is discussed. Further, the simulation results of the protocol in terms of execution time, verification delay, and re-batch verification delay are observed and reported.

The EasySMS protocol for end-to-end secure transmission of SMS in the cellular networks is proposed in chapter 8. The security of the protocol is analyzed under an attack model. The performance evaluation of EasySMS protocol is presented and reports with respect to overheads and bandwidth utilization. Further, the implementation results of a variant of AES algorithm called MAES are discussed. The proposed algorithm is compared with other existing algorithms in terms of encryption and decryption time. Next, the confidential interval for different symmetric key algorithms are calculated in order to determine the algorithm with most strict output.

Chapter 9 proposes a BOPSMS protocol for SMS security in a batch mode under the system, communication and attack models. The security and performance analysis are discussed in terms of resistance to various attacks, communication and computation overheads, and batch and re-batch verification delay. The simulation results of BOPSMS are reported and formal proof of BOPSMS protocol is presented.

Finally, last chapter summarizes the conclusions of this work with discussion, and gives future research directions.

Chapter 2

Literature Work

This section reviews various authentication protocols, which have been proposed by various researchers for 2G GSM, 3G UMTS and 4G LTE networks, for delivering value added services to mobile users, and for secure transmission of SMS between the mobile users.

2.1 2G GSM Network

Various authentication protocols have been proposed to improve the existing GSM authentication protocol. However, most of them could not solve all the drawbacks of GSM-AKA, mentioned in chapter 1. Some of them even proposed to change the original architecture of GSM.

The challenge-response mechanism present in GSM-AKA protocol is a simple authentication mechanism and is insecure [1], [2], [3]. Most of the mobile operators implement a single COMP-128 keyed hash function to generate SRES and KC, instead of using A3 and A8 algorithms separately. In 1997, a leaked document led to publication of COMP-128, which was used in place of A3/A8 [4]. In 1998-99, Briceno, Goldberg, and Wagner published an attack on COMP-128, with which it is possible to find out the secret key K_i [5]. A partition attack was launched on COMP-128 by Rao J. R. et al. [6]. The cipher algorithms used in the GSM network, *i.e.*, A5/1 [7], [8] and A5/2 [9], have already been proved vulnerable. The latest attacks on A5/1 also suggest that this scheme should be replaced [10].

Harn and Lin′s approach [2] eliminates the stored information in the VLR and stores only the one-way result. However, the overhead occurred in each session for the computation of signed response (SRES) and secret key (KC) by each subscriber is huge. Al-Tawil K. et al. [11] were interested in reducing the authentication delay and the network signaling overhead by reducing the number of messages, which leads to decreasing the call setup time without compromising the GSM security. In the year of 1999, Lo and Chen [12], [13] proposed secure communication protocols for the GSM network, which were more secure than the existing GSM protocol. However, the proposed architecture of GSM was changed and based on public-key cryptography. Unfortunately this protocol could not solve the drawbacks of GSM-AKA. Afterward, Lee et al. [1] proposed an enhanced privacy and authentication method for the GSM network that could solve some drawbacks. The parameters are transmitted from the HLR to the VLR in an encrypted mode that protects the signaling data from eavesdropping.

Further, some old proposed protocols based on the GSM architecture are reviewed. Lee et al. [14] proposed a mutual authentication technique for the GSM network. The main idea is that the HLR issues a ticket for the new VLR where first time the MS is authenticated in a new location. Detailed analysis shows that this technique only works correctly for the first MS authentication, and cannot subsequently verify the network [15]. Chang et al. [15] developed a modified version of the approach presented in [14], which overcomes this security flaw. After the MS joins a new visiting area, a timestamp is added to the initial message in the MS request. Afterwards, the HLR generates a certificate for the VLR to authenticate the VLR by the MS. The secret key used for authentication must be of at least 64 bits because an adversary can easily compromise 32-bit key by listening to the wireless channel and by brute force attack [16].

Now, a brief look of recent technological advancement in the GSM architecture and its networks are presented. Fanian A. et al. [17] proposed a novel mutual entity authentication using the TESLA protocol in 2009. However, using the TESLA protocol in the proposed mutual authentication protocol does increase memory usage and processing time at MS. Further, they [18] proposed a new approach to mutual entity authentication based on symmetric polynomials in 2010. Yubo S. et al. proposed an idea in 2011 about catching the phone number over the air by man-in-the-middle attack [19]. Unlike other terminal detectors or interceptors, which only can catch the IMSI of the phone number, this concept do the phone number catching by a man in-the-middle attack between the victim′s mobile station and the operator′s GSM network. Southern E. et al. [20] proposes simple and effective solutions in 2011 to reduce the possible attacks on the UMTS systems due to the integration with GSM. Further, Firoozjaei M. D. et al. introduced a new location depended key generation management mechanism in 2012 and evaluated its applicability in the GSM network [21]. They have used MS′s location information to generate the key by geo-encryption algorithm. Unfortunately, all these recent protocols are proposed with change in the original GSM architecture.

2.2 3G UMTS Network

In the UMTS-AKA protocol, each MS shares a secret key SK and certain cryptographic functions with the home network. The HLR and the MS, each maintains a counter to prevent replay attack [22]. Various AKA protocols [23], [24], [25], [26], [27] were proposed to ensure the mutual authentication between the communicating parties in the UMTS network. Further, many other symmetric key-based protocols [28], [29], [30], [31], and [32] were proposed in order to improve the security of the UMTS-AKA protocol and proper utilization of the bandwidth during the authentication process. However, the UMTS-AKA protocol does not prevent man-in-the-middle and redirection attacks. Zhang and Fang [30] proposed a new protocol named AP-AKA to debacle the redirection attack and intensely down the impact of attacks in corrupted networks, however, the protocol is under MITM attack. The X-AKA protocol [31] limits the transmission of AVs in the UMTS network and effectively utilizes the bandwidth. But, the X-AKA protocol is suffered from redirection and man-in-the-middle attacks. Further, Al-saraireh et al. EURASIP-AKA [28] and EXT-AKA [33] protocols primarily emphasis on the bandwidth reduction for transmitting the authentication vectors. However, these protocols do not make clear the security prevention against redirection as well as man-in-the-middle attacks. Ou et al. [29] proposed a new COCKTAIL-AKA protocol, to vanquish the issues of the UMTS-AKA protocol. But unfortunately, it is penetrable to DoS and impersonation attacks [34]. It does not solve the synchronization problem that exists between the MS and the HLR. The S-AKA protocol defeats both, redirection and man-in-the-middle attacks [35], and reduces the bandwidth consumption upto 38% (with number of authentication requests n = 2, 5, 10, 20, 50, and 100). However, our analysis states that S-AKA can reduce bandwidth consumption upto 29% (with n = 50, 100, 200, 500, and 1000). The S-AKA protocol partially prevents DoS attack, since the forged message in phase-2 is detected by the Serving GPRS Support Node (SGSN) but, the forged message in phase-1 is only identified by home network.

2.3 4G LTE Network

Prevention from various threats and attacks is a major concern in the security of 4G LTE network [36], [37]. Hence, it is highly recommended to continuously explore and identify different issues and challenges [38], [39], and improve the protocols used in 4G networks [40]. The main focus in 4G LTE network is the security aspects of AKA protocol. In the last 5 years, both, the symmetric key [41], [42], [43] as well as asymmetric key [44], [45] based AKA protocols for the LTE network have been proposed by many researchers.

Different authentication protocols [41], [42], [43], [44], [45], [46], [47] for 4G LTE network have been proposed and out of these protocols only [47] provides the user protection over the LTE network. However, this protocol generates huge communication overhead and does not discuss about the prevention from various possible attacks. Additionally, the protocol suffers from the problem of synchronization, similar to original EPS-AKA. None of the above mentioned protocols is able to protect KSI$_{ASME}$ over the LTE network.

2.4 Secure Delivery of VAS using SMS

In order to provide security to the SMS, several SMS-based frameworks and protocols have been proposed like SMSSec protocol [48], PK-SIM protocol [49], Marko′s SMS framework [50], Songyang′s SMS framework [51], Alfredo and Aniello′s SEESMS framework [52], [53], SSMS protocol [54] and, Marko and Konstantin′s protocol [55]. But, there are many issues with the justification of these protocols in terms of security analysis, communication and computation overheads, prevention from various threats and attacks, and bandwidth utilization. Only SMSSec protocol [48] and PK-SIM protocol [49] justify themselves up to a little extend. They discuss the nature of some basic attacks like SMS disclosure, SMS spoofing, and replay attack but do not explain the resistance mechanisms against various possible attacks. The SMSSec protocol proposes a technique for fault tolerance, however, it does not explain the impact of various attacks on this protocol. The SMSSec is based on symmetric as well as asymmetric cryptography, which results in higher cost as compared to the symmetric cryptography-based protocol. The PK-SIM protocol is also doubtful from the security point of view as there is no discussion in the paper about the resistance to various attacks over the protocol. Apart from this, some other protocols are discussed in [56], but, they do not fit to be considered as an authentication protocol for providing end-to-end security to the SMS. Moreover, these protocols are based on the public key cryptography, which is not suitable for the SMS security because of its higher implementation cost. Since, the mobile phones are infected with various security measures including mobile phones malware, identity theft, and phishing with SMS [57], thus, developing a protocol is a good choice in comparison to framework deployment.

2.5 Batch Oriented Secure Delivery of VAS using SMS

Many security solutions have been proposed by various researchers for value added services in the vehicular ad hoc networks [58], [59], [60], [61], [62], and social networks [63]. However, in the literature, with the best of our knowledge there is no such batch verification oriented protocol, providing value added services to the mobile users. Thus, it is strongly recommended to implement such application oriented protocols to facilitate the mobile users with different value added services, where a server receives authentication requests by several mobile phones at one time.

2.6 End-to-End SMS Security for Mobile Users

Various researchers have proposed different techniques to provide security to the transmitted message. An implementation of a public key cryptosystem for the SMS in a mobile phone network has been presented in [50], but, the security analysis of the protocol has not been discussed. A secure SMS is considered to provide mobile commerce services and is based on public key infrastructure [51]. Further, a Secure Extensible and Efficient SMS framework (SEESMS) is presented by Santis A. et al., which allows two peers to exchange encrypted communication between the peers by using public key cryptography [52]. Another new application layer framework called SSMS is introduced by Toorani M. et al., to efficiently embed the desired security attributes in the SMS to be used as a secure bearer for m-payment systems. This solution is based on the elliptic curve-based public key that uses public keys for the secret key establishment [54]. The protocols in [51] and [54] generate shared key for each session but also generate huge overheads and not suitable for the real world applications. An efficient framework for automated acquisition and storage of medical data using the SMS-based infrastructure is presented in [64]. In all [50], [51], [52], [54] and [64], it is not clear whether the proposed approaches are able to prevent the SMS against various possible threats and attacks.

All the above mentioned approaches/protocols/frameworks generate a large overhead as they propose an additional framework for the security of SMS. Due to physical limitations of the mobile phones, it is recommended to develop a protocol, which would make minimum use of computing resources and would provide better security. The implementation of framework always increases the overall overhead, which is not suitable for the resource constraints devices such as mobile phones. Thus, it is recommended to compare a new designed protocol with the existing SMSSec and PK-SIM protocols. The reason for chosen these protocols for comparison is that these are the only existing protocols, which do not propose to change the existing architecture of cellular networks. However, there is no protocol available for end-to-end SMS security with symmetric key cryptography. The SMSSec protocol is used to secure an SMS communication sent by Java′s Wireless Messaging API (WMA), while the PK-SIM protocol proposes a standard SIM card with additional PKI functionality. Both protocols are based on client-server paradigm, *i.e.*, one side is mobile user and the other side is authentication server, but, they do not present any scenario, where an SMS is sent from one mobile user to another mobile user. The SMSSec protocol does not discuss the security analysis.

2.7 Batch Oriented End-to-End SMS Security for Mobile Users

Various batch verification-based solutions have been proposed for the value added services in vehicular ad hoc networks [58], [59], [60], [62], [65], public-private key-based vehicular communication system [66], and digital signature for batch [67], [68] to achieve high efficiency. Some SMS-based wireless protocols [48], [49], [69], and SMS-based attacks and their countermeasures [65], [70] are discussed by various researchers, but they do not consider simultaneous transmission of multiple authentication requests in their work. In the literature, we did not find any batch verification protocol, providing end-to-end secure transmission of SMS to the mobile users. As a solution to this problem, there are various application software like SMSzipper, TextSecure, moGile Secure SMS, CryptoSMS, available in the market, which provide the facility to send secure SMS. However, there are certain limitations with these software such as (i) we need to install them in the phone memory/memory card, (ii) we require to provide a secret key explicitly to the recipient of the SMS, and (iii) they do not work, when we send the SMS to many people simultaneously.

2.8 Chapter Summary

An overview of the background of authentication protocols for the cellular networks and for the SMS-based applications has been presented. In the literature, we have also covered the main approaches proposed by many researchers to tackle various security issues and challenges of the cellular networks and highlighted the strengths and weaknesses of each one.

Chapter 3

AKA Protocol in 2G GSM Network

3.1 Introduction

Weaknesses of the GSM challenge response scheme have been uncovered over time, where the authentication is only unidirectional and the subscriber cannot authenticate the serving network. Several drawbacks related to security and performance issues of the original GSM authentication protocol are as follows:

1. Mutual authentication between the MS and the VLR is not provided in the original GSM protocol. Only VLR authenticates the MS, but the VLR is not authenticated by the MS.

2. When the MS is in foreign network, the n-copies of authentication tokens are transmitted from the HLR to the VLR, which are normally stored in the VLR′s database. This approach generates a huge storage space overhead.

3. If any MS resides in a VLR for long time duration and consumes all the available authenticating tokens, the VLR will again request the HLR to send n-copies of next authentication tokens. Additionally, it may be the case for the MS to move frequently from one foreign network to another in a short span of time. In such cases, each VLR will request the HLR to send n-authentication tokens, which results in huge bandwidth consumption between the VLR and the HLR.

4. The HLR is also overloaded with the authentication tokens of mobile users because all triplets are generated at HLR.

5. The synchronization problem exists between the MS and its HLR.

6. The GSM network is affected by various security attacks like replay attack, redirection attack, impersonation attack, and man-in-the-middle attack.

The encryption algorithm A5/1 was normally used in western countries and was considered strong encryption, but few years back, it was reverse engineered and proved vulnerable. Further, the A5/2 has been cracked by Wagner and Goldberg making A5/2 almost useless. This proves that there is always a possibility of eavesdropping. Thus, it is required to investigate these algorithms and their performance results under a secure GSM architecture.

3.2 GSM Architecture

The GSM system can be divided into three subsystems: Base Station Subsystem (BSS), Network and Switching Subsystem (NSS), and Operational Subsystem (OSS). Figure 3.1 illustrates an overview of GSM architecture. Here, SMS-GMSC represents the SMS Gateway Mobile Switching Center. The description of each subsystem of the GSM network is as follows:

1. *Base Station Subsystem (BSS):* The functions related to radios are executed in the BSS. Basically, a BSS is divided into 2 parts: a) Base Transceiver Station (BTS) and b) Base Station Controller (BSC).

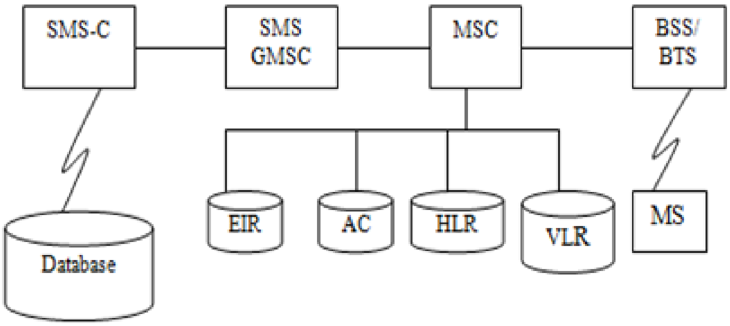


Figure 3.1: Overview of GSM architecture

a) Base Transceiver Station (BTS): The BTS is responsible for the radio interface to the MS. It is required by wireless network to facilitate each cell

Table 3.1: Symbols and abbreviations

Symbol	Definition	Size (Bits)
IMSI	International Mobile Subscriber Identity	128
SQN	Sequence Number	48
LAI	Location Area Identity	40
User Profile	User Mobile Number/Reference Number	40
Service Request	Location Update Request/Call Attempt	8
VLR_ID	Identification of VLR	28
Count	Integer Number	24
RAND	Random Number	128
VLR_C	Certificate of VLR	64
AV	Authentication Vector	Variable
K_i/SK/TK	Secret key shared b/w MS and HLR	128
KC	Temporary session key	64
DK	Delegation key	128
MAC/XMAC	Message Authentication Code	64
SRES/AUTR/ AUTHR	Signed Response	64
T	Timestamp	64

Table 3.2: Cryptographic functions definition

Function	Definition
A_3/NewA_3	Authentication algorithm
A_8/NewA_8	Key agreement algorithm
A_5/NewA_5	Specified for data encryption and decryption
||	Concatenation

in the network. A number of BTSs (10-100) are controlled and monitored by a single BSC.

b) Base Station Controller (BSC): The control functions performed in the network and the physical links between the Mobile Switching Center (MSC) and the BTS are provided by BSC. It acts like a high-capacity switch, which serves handover, cell configuration data, and allocation of radio channel. A number of BSCs are managed by a single MSC.

2. *Network Switching Subsystem (NSS):* It is responsible for performing main switching functions in the GSM network. It also performs various functions related to call switching and mobility management. This system mainly includes circuit switch core network in the traditional GSM network, however, it has extended to provide packet switched services in a General Purpose Radio Service (GPRS) network.

a) Mobile Switching Center (MSC): The MSC provides routing to the voice calls and the SMS. It executes the switching functions required for the MS (within MSC) and carries out handover. The MSC is also involved in internetworking to facilitate and communicate with other networks such as the Public Switched Telephone Network (PSTN).

b) Home Location Register (HLR): The HLR is a central database responsible for the mobile phone data storage and its management. It handles the data about the subscribers including the service profile, location information, and activity status. It also stores the information about the SIM cards.

c) Visitor Location Register (VLR): The VLR database is responsible for providing the roaming service to the subscribers. It is connected with one or more MSCs and dynamically accumulates user information, when he/she is in roaming area. When an MS moves to the MSC of that roaming area, the MSC informs to the associated VLR about the movement of MS.

3. *Operational Subsystem (OSS):* It handles various tasks to maintain system security such as validation of user identities and proper execution of the protocol, which are performed in the Authentication Center (AuC) and Equipment Identity Register (EIR). One Operation and Maintenance Center (OMC) can serve for several MSCs and each is responsible to authenticate each SIM card that connects to the GSM network. EIR stores the identity of mobile phones. The stolen mobile phones can be tracked by this information.

Each VLR has a specified area with a unique Location Area Identifier (LAI) code consists of Mobile Country Code (MCC), Mobile Network Code (MNC), and Local Area Code (LAC). Each MSC can have number of subareas as BSC (with a unique LAC). The BSC area consists of several BTS (with a unique cell-id) while each BTS may have several sectors (1-6 sectors with sector-id).

Table 3.1 lists various symbols with their definitions and size that are used in this chapter, while Table 3.2 represents the definition of various functions.

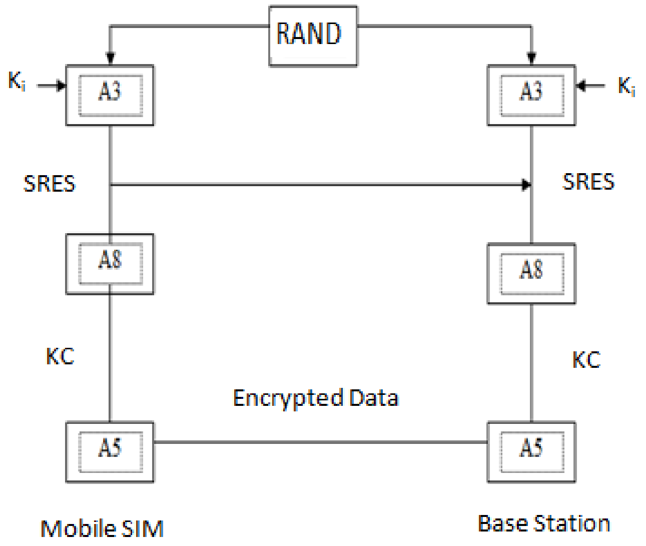


Figure 3.2: GSM authentication and encryption

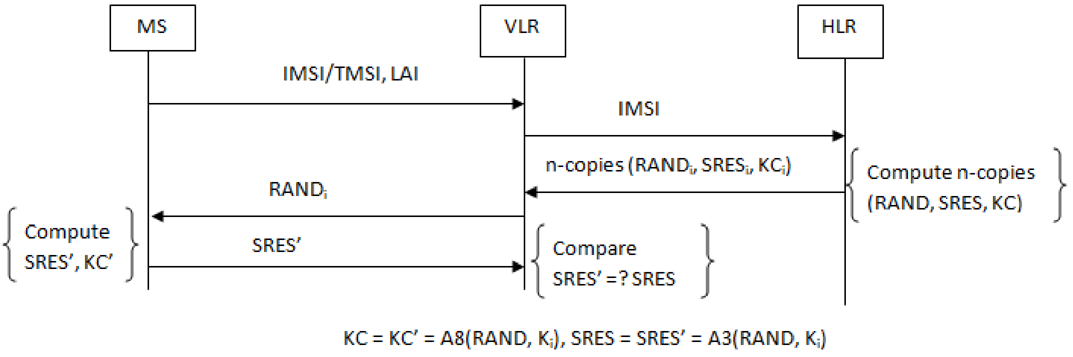

MS
VLR
HLR
IMSI/TMSI, LAI
IMSI
n-copies (RAND$_i$, SRES$_i$, KC$_i$)
Compute n-copies (RAND, SRES, KC)
RAND$_i$
Compute SRES', KC'
SRES'
Compare SRES' =? SRES
KC = KC' = A8(RAND, K$_i$), SRES = SRES' = A3(RAND, K$_i$)

Figure 3.3: GSM authentication protocol

3.3 Existing GSM-AKA Protocol

The existing security protocol of the GSM network consists of three tiers as shown in Figure 3.2. First tier is the A3 algorithm, which uses the challenge-response method for user authentication. Second tier is the A8 algorithm, which uses the output of A3 algorithm to generate the secret key. Last tier is the A5 algorithm, which uses a cipher algorithm for message encryption and decryption. This protocol has some cryptographic problems like challenge-response is not secure.

The authentication of SIM or user depends upon a secret key shared between the SIM and the AuC called K$_i$. This K$_i$ is embedded into the SIM card during manufacturing, and is also securely replicated onto the AuC. When the AuC authenticates a SIM, it generates a random number known as RAND and sends it to the subscriber. Both, the AuC and the SIM feed K_i and RAND values into A3/A8 (COMP-128) and a 32-bit SRES is generated by both parties. If the SIM′ SRES matches with the AuC′ SRES, then the SIM is successfully authenticated. Further, both, the AuC and the SIM calculates a second 54-bit secret key namely KC by feeding K_i and RAND values into A5 algorithm. This key is used to encrypt and decrypt the communication session. Note that the A8 algorithm generates a 64-bit KC, thus, it is obvious that COMP-128 hash generates a much weaker KC of 54-bit. All computations take place at AuC and results are returned to the HLR.

The size of only few parameters are mentioned in the work by Al-saraireh K. et al., thus, all the parameters are considered with global values for all existing and proposed GSM protocols. The user profile is assumed as a user mobile number/reference number, which is a 10 digit number (between 0-9 numbers in each place and is of 40 bits in size). Lo C. C. et al. [13] and Stach J. F. et al. [71] proposed the changed architecture of GSM system and for this reason, these two protocols have not been included while calculating the computation and communication overheads.

The existing GSM-AKA protocol has shown in Figure 3.3. An authentication request is sent to the VLR, when the MS moves into a new visiting network and asks for a new communication service. The authentication request includes a TMSI and LAI. After receiving the request, the new VLR uses the received TMSI to get the IMSI from the old VLR and then sends IMSI to the HLR. Then, the HLR generates n-distinct sets of authenticating parameters $\{\mathrm{SRES}_i$, RAND_i, $\mathrm{KC}_i\}$, when i=1, 2, ...,n, and sends them to the VLR. Next, the VLR sends the selected RAND_i to the MS. Once the MS receives RAND_i from the VLR, it computes $\mathrm{SRES}' = \mathrm{A3}(\mathrm{RAND}_i, \mathrm{K}_i)$ and a temporary session key $\mathrm{KC}' = \mathrm{A8}(\mathrm{RAND}_i, \mathrm{K}_i)$. Thereafter, the SRES′ is sent back to the VLR. Upon receiving SRES′ from the MS, the VLR compares it with the selected SRES kept in its own database. If they are not same, the authentication is failed otherwise, the MS is a legal user.

3.4 GSM-AKA Protocol Algorithms

This section presents the overview of COMP-128 algorithm and cipher algorithms A5/1 and A5/2 that are used in the original GSM network.

3.4.1 COMP-128 Algorithm

In the GSM network, A3/A8 algorithms are usually implemented together as COMP-128 algorithm, which was completely private and is used to generate 32-bit SRES and 64-bit KC [72]. Various parameters of the COMP-128 algorithm are as follows:
RAND[0...15]: the challenge from the base station
key[0...15]: the SIM′s A3/A8 long-term key K_i
A3 Input: 128-bit RAND random challenge, 128-bit K_i secret key
A3 Output: 32-bit SRES signed response
A8 Input: 128-bit RAND random challenge, 128-bit K_i secret key
A8 Output: 64-bit KC cipher key used for A5 algorithm
simoutput[0...11]: Output of SIM, out of which simoutput[0...3] is SRES, and simoutput[4...11] is KC. Note that KC is of 64-bit from 74....127 of COMP-128 output (54-bit) followed by 10 zeros, thus, A5 is keyed with only 54 bits.

3.4.2 A5/1 Algorithm

A5/1 is a stream cipher, which is used in the GSM standard. Several time-memory trade-off attacks against A5/1 have been proposed, some of that break A5/1 in seconds using huge precomputation time and memory [73], [74]. Later, Guneysu T. et al. describe time-memory trade-off techniques that can be used for attacking the popular A5/1 algorithm used in GSM voice encryption [75]. A5/1 stream cipher is a binary linear feedback shift register (LFSR)-based key stream generator. All of the registers used in A5/1 are first zeroed and then a 64-bit secret session key K and 22-bit frame number F are XOR′ed in parallel into the least significant bits of the three registers. Then, all LFSRs are clocked for 100 clock cycles according to majority rule, however, no output is produced. Finally, three LFSRs are clocked according to majority rule to generate 228 bits of key stream sequence [76].

3.4.3 A5/2 Algorithm

A5/1 and A5/2 were reverse-engineered from a GSM handset [5] and published by Briceno et al. [73]. A5/2 is built from four LFSRs of lengths 19, 22, 23, 17 bits denoted by R1, R2, R3, and R4 respectively. Clocking of R1, R2, and R3 are controlled by R4, and R4 is regularly clocked in each clock cycle. The clock control mechanism of A5/1 and A5/2 depend on majority rule. However, input to the clocking control mechanism are given from R4 in case of A5/2, while in A5/1 input are from R1, R2, and R3. In each register, majority of two bits and complementary of a third bit is calculated, and the results of all the majorities and the right most bit from each register are XOR′ed to form the output bit [9].

3.5 Attack Model

This subsection discusses an attack model with various possible attacks in the GSM network like replay attack, impersonation attack, man-in-the-middle attack, and redirection attack.

1. *Replay Attack:* An intruder delays the transmitted packet or later can misuse the captured information by retransmitting the message to the original destination. This attack is applied to gain access to the resources by resending the authentication message.

2. *Impersonation Attack:* In this attack, the adversary eavesdrops voice signaling and data (message) of the target users. Moreover, the adversary captures the authentication information of the authentic user. Then, the adversary sends this information between the legitimate user and the network in order to perform this attack.

3. *Man-in-the-middle Attack:* In the MITM attack, an adversary puts itself in between the target user and a genuine network, and can capture, modify, eavesdrop, and spoof signaling and data exchanged between both the parties.

4. *Compromising Authentication Information in the Network:* The adversary compromises the network security or capture the required authentication information from the corrupted network.

5. *Redirection Attack:* These attacks can be easily launched with fake BTS equipments. A fake BTS equipment is normally located in a bound area that can act as a real BTS and broadcasts the BTS signal over the air to the mobile phones. The adversary may redirect the user traffic to a network, in which the ciphering of data is not provided or is very limited. The redirection attack causes the mischarge billing problem while the user is in his/her home network and charged as visitor at a foreign network on a higher rate.

3.6 Proposed SAKA Protocol for GSM Network

To resolve the security issues associated with GSM authentication protocol, we propose and present a new authentication and key agreement protocol namely SAKA, which prevents the GSM network from redirection attack, replay attack, man-in-the-middle attack, and impersonation attack. This SAKA protocol strictly follows the framework of the original GSM architecture and solves the problem of synchronization between the MS and its HLR. In the SAKA protocol, each MS and its HLR share a secret key K$_i$. The cryptographic algorithms A3, A8, and A5 used in the original GSM protocol are proposed to replace with NewA3, NewA8, and NewA5 algorithms. The NewA3 algorithm is shared between the MS and the HLR, while the NewA8 algorithm is shared between the MS and the VLR. The NewA5 algorithm is replicated on MS, VLR, and HLR. The SAKA protocol is shown in Figure 3.4, which is described as follows:

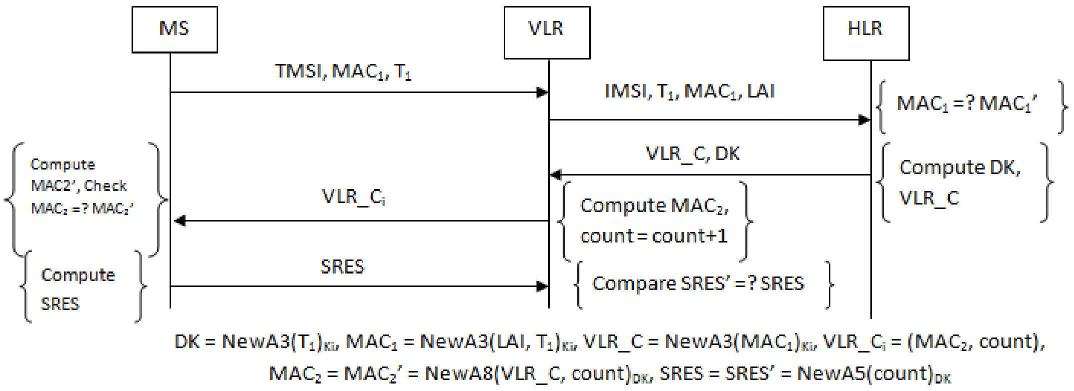

MS
VLR
HLR
TMSI, MAC$_1$, T$_1$
IMSI, T$_1$, MAC$_1$, LAI
MAC$_1$ =? MAC$_1$'
VLR_C, DK
Compute DK, VLR_C
Compute MAC2', Check MAC$_2$ =? MAC$_2$'
VLR_C$_i$
Compute MAC$_2$, count = count+1
SRES
Compute SRES
Compare SRES' =? SRES
DK = NewA3(T$_1$)$_{Ki}$, MAC$_1$ = NewA3(LAI, T$_1$)$_{Ki}$, VLR_C = NewA3(MAC$_1$)$_{Ki}$, VLR_C$_i$ = (MAC$_2$, count),
MAC$_2$ = MAC$_2$' = NewA8(VLR_C, count)$_{DK}$, SRES = SRES' = NewA5(count)$_{DK}$

Figure 3.4: SAKA protocol for the GSM network

In the first step, the MS makes a request to the VLR to create a connection for the authentication by sending TMSI/IMSI, a timestamp T_1, and a computed message authentication code MAC_1 to the VLR, where MAC_1 = NewA3(T_1, LAI)$_{K_i}$ and K_i is the secret key shared between the MS and the HLR.

MS to VLR: TMSI, MAC_1, T_1

If the VLR receives TMSI from the MS then it retrieves the IMSI from the TMSI by communicating with HLR. The VLR passes (IMSI, T_1, MAC_1, LAI) to the HLR, where LAI is the location area identity of the MS. The VLR knows the location of MS and sends this information to the HLR.

VLR to HLR: IMSI, T_1, MAC_1, LAI

After receiving such information, the HLR calculates MAC'_1 = NewA3(T_1, LAI)$_{K_i}$ and compares it with the received MAC_1. If both are equal then the MS is a legitimate user. Thereafter, the HLR generates a delegation key DK and a certificate VLR_C, and passes them to the VLR.

HLR to VLR: VLR_C, DK

The VLR maintains a count, which auto increments after every transmission of VLR_C_i to the MS, consisting of MAC_2 and count. Please note that it does not require synchronization at both ends because count increment is only used to generate a unique number, every time is sent from the VLR to the MS. The MAC_2 and count are updated and calculated every time an authentication is requested by the MS.

VLR to MS: VLR_C_i

After receiving the VLR_C_i, the MS verifies the value of count. If the received count is greater than the previous count, then the MS calculates MAC'_2 and compares it with the received MAC_2, otherwise, the request is discarded and connection is terminated. Then, the MS generates DK and computes SRES. The MS sends this SRES to the VLR.

MS to VLR: SRES

The VLR calculates SRES′ and compares it with the received SRES. If SRES′ and SRES are equal, then authentication is successful, otherwise, the connection is terminated. Afterwards, the message information is encrypted using the NewA5 algorithm and the communication between the MS and the VLR takes place.

3.7 Security and Performance Analysis of SAKA Protocol

This section presents security and performance analysis of the SAKA protocol in terms of mutual authentication, prevention from various attacks, efficiency of protocol with respect to communication, computation and storage overheads, and bandwidth utilization.

3.7.1 Mutual Authentication between the MS and the HLR/ VLR

In the SAKA protocol, the HLR authenticates the MS by verifying MAC_1 and to authenticate the HLR, the MS checks the received VLR_C_i. The MS can acquire the expected authentication code of VLR as MAC'_2 = NewA8(VLR_C, count)$_{DK}$. If MAC'_2 is equal to the MAC_2, then both, the HLR and the VLR are authenticated. This ensures the mutual authentication between the MS and the HLR. For the subsequent authentications, even it may be the case where HLR is not involved, the MS can still authenticate the HLR with the involvement of DK key. Next, the VLR authenticates the MS by verifying the SRES. After receiving message, the VLR calculates SRES′ and checks whether SRES ?= SRES′, where SRES′ = NewA8(count)$_{DK}$. The MS is verified authentic if this equality holds. The same procedure takes place in authenticating the MS, when the VLR receives SRES while communicating with only VLR. Now, on receiving MAC_2, the MS computes MAC'_2 to authenticate the VLR and verifies MAC_2 ?= MAC'_2. All this ensures the mutual authentication between the MS and the HLR/VLR.

3.7.2 Resistance to Attacks

In this subsection, we justify that the SAKA protocol is free from various attacks.

1. *Replay Attack:* The unique random numbers are often used to prevent the system from replayed message attack. Sequence number, timestamp, and the challenge response are three different types of random number or nonce. However, each one of it has its own boundaries to use. The SAKA

protocol is free from this attack by sending timestamp T_1 and count with the information while transmitting the information over the network.

2. *Impersonation Attack:* In the original GSM protocol, the corruption of a network′s VLR/HLR affects the security of whole system. There are two possible cases for the communication between the MS and the VLR/HLR, and each of that is described as follows:
   i) When the adversary tries to impersonate the MS: At the beginning of SAKA protocol, MAC_1 is computed at MS and is sent to the HLR. The HLR then computes MAC'_1 through NewA3 by passing the input as timestamp T_1, and LAI of the MS (notified and sent by the VLR to the HLR) with K_i key. Additionally, if the adversary is able to eavesdrop all the messages that can be sent to other networks then the adversary must reply with a valid response SRES to the VLR in order to impersonate the MS. However, the adversary cannot obtain the correct SRES.
   ii) The adversary tries to impersonate the VLR: If the adversary tries to impersonate the VLR, an attempt to impersonate the VLR will be failed as the MS can verify that the authentication was not requested by the VLR (if MS receives the VLR_C_i at any point of time while previously MS has not sent any request to the VLR). Thus, the impact of network corruption in terms of impersonation is eliminated by the SAKA protocol.

3. *Man-in-the-middle Attack:* A man-in-the-middle attack can occur when the MS tries to connect to a BTS/BSS. In the SAKA protocol, a new algorithm of A5, *i.e.*, NewA5, is conversed between the MS and the VLR. The message content are encrypted using DK key. Thus, this system prevents the communication from being eavesdropped.

4. *Redirection Attack:* In the GSM protocol, the location of BTS/BSS (LAI) is not protected and can be changed by an adversary with the redirection attack. The SAKA protocol uses message authentication codes (MACs) to maintain the integrity of LAI. This concept thereby prevents the network from the redirection attack. This attack is easily possible when the adversary gets the correct user′s MS information. In the SAKA protocol, the MS involves the LAI of BTS/BSS in MAC_1 and transmits MAC_1 to the VLR. The authentication request is discarded, when the HLR fails to

match the LAI sent by the VLR and embedded in MAC_1. Such a technique also solves the mischarged billing problem.

5. *Compromising Authentication Information in the Network:* The adversary cannot compromise the authentication information in the SAKA protocol. The adversary cannot obtain any information because the integrity in terms of MAC is properly maintained in the SAKA protocol.

3.7.3 Performance Analysis

This subsection provides performance analysis of the SAKA protocol in terms of VLR storage overhead, reduction in bandwidth consumption, and efficiency of new system with respect to various parameters.

1. *The VLR storage overhead:* Obviously, the VLR needs to store only DK key and count in its database instead of n-sets of authentication parameters $\{SRES_i, RAND_i, KC_i\}$, where n =1, 2,..., n. Therefore, the SAKA protocol is certainly uses less memory space at VLR as compared to the original GSM protocol.

2. *Bandwidth consumption between the HLR and the VLR:* Instead of generating n-sets of authentication parameters for the VLR to authenticate the MS, the HLR computes a session delegation key DK and sends it to the VLR. Therefore, the VLR can use DK key to compute the signed result/response in order to authenticate the MS. Thus, the SAKA protocol greatly reduces the bandwidth consumption between the HLR and the VLR.

3. *Efficiency of the SAKA protocol:* Since the original GSM authentication protocol needs to generate n-copies of the authentication parameters, the A3 algorithm has to be executed at least n-times at HLR. Thus, the SAKA protocol is more efficient than the GSM authentication protocol. Furthermore, instead of generating a random number for each communication request, the VLR only needs to keep DK key and count, which feed as input to the NewA8 in order to compute SRES.

A comparison among various existing GSM protocols along with SAKA protocol is shown in Table 3.3. The table summarized the same in terms of the

Table 3.3: Comparison of various authentication protocols

Parameters	GSM-AKA	Harn L. et al. [2]	Lee C. H. et al. [1]	Lee C. C. et al. (2005) [15]	Lee C. C. et al.(2003) [14]	Al-tawil K. et al. [11]	SAKA Protocol
Computation Overhead	2nT(A3)	n+ $(\sum_{j=1}^{(n-1)}$j) T(f)	2T(A3)+ 2nT(A5)	(4+2n)T(A3)+ (2+2n)T(A5)	2T(A3)+ 2nT(A5)	2T(A3)+ 2T(A8)+ 2nT(A5)	6T(A3)+ 4nT(A8)
Total Storage (VLR)	nS(SRES)	S(f^{n-j}(SRES))	S(TK$_i$)	S(T$_{i-1}$, TK)	S(TK$_i$)	S(KC,User Profile)	S(DK, count)
Communication Overhead	296+512*n	448+320*n	512+384*n	1082+384*n	708+448*n	888+384*n	784+176*n

Table 3.4: Comparison of GSM authentication protocols vs. requirements

Parameters	GSM-AKA	Lee C. H. et al. [1]	Lee C. C. et al. (2003) [14]	Harn L. et al. [2]	Al-tawil K. et al. [11]	Lo C. C. et al. [13]	Stach J. F. et al. [71]	Lee C. C. et al. (2005) [15]	SAKA Protocol
MAUTH	No	No	Yes	No	No	Yes	Yes	Yes	Yes
AVLR	No	Yes	Yes	No	No	No	No	Yes	Yes
SSVLR	No	Yes	Yes	No	No	No	No	Yes	Yes
SBWC	No	Yes	Yes	Yes	Yes	No	No	Yes	Yes
CGSMA	-	No	No	Yes	Yes	Yes	Yes	No	No

Table 3.5: Prevention analysis from various attacks

Protocol Attacks	Original GSM-AKA	SAKA- Protocol
Replay Attack	Yes	Yes
Man-in-the-middle Attack	No	Yes
Redirection Attack	No	Yes
Impersonation Attack	No	Yes

computation overhead, total storage at VLR, and the communication overhead. The storage space used by SAKA is S(DK, count). The computation overhead generated by SAKA is slightly more but, the communication overhead is drastically reduced in the SAKA protocol in comparison to all other protocols mentioned in Table 3.3. The generated computation overhead by the SAKA protocol is only better than [15] as compared to all existing GSM authentication protocols. Table 3.4 represents the requirements to solve the issues exist in the GSM network with respect to various GSM protocols. Some of these requirements are MAUTH: mutual authentication, SSVLR: solve problem of space overhead at VLR, SBWC: solve problem of bandwidth consumption, CGSMA: change in GSM architecture, and AVLR: authentication of MS by VLR instead of HLR. It is clear that SAKA protocol fulfills all such requirements of the GSM network. Next, Table 3.5 represents the prevention of GSM network from various attacks like replay attack, man-in-the-middle attack, redirection attack, and impersonation attack. The SAKA protocol protects the GSM network from all these attacks successfully. The total bandwidth consumed during the authentication process by original GSM protocol and SAKA protocol are computed in Table 3.6. On an average, the SAKA protocol results 56% reduction in the

Table 3.6: Bandwidth consumption in various GSM authentication protocols

Auth. Token (n)	Harn L. et al. [2]/ GSM-AKA	Lee C. H. et al. [1]/ GSM-AKA	Lee C. C. et al. (2005) [15]/ GSM-AKA	Lee C. C. et al. (2003) [14]/ GSM-AKA	Al-tawil K. et al. [11]/ GSM-AKA	SAKA Protocol/ GSM-AKA
5	0.71	0.85	1.05	1.03	0.98	0.58
10	0.67	0.80	0.90	0.95	0.87	0.46
50	0.63	0.76	0.78	0.89	0.77	0.37
100	0.63	0.75	0.76	0.88	0.76	0.35
Average	0.66	0.79	0.87	0.93	0.84	0.44

bandwidth consumption as compared to the GSM protocol, when number of authentication tokens n = 5, 10, 50, and 100.

3.7.4 Formal Proof of SAKA Protocol

In order to clear statement of our analysis, the BAN-Logic symbols are used to formally proof the authentication process of the proposed protocol. BAN Logic is a way of logical verification of authentication protocol, which formally states the knowledge of information. It formally verifies the information exchanged and the trust among communicating and involved parties at every step in the authentication protocol. (1) P$|\equiv$ X: P believes X. P believes that X is correct and P trusts on its outcome. (2) P $\triangleleft$ X: P sees X. A message X is received by user P and now P can read and understand it. (3)P $|\tilde{}$X : P once said X. At any point of time, user P sends a message including the statement X. (4) P$|\Rightarrow$ X : P has jurisdiction over X. User P has an authority over X statement and P should be treated as a trusted user for this statement. (5) #(X) : The number/equation X is fresh and is being sent for the first time. It has not been sent previously in any message during the current run of the protocol. (6) P $\overset{K}{\leftrightarrow}$ Q : P and Q use the shared key K to communicate. The users or entities, P and Q use K as shared secret key for the secure communication over the network. (7) P $\overset{K}{\Leftrightarrow}$ Q : The formula/variable X is a secret known only to P and Q. The X is not publically available and is known only to P and Q. (8) $(X)_Y$: X is combined with the formula Y that Y be a secret. It represents that X is a formula/equation and Y is a secret key. But this formula X is always used with the secret key Y.

1. *The Formal Messages in SAKA Protocol:*

   (1) MS → VLR: Na, NewA$_3$(Na, LAI)$_{K_i}$, MS $\overset{K_i}{\leftrightarrow}$ HLR

   (2) VLR → HLR: Na, NewA$_3$(Na, LAI)$_{K_i}$

   (3) HLR → VLR: NewA$_3$(Na)$_{K_i}$, NewA$_3$(NewA$_3$(Na, LAI)$_{K_i}$)$_{K_i}$, VLR $\overset{DK}{\leftrightarrow}$ HLR

   (4) VLR → MS: Nb, NewA$_8$(NewA$_3$(NewA$_3$(Na, LAI)$_{K_i}$)$_{K_i}$), Nb)$_{DK}$

   (5) MS → VLR: NewA$_3$(Na)$_{K_i}$, NewA$_8$(Nb)$_{DK}$

2. *Security Assumptions:*

   a) It is assumed that K$_i$ key is shared between the MS and its HLR.

   (1) MS has the secure key K$_i$ and MS $|\equiv$ MS $\overset{K_i}{\leftrightarrow}$ HLR

   (2) HLR has the secure key K$_i$ and HLR $|\equiv$ MS $\overset{K_i}{\leftrightarrow}$ HLR

   b) It is assumed that the VLR trusts the HLR.

   (1) VLR $|\equiv$ HLR $|\Rightarrow$ MS $\overset{K_i}{\leftrightarrow}$ HLR,

   (2) $\frac{VLR \lceil P, HLR \triangleleft P}{HLR|\equiv VLR|\equiv P}$

   (3) $\frac{HLR \lceil P, VLR \triangleleft P}{VLR|\equiv HLR|\equiv P}$

   c) It is assumed that communication between the HLR and the VLR is secure.

   (1) VLR $|\equiv$ VLR $\overset{P}{\Leftrightarrow}$ HLR, P is conveyance message between the VLR and the HLR.

   (2) HLR $|\equiv$ VLR $\overset{P}{\Leftrightarrow}$ HLR, P is conveyance message between the VLR and the HLR.

3. *Security Analysis:*

   (1) MS → VLR: MS$|\equiv$ #(Na) $\wedge$ VLR$|\equiv$ #(Na); VLR $\triangleleft$ Na, NewA$_3$(Na, LAI)$_{K_i}$

   (2) VLR → HLR: HLR $\triangleleft$ Na, LAI, NewA$_3$(Na, LAI)$_{K_i}$, receiving check NewA$_3$(Na, LAI)$_{K_i}$

   (3) HLR → VLR: VLR $\triangleleft$ NewA$_3$(NewA$_3$(Na, LAI)$_{K_i}$))$_{K_i}$, NewA$_3$(Na, LAI)$_{K_i}$, HLR $|\equiv$ $\forall$ (VLR $\overset{DK}{\leftrightarrow}$ MS)

   (4) VLR → MS: MS$|\equiv$ #(Na) $\wedge$ VLR$|\equiv$ #(Nb),

   MS $\triangleleft$ Nb, NewA$_8$(NewA$_3$(NewA$_3$(Na, LAI)$_{K_i}$)$_{K_i}$), Nb)$_{DK}$

   (5) MS → VLR: VLR $\triangleleft$ NewA$_8$(Nb)$_{DK}$, MS$|\equiv$ NewA$_3$(Na)$_{K_i}$, on receiving check NewA$_8$(Nb)$_{DK}$

4. *Message Meaning Rule:*

   (1) $\frac{MS|\equiv(MS\overset{K_i}{\leftrightarrow}HLR)\wedge(VLR\overset{DK}{\leftrightarrow}MS),MS\lhd NewA_3(Na,LAI)_{K_i}}{MS|\equiv HLR\upharpoonleft NewA_3(Na,LAI)_{K_i}}$

   (2) $\frac{VLR|\equiv NewA_3(Na)_{K_i}\wedge(VLR\overset{DK}{\leftrightarrow}MS),VLR\lhd NewA_8(NewA_3(NewA_3(Na,LAI)_{K_i})_{K_i},Nb)_{DK}}{VLR|\equiv HLR\upharpoonleft NewA_8(NewA_3(NewA_3(Na,LAI)_{K_i})_{K_i},Nb)_{DK}}$

5. *Nonce/Timestamp Verification Rule:*

   (1) $\frac{MS|\equiv\#(Na)\wedge\#(Nb),MS|\equiv HLR\upharpoonleft NewA_3(Na,LAI)_{K_i}}{MS|\equiv HLR|\equiv NewA_3(Na,LAI)_{K_i}}$

   (2) $\frac{VLR|\equiv\#(Na)\wedge\#(Nb),VLR|\equiv HLR\upharpoonleft NewA_8(NewA_3(NewA_3(Na,LAI)_{K_i})_{K_i},Nb)_{DK}}{MS|\equiv HLR|\equiv NewA_8(NewA_3(NewA_3(Na,LAI)_{K_i})_{K_i},Nb)_{DK}}$

6. *Jurisdiction Rule:*

   (1) $\frac{MS|\equiv HLR\Rightarrow NewA_3(Na,LAI)_{K_i},MS\lhd VLR\upharpoonleft NewA_3(Na,LAI)_{K_i}}{MS|\equiv VLR|\equiv HLR}$

   (2) $\frac{VLR|\equiv HLR\Rightarrow NewA_8(NewA_3(NewA_3(Na,LAI)_{K_i})_{K_i},Nb)_{DK},Z}{VLR|\equiv HLR|\equiv MS}$,

   where Z = VLR $\lhd$ VLR $\upharpoonleft$ NewA$_8$(NewA$_3$(NewA$_3$(Na, LAI)$_{K_i}$)$_{K_i}$, Nb)$_{DK}$

7. *Protocol Goals:*

   a) Mutual authentication between the MS-VLR and the VLR-HLR:

   MS $|\equiv$ HLR $|\equiv$ VLR $\wedge$ VLR $|\equiv$ HLR $|\equiv$ MS $\rightarrow$ MS $|\equiv$ VLR $\wedge$ VLR $|\equiv$ MS. Thus, mutual authentication holds.

   b) Key agreement between the MS and the VLR:

   A DK key is used between the VLR and the MS to provide agreement.

   MS $|\equiv$ #(Na), MS $|\equiv$ DK $\wedge$ #(Nb), since DK = NewA$_3$(Na)$_{K_i}$

   VLR $|\equiv$ #(Na), VLR $|\equiv$ DK $\wedge$ #(Nb), since HLR $\rightarrow$ VLR $\upharpoonleft$DK

   c) Key freshness between the MS and the VLR:

   MS $|\equiv$ #(Na) $\wedge$ VLR $|\equiv$ #(Na), VLR $|\equiv$ #(Nb) $\wedge$ MS $|\equiv$ #(Nb), DK = NewA$_3$(Na)$_{K_i}$. Thus, key freshness between the MS and the VLR holds.

   d) Confidentiality between the MS and the VLR:

   $\frac{MS|\equiv(MS\overset{DK}{\leftrightarrow}VLR),MS\lhd NewA_5(Msg)_{DK}}{MS|\equiv VLR\upharpoonleft Msg}\wedge\frac{VLR|\equiv(VLR\overset{DK}{\leftrightarrow}MS),VLR\lhd NewA_5(Msg)_{DK}}{VLR|\equiv MS\upharpoonleft Msg}$

   e) Resistance replay attack between the MS and the VLR:

   If the attacker gets #Na from message (1), he/she is unable to forge Message (1) and (2) because he/she does not know K$_i$. If the attacker gets #Nb from message (4), he/she is unable to forge Message (4) and (5) because he/she does not know DK. Since, Na and Nb will be changed at the next time, hence, goal of resistance replay attack between the MS and the VLR is achieved.

   f) Resistance man-in-the-middle attack between the MS and the VLR:

   Since, the attacker knows neither DK nor A5 encryption algorithm, hence, it prevents the communication from being eavesdropped.

g) Resistance redirection attack between the MS and the VLR:

Since, DK = NewA$_3$(Na, LAI)$_{K_i}$ is used to maintain the integrity of LAI, hence, it prevents the system from the redirection attack.

h) Resistance impersonation attack between the MS and the VLR:

(1) Adversary tries to impersonate the MS: Since, NewA$_3$(Na, LAI)$_{K_i}$ is computed at MS and compared at HLR, thus, it avoids the network from impersonation attack. Additionally, the attacker must reply with a valid response NewA$_8$(Nb)$_{DK}$ to the VLR, but, he/she does not have DK key.

(2) Adversary tries to impersonate the VLR: The integrity value NewA$_3$(Na, LAI)$_{K_i}$ at MS and at HLR is violated. Additionally, if the MS receives NewA$_8$(NewA$_3$(NewA$_3$(Na, LAI)$_{K_i}$)$_{K_i}$, Nb)$_{DK}$ at any time, then the connection is terminated because the MS had not previously sent any request to the VLR.

3.8 Secure Algorithms for SAKA Protocol

This section proposes NewA3, NewA8, and NewA5 algorithms for the SAKA protocol in the GSM network.

3.8.1 COMP(NewA3/NewA8) Algorithm

The A3 and A8 algorithms were implemented jointly in the earlier version of COMP-128. But, we propose to implement NewA3 and NewA8 algorithms separately. The reason is that in the SAKA protocol, the NewA3 algorithm is shared between the MS and the HLR while the NewA8 algorithm is shared between the MS and the VLR.

3.8.2 NewA3 Algorithm

The NewA3 algorithm is shared between the MS and the HLR only. There are three different cases, where this algorithm is used in the SAKA protocol.

Case-1: Generate Delegation Key DK at the MS and the HLR.

Input: 1) T$_1$ (64 bits): Timestamp; 2) K$_i$ (128 bits): Secret key shared between the MS and the HLR; *Output:* DK (128 bits): Delegation Key

Case-2: Generate MAC_1 at MS and the HLR.

Input: 1) LAI (40 bits): Location Area Identifier; 2) T_1 (64 bits): Timestamp; 3) K_i (128 bits): Secret key shared between the MS and the HLR; *Output:* MAC_1 (64 bits): Message Authentication Code

Perform bitwise-XOR between T_1 and LAI (with padding if needed), and its output must be used as input to the NewA3 algorithm with K_i key. Here, the LAI must be padded with 24 bits or 3 bytes, *i.e.*, FFF.

Case-3: Generate VLR_C at the HLR and the MS.

Input: 1) MAC_1 (64 bits): Message Authentication Code; 2) K_i (128 bits): Secret key shared between the MS and the HLR; *Output:* VLR_C (64 bits): Certificate of VLR

The NewA3 algorithm is proposed similar to HMAC-SHA256, which is considered as a secure message authentication code.

NewA3 Algorithm: The basic structure of the proposed NewA3 algorithm is as follows:

1. Append zeros (384 zeros) to the left end of key K_i (128-bit) to create a 512-bit string of K_i.

2. Perform XOR between the K_i (512-bit) and the input padding (00110110 repeated 64 times) to produce 512-bit block S_i.

3. Append a zero before the message ($M' = 0||M$) and then append the result to S_i, where each message M' block is of 512-bit in size. If not, make it 512-bit size using padding (zeros), thus, add 448 zeros to the message (64-bit) making it a block of 512-bit. Doing this prevents HMAC from the related key-based narrow pipe attack [77].

4. Apply hashing to the stream generated in step 3 using an initialization vector (IV) of 256-bits. This process generates a hash code of 256-bit.

5. XOR K_i (512-bit) with the output padding (01011100 repeated 64 times) to produce the 512-bit block S_0.

6. Apply padding to the hash code generated in step 4 with 256-bit (256 zeros) and then append the hash code to the S_0.

7. Apply hashing to the stream generated in step 6 and generate the output 256-bit using IV (256-bit). Now, consider the first 128 bits and first 64 bits for the generation of DK and MAC_1/VLR_C according to case-1 and case-2/case-3 respectively.

3.8.3 NewA8 Algorithm

The NewA8 algorithm is shared between the MS and the VLR, and is used in two different cases.

Case-1: Generate MAC_2 at the VLR and the MS.

Input: 1) VLR_C (64 bits): Certificate of VLR; 2) Count (24 bits): Integer Number; 3) DK (128 bits): Delegation Key; *Output:* MAC_2 (64 bits): Message Authentication Code

Perform bitwise-XOR between the VLR_C and the count (with padding if needed), and its output must be used as input to the NewA8 algorithm with DK key. Here, count must be padded with 40 bits or 5 bytes, *i.e.*, FFFFF.

Case-2: Generate SRES at MS and at VLR.

Input: 1) count (24 bits): Integer Number; 2) DK (128 bits): Delegation Key

Output: SRES (32 bits): Signed Response

The NewA8 algorithm is proposed similar to HMAC-SHA1, which is another secure message authentication code.

NewA8 Algorithm: The basic structure of the proposed NewA8 algorithm is as follows:

1. Append zeros (384 zeros) to the left end of key K_i (128-bit) to create a 512-bit string of K_i.

2. Perform XOR between the K_i (512-bit) and the input padding (00110110 repeated 64 times) to produce 512-bit block S_i.

3. Append a zero before the message ($\mathrm{M}' = 0||\mathrm{M}$) and then append the result to S_i, where each message M′ block is of 512-bit in size. If not, make it 512-bit size using padding (zeros), thus, add 448 zeros to the message (64-bit) making it pf a block of 512-bit.

4. Apply hashing to the stream generated in step 3 using an initialization vector (IV) of 160-bits. This process generates a hash code of 160-bit.

5. XOR K_i (512-bit) with the output padding (01011100 repeated 64 times) to produce the 512-bit block S_0.
6. Apply padding to the hash code generated in step 4 with 160-bit (160 zeros) and append the hash code to the S_0.
7. Apply hashing to the stream generated in step 6 and generate the output 160-bit using IV (160-bit). Now, consider the first 64 bits or first 32 bits for the generation of MAC_2 and SRES according to case-1 and case-2 respectively.

Table 3.7: Performance of original COMP(A3/A8) and NewA3 algorithms

Parameters	Original COMP (A3/A8)	NewA3		
		DK	MAC	VLR_C
Avg. Execution Time	32678	211346398	232429396	223993679
Avg. Heap Memory Used	689711	4518168	4518184	4518184
Avg. Total Memory Used	13088293	17766968	17766536	17766520
Process CPU Time	468003000	686404400	686404400	686404400
Used Swap Space	20279377	20380631	20361625	20519444
Used Physical Space	13135052	12300124	12196577	12335169
Committed Virtual Memory	46338048	44998656	45006848	46636480

Table 3.8: Performance of NewA8 algorithm

Parameters	MAC	SRES
NewA8 Avg. Time	190230279	189684939
Avg. Heap memory Used	4521480	4521984
Avg. Total Memory Used	17764296	17764864
Process CPU Time	639604100	592803800
Used Swap Space	20318330	19905536
Used Physical Space	11865579	11725455
Committed Virtual Memory	47562752	47575040

All the proposed protocols simulation and algorithms implementation are performed on Core i3 processor with 256 MB RAM, 320 GB hard drive, Windows7 OS, in JDK1.6 environment. In all tables, the unit of time is nanoseconds and memory/space is measured in bytes. The performance evaluation of the original COMP(A3/A8) algorithm with respect to various parameters have been summarized in Table 3.7. The average time to execute the COMP(A3/A8) algorithm is 0.00003 seconds while the process CPU time is 0.4 seconds. The performance of NewA3 and NewA8 algorithms with respect to various parameters can be observed from Table 3.7 and Table 3.8 respectively. These parameters are average execution time, average heap and total momery used, process CPU time, swap and physical space, and committed vitrual memory. Table 3.7 shows the statistics of the output parameters DK, MAC, and VLR_C that are generated by NewA3 algorithm, while Table 3.8 represents the computation of parameters MAC and SRES that are generated by NewA8 algorithm. The average time for NewA3 algorithm to generate DK, MAC (*i.e.*, MAC_1), and VLR_C is 0.21 seconds, 0.23 seconds, and 0.22 seconds respectively. Similarly, the average time for NewA8 algorithm to generate MAC, *i.e.*, MAC_2, and SRES is 0.19 seconds and 0.18 seconds respectively. The HMAC function provides more security than the normal hash function and uses a secret key, which makes it more difficult to vulnerable than the hash function, where no key is used. Thus, the proposed NewA3 (based on HMACSHA256) and NewA8 (based on HMACSHA1) algorithms are more secure than the original COMP(A3/A8) algorithm.

3.8.4 NewA5 Algorithm

The NewA5 algorithm is shared among MS, VLR, and HLR. The NewA5 algorithm is used to encrypt and decrypt the message with DK key during the communication between the MS and the HLR/VLR over the network.
Input: 1) Data (variable size): Data to be sent from the MS to the BTS and vice versa (consider that data is formed into blocks and each block is of 128 bits); 2) DK (128 bits): Delegation Key *Output:* E/D Data (128 bits): Encrypted/Decrypted Data;

The proposed NewA5 algorithm is similar to the Blowfish algorithm, which is a symmetric algorithm. Blowfish has a 64-bit block size and a variable key length from 32 bits upto 448 bits. It is a 16-round Feistel cipher and uses large key-dependent S-boxes. Blowfish makes use of key K that ranges from 32 bits to 448 bits (1 to 14 32-bit words). This key is used to generate 18 32-bit subkeys that are stored in an array P[0...17] and four 256 S-box entries with 32-bit each as an array S[0...3][0...255]. For a rapid execution, the S and P arrays can be stored rather than re-derived from key, each time algorithm is used. It requires over 4

KB of memory [78]. The reason to choose Blowfish as a basis for the NewA5 algorithm is that till now there is no complete attack on Blowfish algorithm. All the keys in P[0...17] and S[0...3][0...255] are stored onto the SIM card. Nowadays, it is quite possible because 128 KB memory-based SIM cards are available in the market. Thus, the merits of Blowfish algorithm are considered and NewA5 algorithm is proposed and implemented where each block size is of 128 bits while Blowfish has block size of 64 bits. The NewA5 algorithm uses two Feistel cipher F_1 (for 0...63 bits of data block) and F_2 (for 64...127 bits of data block) with 16 rounds each, which provide more computational security to the algorithm. The structures of both Feistel ciphers are as follows:
$F_1(x) = ((S_0, 24 + S_1, 16 \bmod 2^{32}) \oplus S_2, 8) + S_3, 0 \bmod 2^{32}$, and $F_2(x) = ((S_0, 28 + S_1, 21 \bmod 2^{32}) \oplus S_2, 14) + S_3, 7 \bmod 2^{32}$

NewA5 Algorithm: The structure of NewA5 algorithm is as follows:

1. Initialize the 18 32-bit subkeys in an array P[0...17] and four 256 S-box entries with 32-bit each in an array S[0...3][0...255] using the fractional part of constant PI (3.141...), P_1 become leftmost 32 bits of hexadecimal digits of PI (3.141...) and so on. The DK key is used to generate 18 32-bit subkeys.

2. Now, perform bitwise-XOR of P[0] with the first 32 bits of DK key, XORed P[3] with the second 32-bits of the DK key, and so on for all bits of DK key (up to P[17]). The cycle of key bits are repeated until the entire P-array are XORed with key bits.

3. Encrypt 128-bit block of all-zero string using the P-array of subkeys and S-array of S-boxes. Replace P_1 and P_2 with the output of encryption process.

4. Encrypt the output of step 3 using the modified P and S arrays. Replace P_3 and P_4 with ciphertext output.

5. Continue the process, replacing all elements of P-array and then all four S-boxes in order, with the output of continuously changing the algorithm.

6. Encrypt the given input starting at a given offset and place the result in the provided buffer starting at a given offset. The input will be an exact multiple of our block size.

The plaintext is divided into 4 32-bit halves LE, LEM, REM, and RE. For each of 16 rounds do the following:
For i = 1 to 16 do
$\mathrm{LEM}_i = \mathrm{LE}_i \oplus \mathrm{P}_i$ and $\mathrm{RE}_i = \mathrm{REM}_i \oplus \mathrm{P}_i$;
$\mathrm{LE}_i = \mathrm{F}_1[\mathrm{LEM}_i] \oplus \mathrm{LEM}_{i-1}$ and $\mathrm{REM}_i = \mathrm{F}_2[\mathrm{RE}_i] \oplus \mathrm{RE}_{i-1}$;
$\mathrm{LE}_{17} = \mathrm{LEM}_{16} \oplus \mathrm{P}_{18}$ and $\mathrm{REM}_{17} = \mathrm{RE}_{16} \oplus \mathrm{P}_{18}$;
$\mathrm{LEM}_{17} = \mathrm{LE}_{16} \oplus \mathrm{P}_{17}$ and $\mathrm{RE}_{17} = \mathrm{REM}_{16} \oplus \mathrm{P}_{17}$;

7. Decrypt the given input starting at a given offset and place the result in the provided buffer starting at a given offset. The input will be an exact multiple of our block size. The ciphertext is divided into 4 32-bit halves LD, LDM, RDM and RD. The decryption involves the use of subkeys in reverse order. However, the algorithm direction is same as was in encryption process. For each of 16 rounds do the following:
For i = 1 to 16 do
$\mathrm{LDM}_i = \mathrm{LD}_{i-1} \oplus \mathrm{P}_{19-i}$ and $\mathrm{RD}_i = \mathrm{RDM}_{i-1} \oplus \mathrm{P}_{19-i}$;
$\mathrm{LD}_i = \mathrm{F}_1[\mathrm{LDM}_i] \oplus \mathrm{LDM}_{i-1}$ and $\mathrm{RDM}_i = \mathrm{F}_2[\mathrm{RD}_i] \oplus \mathrm{RD}_{i-1}$;
$\mathrm{LD}_{17} = \mathrm{LDM}_{16} \oplus \mathrm{P}_1$ and $\mathrm{RDM}_{17} = \mathrm{RD}_{16} \oplus \mathrm{P}_1$;
$\mathrm{LDM}_{17} = \mathrm{LD}_{16} \oplus \mathrm{P}_2$ and $\mathrm{RD}_{17} = \mathrm{RDM}_{16} \oplus \mathrm{P}_2$;

Table 3.9: Performance of A5/1, A5/2, NewA5, and Blowfish algorithms

Parameters	A5/1	A5/2	NewA5	Blowfish
Avg. Encryption Time	65216	61997	6204	737350
Avg. Decryption Time	65310	61997	4198	835128
Avg. Heap Memory Used	351280	351247	378416	379392
Avg. Total Memory Used	12694936	12693653	12741360	12739132
Process CPU Time	421202700	436802800	436802800	405602600
Used Swap Space	19895787	19924787	20396113	19875184
Used Physical Space	12081725	12053094	11875450	12805283
Committed Virtual Memory	46292992	46305280	46235648	46247936

The performance of A5/1 and A5/2 algorithms based on different parameters can be observed in Table 3.9 along with a difference of runtime performance between the NewA5 and the original Blowfish algorithms. From Table 3.9, it is clear that the NewA5 algorithm is much faster than the original Blowfish,

Table 3.10: Performance of NewA5-CTR, NewA5-CFB, and NewA5-OFB

Parameters	NEWA5-CTR	NEWA5-CFB	NEWA5-OFB
Avg. Encryption Time	19705330	18570896	19122438
Avg. Decryption Time	462114	454324	464074
Avg. Heap Memory Used	1625172	1625028	1624283
Avg. Total Memory Used	14927504	14927344	14926583
Process CPU Time	624004000	608403900	639604100
Used Swap Space	19993026	11924418	19819274
Used Physical Space	11918458	11914403	11704852
Committed Virtual Memory	48377856	48414720	48369664

A5/1, and A5/2 algorithms. The average time to encrypt and decrypt the message by A5/1, A5/2, Blowfish, and NewA5 algorithms are 0.000065, 0.000061, 0.000737350, 0.0000062 seconds and 0.000065310, 0.000061997, 0.000835128, 0.000004198 seconds respectively. Thus, the NewA5 algorithm provides better results for encryption and decryption. One major application for the stream ciphers is very high speed data encryption, this means that the block cipher-based keystream generators are not the complete solution [79]. Due to convenience of use of block ciphers in various protocols, the block cipher behavior can be converted into the stream cipher via modes of operations Counter (CTR), Output Feedback (OFB), and Cipher Feedback (CFB). Since, the original A5/1 and A5/2 algorithms were stream ciphers in nature, thus, we have considered the NewA5 algorithm with three modes of operations, *i.e.*, CFB, CTR, and OFB to convert the NewA5 algorithm message blocks into the bits of streams. Table 3.10 presents the performance of NewA5 algorithm with counter (NewA5-CTR), CFB mode (NewA5-CFB), and OFB mode (NewA5-OFB). All the parameters of these algorithms generate almost same results as shown in Table 3.10, however, in CTR mode, the encryption and decryption can be performed in parallel, thus, the NewA5-CTR is choosen to provide ciphering in the SAKA protocol.

3.9 Security Analysis of Proposed Algorithms for SAKA Protocol

This section provides the justified prevention for the proposed algorithms against various attacks exist in the GSM network along with the security analysis of

NewA3, NewA8, and NewA5 algorithms in terms of cryptanalysis, brute force analysis, and operational analysis.

3.9.1 Attacks on Existing GSM Algorithms

This subsection discusses the security aspects of the proposed algorithms against various attacks that have been found on existing GSM algorithms. The security attacks on the existing GSM algorithms are partitioning attack, narrow pipe attack, collision attack, man-in-the-middle attack, and interleaving attack.

1. *Partitioning Attack:* In case of COMP-128 algorithm, the important characteristic is that the processor can only address 8 bits, but, the first S-box consists of 512 values, which need an address room of 9 bits. Thus, the table needs to be split in minimum two parts. The IBM engineers have made the assumption that the table is just split in the middle [6]. In the proposed scheme, the NewA3 and NewA8 algorithms are implemented separately unlike as in the original GSM protocol where most operators implement COMP-128 algorithm instead of two algorithms. Since, the NewA3 and NewA8 algorithms are based on HMAC functions and do not use S-boxes in the algorithm, thus our algorithms are secure from the partitioning attack.

2. *The Narrow Pipe Attack:* 95% of the collisions in the output are originated in the second round of COMP-128, whereas a collision in the first round is impossible, since it is a one-to-one mapping [80]. Briceno et al. found that in particular, bytes i, i+8, i+16, i+24 at the output of the second round depend only on bytes i, i+8, i+16, i+24 of the input to the COMP-128 algorithm [8]. This fact is called narrow pipe. This attack can compromise the HMAC with hash function theoretically only with single and related key [77]. Since one of the attributes of HMAC is that it considers a hash function as a black box, without any requirement to modify the basic implementation. Here, the aim is to find a way, which does not change the hash function definition and prevents the HMAC from this attack. Peyrin T. et al. proposed a solution where an extra fixed bit (or byte) is placed just before the input message M [77]. To prevent the NewA3 and NewA8 algorithms from the narrow pipe attack, (i) use different initialization vec-

tor (IV) values for inner and outer hash functions in HMAC, (ii) append a zero before the message ($M' = 0||M$) and then append the result to S_i in both the algorithms, where each message M′ block is of 512-bit in size, and (iii) perform XOR operation with a constant 10101010 to the inner and/or outer hash message input to differentiate the input and output computations. We have implemented the first two options to prevent the HMAC-based algorithms from the narrow pipe attack while option 3 may increase computation, hence, is not considered.

3. *Collision Attack:* This attack attempts to compute secret authentication key. An adversary can extract the key information, when two different inputs to the card algorithm produce the same output. In the SAKA protocol, at every MS, the DK key is generated by passing a unique key K_i and a timestamp T_1 to a pre-specified NewA3 algorithm, thus the possibility to generate the same output by two MS is almost impossible. Since, the NewA3 and NewA8 algorithms are based on the HMAC functions, thus, are free from collision attack because collision attack does not affect the security of cryptographic hash/HMAC function [81].

4. *Interleaving Attack:* In the interleaving attack, an adversary can derive the information from the current or previous authentication exchanges. The input for different cases of NewA3 and NewA8 algorithms are (T_1, T_1, and MAC_1, which is also based on T_1) and (count, count) respectively. These T_1 and count change every time an authentication request is generated within an expiry time. Thus, if the other parameters are same for NewA3 and NewA8 algorithms then an adversary cannot derive DK or K_i key-based on previous authentication exchanges. Thus, these algorithms are secure from the interleaving attack.

5. *Man-in-the-middle Attack:* It allows an intruder to access the secret information by intercepting and altering the communication between the communicating parties. Since, in the SAKA protocol, the communication between the MS and the VLR/HLR takes place using the NewA5 algorithm, which is used to encrypt and decrypt the message, thus, the GSM network is free from the MITM attack.

3.9.2 Security Analysis of NewA3 and NewA8 Algorithms

This subsection discusses the security analysis of NewA3 and NewA8 algorithms based on cryptanalysis, brute force analysis, and operational analysis.

1. *Cryptanalysis:* As per the discussion stated in the previous subsection, the NewA3 and NewA8 algorithms are free from partition attack, narrow pipe attack, interleaving attack and collision attack. The NewA3 and NewA8 algorithms are based on the structure of HMAC functions, which are the combinations of hash functions and MAC functions, thus, it is very difficult to break these functions. These HMAC functions are used with SHA256 and SHA1 hash functions respectively for NewA3 and NewA8 algorithms. No full attack has been found on the HMAC-SHA256 and HMAC-SHA1.

2. *Brute Force Analysis:* The size of keys used in NewA3 and NewA8 algorithms are 128 bits each for K_i and DK keys. The possible brute force attack for both the algorithms is 2^{128}, each for NewA3 and NewA8, which is difficult to break in the real world scenario. The brute force analysis for HMAC-SHA256 and HMAC-SHA1 are 2^{256} and 2^{160} respectively, which is very difficult to prove vulnerable.

3. *Operational Analysis:* Since, the NewA3 and NewA8 algorithms are based on complex functions HMAC-SHA256 and HMAC-SHA1 respectively, thus, it is not easy to interpret the structure and operations performed by these algorithms. Additionally, these functions use two separate initialization vectors and additional padding (input and output) to make the operational analysis more difficult for an attacker. The operators may also implement a variant of these algorithms.

3.9.3 Security Analysis of NewA5 Algorithm

This subsection discusses the security analysis of NewA5 algorithm in terms of cryptanalysis, brute force analysis, and operational analysis.

1. *Cryptanalysis:* In the NewA5 algorithm, the sub-keys and S-boxes are generated by a repeated process, similar to the Blowfish algorithm. This makes the cryptanalysis of NewA5 very difficult. However, the boxes are

well designed to resist security attacks, while they are randomly generated in Blowfish. There is no effective cryptanalysis publicly known on the full-round version of Blowfish. Since, in the NewA5 algorithm, the DK key is mixed with the initial set of keys to generate a final keys for each round, thus, the NewA5 algorithm is free from the reflection attack.

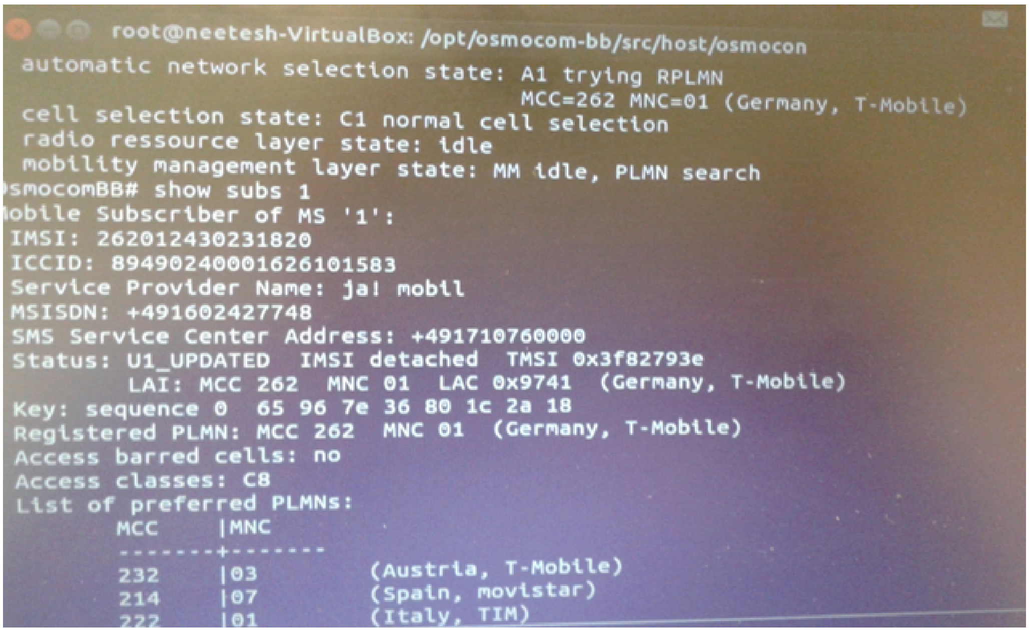
2. *Brute force Analysis:* Brute force analysis depends on the size of key used in the algorithm, which is of 128 bits for DK key. Thus, the possible brute force attack is 2^{128}, which is not easy to break.

3. *Operational Analysis:* Two primitive operations, addition and bitwise-XOR, are considered as a part of the NewA5 algorithm. These two operations do not commute and make cryptanalysis and operational analysis more difficult. Apart from this, the block size of the NewA5 algorithm is 128 bits and operations are performed in four halves of data (32-bit each). These four 32-bit data are processed with two different complex functions F_1 and F_2 for the encryption and decryption. The NewA5 algorithm is also able to prevent the GSM network from the man-in-the-middle attack, thus, the NewA5 algorithm with 128-bit block size is secure.

3.10 Analysis of Attacks on GSM Phone

In order to analyze GSM protocol stack more closely, one project called "OpenBSC" was started in 2008 at network side, which works with OpenBTS/nanoBTS, and developers/researchers can test it in the real world environment with additional hardware equipments. Another project "Open Source Mobile Communication Base Band (OsmocomBB)" was started in 2010 at handset side, which facilitates the developers/ researchers to run and test the real SIM card and other related applications on real phone. Recently, in 2013, researchers demonstrated a possibility of DoS attack and mobile terminated session hijacking [82]. Such an analysis of exploiting various attacks in the GSM network opens door for evaluating and observing the GSM protocol stack in-depth. We explored various services and functionalities provided by these open source software namely OsmocomBB, OpenBSC, and OpenBTS, observed the outcomes of different applications (like mobile, ccch_scan etc.) with Wireshark, and analyzed the GSM protocol stack for the possibilities of attacks in the GSM network.

For the DoS and session hijacking attacks, the main interested logical channels in cellular networks are Paging Channel (PCH) and Random Access Channel (RACH). The PCH is used by the BTS, where it informs the MS about an incoming service (on downlink) while the RACH is utilized by the MS for requesting a dedicated channel from the BTS (on uplink). A DoS attack is possible if an adversary replies with a paging request answer faster than the victim. In such a case, the reply of the victim will be ignored. As a result, the network sends a channel release message to the victim. The setup will not be completed if the adversary does not input the correct cryptographic keys for the authentication and encryption. Since, an adversary do not have the same, thus, the call will be dropped. The repetition of this process results in DoS attack.



```
 automatic network selection state: A1 trying RPLMN
                                    MCC=262 MNC=01 (Germany, T-Mobile)
 cell selection state: C1 normal cell selection
 radio ressource layer state: idle
 mobility management layer state: MM idle, PLMN search
smocomBB# show subs 1
obile Subscriber of MS '1':
IMSI: 262012430231820
ICCID: 89490240001626101583
Service Provider Name: ja! mobil
MSISDN: +491602427748
SMS Service Center Address: +491710760000
Status: U1_UPDATED  IMSI detached  TMSI 0x3f82793e
        LAI: MCC 262  MNC 01  LAC 0x9741  (Germany, T-Mobile)
Key: sequence 0  65 96 7e 36 80 1c 2a 18
Registered PLMN: MCC 262  MNC 01  (Germany, T-Mobile)
Access barred cells: no
Access classes: C8
List of preferred PLMNs:
        MCC     |MNC
        -------+-------
        232     |03         (Austria, T-Mobile)
        214     |07         (Spain, movistar)
        222     |01         (Italy, TIM)
```

Figure 3.5: OsmocomBB: IMSI and session key extraction

The paging procedure is followed by the cipher setup. Several countries do not provide the encryption and proper authentication. For example, only under 20% of the networks analyzed by the gsmmap project authenticate mobile terminated phone calls 100% of the time and 50% of the tested networks only authenticate 10% of the services [82]. An adversary can easily takeover a session in such networks. If the adversary is able to win the race condition by sending the faster response than victim then it may be possible to hijack a session in the unencrypted network or weak encrypted network. An adversary impersonate the victim by cracking the session key of weak encrypted network, if the proper authentication is not provided. The session key can be extracted by the Kraken tool or with a blackberry handset in engineering mode. In order to implement such attacks, we require the hardware and software to interact with GSM base station that allow us to access and modify the GSM stack using baseband (BB) implementation. In the current scenario, we have three choices to select a radio device (as a handset): USRP, Vitelcom TSM30, and TI Calypso chipset-based phones. All three devices are used as GSM radio transceivers with the baseband software modifications. Presently, there is no baseband implementation available for the USRP. Baseband is available for TSM30, but its code is very complex to follow and work on. They are also not easily available. We preferred to use TI Calypso chipset-based phones. These phones are easy to obtain, inexpensive, and provide baseband implementation. These phones typically have a serial port at 3.3V level, which are sometimes called T191 unlock cable and available in a variety of fashions. In this work, Motorola C118 is used to analyze attacks on GSM phone and the USB variant of PL2303 is used as a serial cable. As shown in Figure 3.5, ISMI and session key are successfully extracted during this work.

3.11 Chapter Summary

This chapter presents an efficient SAKA protocol for a better service in the GSM network. The protocol resolves the primary issue of authentication in the GSM network by providing mutual authentication between the user and the network without increasing the communication overhead. The protocol prevents replay attack, man-in-the-middle attack, redirection attack, and impersonation attack. On an average, the SAKA protocol saves 56% of the total bandwidth used for authentication in the GSM network, when the number of authentication requests within a session are n = 5, 10, 50, and 100. Further, a set of new secure algorithms namely NewA3, NewA8, and NewA5, have been proposed and implemented, in order to overcome the known vulnerabilities in A3, A8, and A5 algorithms used in the original GSM protocol. Additionally, three variants of NewA5 algorithm have been proposed and implemented that convert the block ciphers in the stream ciphers. The NewA5 algorithm with counter mode provides parallelism to the encryption process, hence, it can be preferred as cipher algorithm in the GSM network.

Chapter 4

AKA Protocol in 3G UMTS Network

4.1 Introduction

In the current scenario, third generation UMTS technology has become popular enough to gradually supersede GSM. In order to overcome the security issues in GSM, the UMTS-AKA was developed to authenticate mobile users. Although UMTS-AKA has overcome most of the vulnerabilities and security issues found in the GSM-AKA, nevertheless, it is still vulnerable to redirection attack, impersonation attack, man-in-the-middle attack, and DoS attack. There are several other issues with the UMTS-AKA including the huge bandwidth usage between the HLR and the VLR, large storage space overhead at VLR, and the counter synchronization problem between the MS and the HLR/VLR. This protocol also generates huge communication and computation overheads in order to provide the mutual authentication between the MS and the VLR/HLR.

4.2 UMTS-AKA Protocol

In the UMTS-AKA protocol, each user shares a secret key SK and certain cryptographic functions with the HLR. In addition, the HLR and the MS, each maintains a counter for the synchronization purpose. The cryptographic algorithms/functions are shared between the HLR and the MS including two message authentication codes f_1 and f_2, and three key generation functions f_3, f_4, and f_5.

Table 4.1: Symbols and abbreviations

Symbol	Definitions	Bits
IMSI	International Mobile Subscriber Identity	128
TID	Temporary Identity	128
PID	Proxy Identity	128
ACC	Accumulator Number	24
ReqNo	Request Number	128
Puz	Puzzle	128
Sr	Service Request	8
ID/idx/Count	Identity/Integer Number	28
SQN/XSQN	Sequence Number	48
AMF	Authentication Management Field	48
LAI	Location Area Identity	40
POS	Position Information	40
AUTN/AUTH	Authentication Token	Variable
AV/PAV/MAV/TAV	Authentication Vector	Variable
Y/N	Yes/No Flag	1
CK/TCK/PCK	Cipher key	128
IK/TIK/PIK/IIK	Integrity key	128
AK/XAK	Anonymity key	128
SK/K	Secret key b/w MS-HLR	128
DK	Delegation key	128
TK	Temporary key	128
RAND/RN/PR/ FRESH/RNidx	Random Number	128
MAC/VAC/XMAC/ PMAC/XVAC	Message Authentication Code	64
α/H	Hash Code	64
RES/XRES/PRES	Response/Expected Response	64
PLK/TEK	Payload Encryption key	128
T/t	Timestamp	64
ACK	Acknowledgement	16
Solu$_1$	Solution$_1$ of puzzle	128
Solu$_2$	Complete solution	128

Table 4.2: Cryptographic functions definition

Function	Definition
f_1/FK	Function to generate MAC
f_2/FK	Function to generate RES/XRES
f_3	Key generation function for CK/PCK
f_4	Key generation function for IK/PIK
f_5	Key generation function for AK
f_6	Key generation function for DK
f_7	Key generation function for PLK/Cipher function for TEK
f'	Function to generate ACC_3
GK	Secret session key generation for SK
HK	Random number generation for RNidx
Fx	Key generation function for TK
f_{99}	Key generation function for Ktemp
$\|$	Concatenation

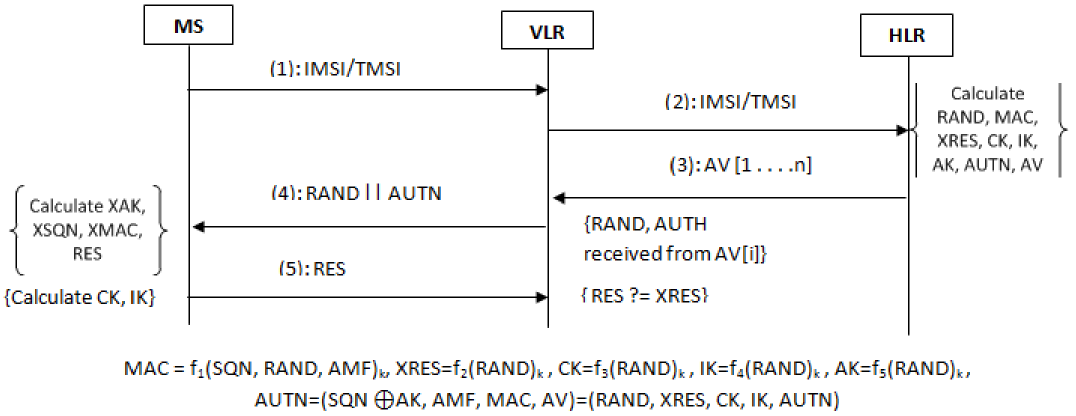

MS
VLR
HLR
(1): IMSI/TMSI
(2): IMSI/TMSI
Calculate RAND, MAC, XRES, CK, IK, AK, AUTN, AV
(3): AV [1 . . . .n]
(4): RAND | | AUTN
Calculate XAK, XSQN, XMAC, RES
{RAND, AUTH received from AV[i]}
(5): RES
{Calculate CK, IK}
{ RES ?= XRES}
MAC = f₁(SQN, RAND, AMF)ₖ, XRES=f₂(RAND)ₖ , CK=f₃(RAND)ₖ , IK=f₄(RAND)ₖ , AK=f₅(RAND)ₖ ,
AUTN=(SQN ⊕AK, AMF, MAC, AV)=(RAND, XRES, CK, IK, AUTN)

Figure 4.1: UMTS-AKA protocol

The UMTS-AKA protocol has shown in Figure 4.1, where the MS and the HLR share a secret key SK and maintain sequence numbers to prevent replay attack [22]. AK/XAK is the anonymity key, which is used to conceal the sequence number in the original UMTS-AKA protocol. Table 4.1 and Table 4.2 describe the definition of various symbols, abbreviations, and cryptographic functions used in various AKA protocols discussed in this chapter.

4.3 Communication, Trust and Attack Models

This section first presents the system model in terms of communication and trust scenario, and then discusses an attack model.

4.3.1 Communication and Trust Model

When a user is in his/her home network, then the mutual authentication takes place between the MS and the HLR. The HLR generates the AVs as per the authentication requests received from various MS. A trust model comes into the picture, when a user moves to a roaming area. The MS requests to one of the roaming operators for providing the service and sends an authentication request to the nearest VLR. The communication of AVs (or some authentication information) takes place between the VLR and the HLR, and then rest of the mutual authentication process executes between the MS and the VLR. In this trust model, it is assumed that a secret key SK is shared between the MS and the HLR. The authentic information based on SK key is generated by the HLR and is sent to the VLR, which avoids the access of such information by any malicious VLR.

4.3.2 Attack Model

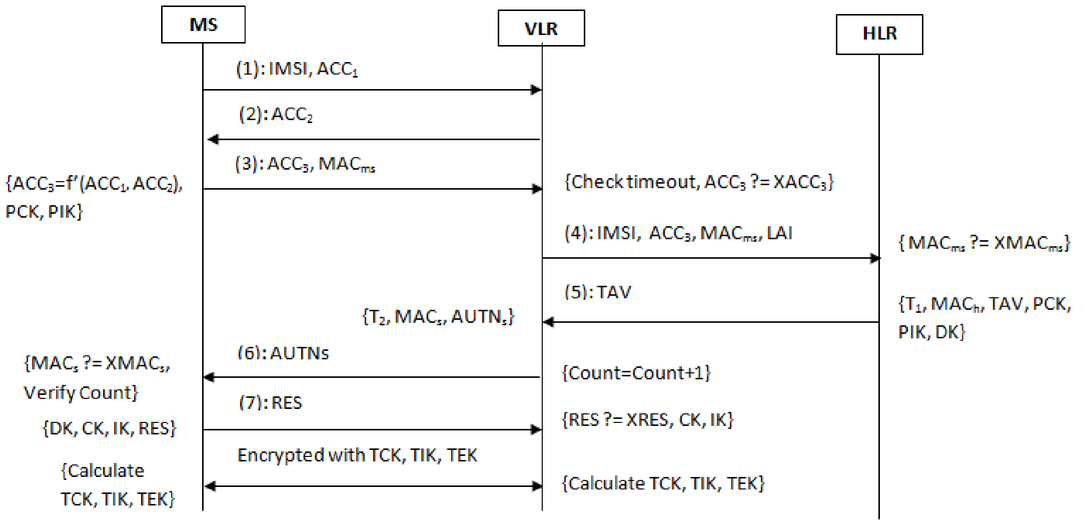
An attack model describes various scenarios where a malicious MS or VLR can access the authentic information, misguide the legitimate MS, and corrupt the network. A malicious VLR can redirect the legitimate MS and can receive the valid tokens using a false base station device such as IMSI catcher. Another possibility of attack is to delay or reuse the authentication messages, if they do not contain any nonce or timestamp value, which leads to the replay attack. Since, the AVs are moved within and between the networks, thus, the authentication process may disturb if the network is corrupted. An adversary can forge the authentication data request to obtain authentication vectors and use it to impersonate the network that is independent of actual location of the user. An attacker can hide itself between the MS and the VLR, and may be able to crack the UMTS security. The attacker can eavesdrop the session initiated by legitimate MS that leads to the man-in-the-middle attack. For such an attack, the adversary must be able to intercept and inject some traffic. Apart from these attacks, various forms of DoS attack such as primary DoS, and distributed DoS (DDoS) are quite possible. The existing DoS and DDoS defense mechanisms anticipate a flood of authentication requests as attacks and exploit the vulnerabilities of the system. The low rate DoS (LDoS) attack is difficult to detect as compared to other forms of DoS attacks as it exploits many factors and vulnerabilities that vary from the iterative hops to the fixed minimum retransmission timeout (RTO). Our primary focus is to prevent the network from authentication requests-based flood DoS attack.

4.3.3 Other Issues of UMTS-AKA Protocol

This subsection describes other existing issues of UMTS-AKA protocol.

1. *Operational Difficulty with Sequence Numbers:* In UMTS-AKA, the HLR maintains a dynamic counter for each MS. Sometimes whenever a sequence number is considered to be not in the correct range, the MS assumes that a synchronization failure has occurred in the HLR and it initiates a resynchronization request to the HLR.

2. *Bandwidth Consumption:* In UMTS-AKA, the HLR sends n AVs to the VLR after authenticating the MS. The VLR needs to make a request for another authentication, when these available AVs are finished. However, it requires a high bandwidth to transmit a huge number of AVs every time.


MS
VLR
HLR
(1): IMSI, ACC₁
(2): ACC₂
(3): ACC₃, MACₘₛ
{ACC₃=f'(ACC₁, ACC₂), PCK, PIK}
{Check timeout, ACC₃ ?= XACC₃}
(4): IMSI, ACC₃, MACₘₛ, LAI
{MACₘₛ ?= XMACₘₛ}
(5): TAV
{T₁, MACₕ, TAV, PCK, PIK, DK}
{T₂, MACₛ, AUTNₛ}
(6): AUTNs
{MACₛ ?= XMACₛ, Verify Count}
{Count=Count+1}
(7): RES
{DK, CK, IK, RES}
{RES ?= XRES, CK, IK}
Encrypted with TCK, TIK, TEK
{Calculate TCK, TIK, TEK}
{Calculate TCK, TIK, TEK}

$MAC_{ms}=XMAC_{ms}=f_1(ACC_3, LAI)_{SK}$, $PCK=f_3(ACC_3)_{SK}$, $PIK=f_4(ACC_3)_{SK}$, $MAC_h=f_1(T_1, AMF)_{SK}$, $DK=f_6(ACC_3)_{SK}$, $TAV=(AUTN_t, PCK, PIK, DK)$, $AUTN_t=(MAC_h, T_1, AMF)$, $MAC_s=f_1(MAC_h, T_1, T_2, Count)_{DK}$, $AUTN_s=(MAC_s, T_2, T_1, AMF, Count)$, $XMAC_s=f_1(f_1(T_1, AMF)_{SK}, T_1, T_2, Count)_{DK}$, $CK=f_3(T_2)_{DK}$, $IK=f_4(T_2)_{DK}$, $RES=XRES=f_2(T_2)_{DK}$, $TCK=PCK\oplus CK$, $TIK=PIK\oplus IK$, $TEK=CK\oplus IK$

Figure 4.2: Proposed ES-AKA protocol

4.4 Proposed ES-AKA Protocol

To defeat the security issues puzzled with the UMTS-AKA protocol, we propose and present an authentication and key agreement protocol namely ES-AKA, which defeats redirection attack, man-in-the-middle attack, DoS attack, and intensely down the impact of network corruption. This ES-AKA protocol strictly follows the framework of 3GPP UMTS-AKA protocol and throw out the problem of synchronization between the MS and the HLR. In the ES-AKA protocol, each MS and its HLR share a secret key SK. Some cryptographic algorithms/functions f_1, f_2, f_3, f_4, f_6, and f_7 are used in this protocol and Table 4.2 describes the role of each function. There are three accumulator numbers (ACC) used in the ES-AKA protocol. They are stored at MS as well as at VLR in first three steps of the ES-AKA protocol. Each ACC has a fixed size of 24 bits that can be clearly observed from Table 4.1. These numbers are sent during the communication to establish a connection for the authentication between the MS and the VLR. ACC_1 and ACC_2 are randomly generated, while the ACC_3 is used to generate the delegation key DK at HLR. The ES-AKA protocol is divided into two phases as shown in Figure 4.2. These two phases are as follows:

Phase-1: In this protocol, first three messages are used to prevent the UMTS network from the DoS attack. First, the MS makes a request to the VLR to create a connection for the authentication by sending IMSI and a random accumulator ACC_1 to the VLR (message (1)).

MS to VLR: IMSI, ACC_1

In response, the VLR stores (IMSI and ACC_1) in its database and sends another random accumulator ACC_2 to the MS, to check whether the MS is active or not (message (2)).

VLR to MS: ACC_2

After sending ACC_2 to the MS, the VLR waits for a response till a random timeout period (random timeout is used to prevent DoS). If the MS does not receive any authentication information during this period, it simply retransmits the request. Further, if the MS is active and is a legitimate MS then it calculates ACC_3 ($ACC_3 = f'(ACC_1, ACC_2)$) as well as MAC_{ms} ($MAC_{ms} = f_1(ACC_3, LAI)_{SK}$)and sends them to the VLR (message (3)).

MS to VLR: ACC_3, MAC_{ms}

The VLR checks the timeout period (which is random in nature) and if the message is received within timeout period then the VLR also calculates ACC_3 at its end with the same f′ function, and checks whether (ACC_3 ?= XACC_3). Here, f′ is a complex function (it may be complex hash function such as SHA1, HMAC etc.), which is known to the MS and the VLR/HLR. If the VLR does not get any response within time period, then the request for the authentication is discarded. Since a legitimate MS may also request for the authentication, so it will wait for a specified timeout period. Here, the timeout period for the MS is larger than the VLR. If a particular MS is discarded for a certain number of attempts then the VLR will mark its IMSI into the blacklist. After suspension period, if the MS gets authenticated, then its IMSI will be removed from that blacklist. After accepting and checking ACC_3, the VLR passes (IMSI, ACC_3, MAC_{ms}, LAI) to the HLR (message (4)), where LAI is the location area identity of the MS.

VLR to HLR: IMSI, ACC_3, MAC_{ms}, LAI

After receiving such information, the HLR calculates $\text{XMAC}_{ms} = \text{f}_1(\text{ACC}_3, \text{LAI})_{SK}$ and compares it with the received MAC_{ms}. If both are equal, then the MS is a legitimate user. The HLR generates (T_1, DK, PCK, PIK) and calculates (MAC_h, TAV), where AMF is Authentication Management Field. Now, the HLR passes TAV to the VLR (message (5)).

HLR to VLR: PCK, PIK, DK, MAC_h, T_1, AMF

Phase-2: The phase-2 is executed for each subsequent authentication within the expiry time of DK key. At VLR, the count is incremented after every transmission of authentication token AUTN_s and stores the token (T_2, MAC_s, AUTN_s). This token is calculated and sent every time, when an AV is requested by the MS. The VLR passes the AUTNs to the MS (message (6)).

VLR to MS: MAC_s, T_1, T_2, AMF, count

After receiving the AUTN_s, the MS verifies the count. If the received count is greater than the previous count, then the MS calculates XMAC_s and compares it with the received MAC_s, otherwise, the request is discarded and connection is terminated. Additionally, at MS, the current received T_2 is compared with the T_2 received in the previous subsequent authentication request. Now, the MS generates (DK, CK, IK, PCK, PIK, RES) and sends RES to the VLR in response to AUTN_s (message (7)).

MS to VLR: RES

The VLR calculates XRES and compares it with the received RES. If the XRES and RES are equal then the VLR generates CK and IK keys. Some standard functions are used in order to generate PCK, PIK, CK, and IK keys. The functions to generate PCK and PIK keys take input of 24 bits while functions to generate CK and IK keys take 64 bits input. All functions generate 128 bits output keys. Now, both, the MS and the VLR, generate TCK, TIK, and TEK keys for Confidentiality, Integrity, and Encryption. Thus, this protocol does not transmit actual CK and IK keys during the authentication.

4.5 Analysis of Proposed ES-AKA Protocol

This section discusses the security and performance analysis of the ES-AKA protocol and compares the proposed protocol with the existing protocols.

4.5.1 Security Analysis

1. *Mutual Authentication between the MS and the HLR:* In ES-AKA, on receipt of message (4), the HLR authenticates the MS by verifying ACC_3 and MAC_{ms}. To authenticate the HLR, the MS checks AUTNs received in message (6). The MS can acquire the expected authentication codes of VLR as $XMAC_s = f_1(f_1(T_1, AMF)_{SK}, T_1, T_2, count)_{DK}$. If $XMAC_s$ is equal to MAC_s, then both, the HLR and the VLR are authenticated. This ensures the mutual authentication between the MS and the HLR. For the subsequent authentications, the MS can still authenticate the HLR with message (6) and (7) with the involvement of DK key.

2. *Mutual Authentication between the MS and the VLR:* In ES-AKA, the VLR authenticates the MS by verifying RES in message (7). After receiving the message, the VLR calculates XRES and checks whether RES ?= XRES, where $XRES = f_2(T_2)_{DK}$. The MS is verified authentic if this equality is true. The same procedure takes place in authenticating the MS, when the VLR receives message (7) while communicating only with the VLR. Now, on receiving $AUTN_s$, the MS computes $XMAC_h$ and $XMAC_s$ to authenticate the VLR, if MAC_s ?= $XMAC_s$ holds, where

XMAC$_h$=f$_1$(T$_1$, AMF)$_{SK}$ and XMAC$_s$= f$_1$(XMAC$_h$, T$_1$, T$_2$, count)$_{DK}$. All this ensures the mutual authentication between the MS and the VLR.

3. *Freshness of Security Keys:* In ES-AKA, the CK and IK are generated from the functions of timestamp T$_2$, thus, the freshness of these keys can be ensured by T$_2$. In message (6), the count is incremented on each successful authentication and can be used to guarantee the freshness of the message. This ensures the freshness of CK and IK keys. Other keys TCK, TIK, and TEK are generated on the basis of freshness of CK and IK keys, thus, these keys also ensure the freshness of their generation.

4.5.2 Resistance to Attacks

1. *Redirection Attack:* In the UMTS-AKA protocol, the LAI, which recognizes the location of the BSS, is not guarded and can be modified by an adversary with some redirection attack. ES-AKA uses message authentication code to maintain the integrity of LAI, thereby preventing the network from redirection attack. The redirection attack fails if the adversary is unable to obtain user′s MS information. In ES-AKA, the MS involves LAI in MAC$_{ms}$ and transmits MAC$_{ms}$ to the VLR in message (3). The authentication request is refused, if the HLR fails to match the LAI sent by the VLR (message (4)) and the LAI embedded in MAC$_{ms}$. Such a technique solves the mischarged billing in the network.

2. *Replay Attack:* The ES-AKA protocol is free from this attack by sending timestamp (T$_1$, T$_2$) or random values (ACC$_1$, ACC$_2$, ACC$_3$) with the transmitted message over the network.

3. *Man-in-the-middle Attack:* A man-in-the-middle attack can occur, when an MS tries to connect to a BTS in weak encrypted network. To cipher payloads, a new encryption TEK key is conversed between the MS and the VLR in the ES-AKA protocol. The TEK inhibits the communication from being eavesdropped while previously in UMTS-AKA no such key was present. Similar to S-AKA, the ES-AKA protocol introduced a new key TEK to make the communication more reliable between the MS and the VLR. The TEK is consulted by the MS and the VLR after exchanging

messages (6) and (7) in the ES-AKA protocol.

4. *Impact of Network Corruption:* The UMTS-AKA protocol does not provide the solution against corrupted VLR/HLR. There are three possible cases for the communication between the MS and the VLR/HLR, and each of that is described as follows:

   Case-1: The adversary tries to impersonate the MS: Let us consider that a VLR is impaired and the adversary can eavesdrop all its messages and can send to another network. Thus, the adversary can impersonate to establish a communication session with the MS in its VLR while the user is in roaming area. Now, consider a fresh uncorrupted roaming network VLR′. The adversary must reply with a valid response RES to the VLR in order to impersonate the MS, but, the adversary cannot obtain correct RES, since RES was only transferred between the HLR and the VLR′.

   Case-2: The adversary requests for AVs with the corrupted network VLR: When the adversary tries to impersonate the network VLR′, the adversary begins the protocol by sending ACC_2 to the MS in reply of message (1) and the MS replies back with ACC_3 and MAC_{ms}. Now, the adversary requests the AVs for the MS through the impaired network VLR. But, after verifying MAC_{ms}, the HLR concludes that the MS is not in the VLR and rejects the concern request.

   Case-3: The adversary tries to impersonate the uncorrupted network VLR′: If the adversary tries to impersonate the network VLR′, an attempt to impersonate the VLR′ gets fail as the MS can verify that the AV was not requested by the VLR′. Thus, the impact of network corruption is drastically reduced with the ES-AKA protocol in comparison to UMTS-AKA protocol.

5. *DoS Attack:* The DoS attack and its variants are addressed in the following scenarios: The attacker MS′ floods the victim VLR with the authentication request by spoofing the IMSI and a random accumulator number ACC_1. Then, ACC_2 are sent from the VLR to the spoofed source MS. Hence, the VLR will not get the final information to complete the authentication request. This leads to half-open authentication requests at the VLR. There is a timeout period for each MS to maintain the state of half-opened au-

thentication requests. If the attacker MS′ causes overflow at victim VLR with the half-open authentication requests, then the VLR cannot accept any new incoming authentication request.

Case-1: The attacker MS′ floods the victim VLR by self IMSI: First, the attacker MS′ can configure self to flood the victim VLR with ACC_1 and correct IMSI, and does not send the final information for the authentication. The VLR responds with ACC_2 message for every authentication request. Since, the attacker MS′ is configured not to send the final information, the VLR will not receive any final information message from the MS′. In this case, the VLR retransmits ACC_2 message and waits for the final information message. This process is repeated until the timeout period is reached. After the timeout period, the VLR resets the authentication request and makes free the resources that are used to maintain the authentication request status.

Case-2: The attacker MS′ floods the victim VLR by spoofing IMSI: The attacker MS′ can flood the victim VLR with ACC_1 by spoofing IMSI. Now, ACC_2 message from the victim VLR will be communicated to the MS, with which the attacker MS′ has spoofed IMSI. There are two different scenarios for such a case.

Scenario-1: In the first scenario, if the actual MS that receives ACC_2 message is not active, then the VLR will not receive any final information from the MS. In this scenario, the process is similar to case-1.

Scenario-2: In the second scenario, if the MS is active, it sends a reset signal to the VLR, since it did not initiate any authentication request. If the victim VLR receives the reset signal, it can free the resources that are used to maintain the authentication request status. The low rate DoS attack exploits the fact that most systems have a RTO of one second. It would be difficult for the attacker to predict the next RTO value if the RTO is set to any random value and this ultimately would help in controlling the rate of attack [83].

During the initial authentication, a malicious MS′ may launch a DoS attack either to the HLR or to the VLR. If the MS′ forges message (3), the forged message can be detected by the HLR on receipt of message (4), and that can also be immediately detected by the VLR with DK key authorized

Table 4.3: Protocols vs. prevention from different attacks

Attacks	UMTS-AKA	AP-AKA	X-AKA	EXT-AKA	EURA-SIP-AKA	COCK-TAIL-AKA	S-AKA	ES-AKA
Replay	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Corrupt N/w	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
MITM	No	No	No	No	No	Yes	Yes	Yes
Redirection	No	Yes	No	No	No	Yes	Yes	Yes
DoS	No	No	No	No	No	No	Partial	Yes

by the HLR, in phase-2 message exchanges. Thus, the ES-AKA protocol resists such DoS attack, since the forged messages can be detected by the HLR and the VLR. Table 4.3 summaries the prevention against known attacks by various proposed and existing AKA protocols for the UMTS network. All the protocols mentioned in Table 4.3 are free from replay attack as well as active attacks in the corrupted network.

The UMTS-AKA, X-AKA, EXT-AKA, and EURASIP-AKA protocols do not prevent MITM and redirection attacks. However, the COCKTAIL-AKA, S-AKA, and ES-AKA protocols are able to stop MITM and redirection attack while the AP-AKA protocol does not resist the MITM attack but, is free from redirection attack. The ES-AKA protocol detects DoS attack (authentication request flood-based attack) and prevents the system, because in both phases of this protocol, DoS attack can be immediately detected by the VLR. However, the S-AKA protocol can solve the problem of DoS attack partially.

4.5.3 Communication Overhead

This subsection calculates the transmitted message size, in order to evaluate total communication overhead with respect to UMTS-AKA, ES-AKA, and other AKA protocols.It is assumed that all protocols are with global size parameters that are specified in Table 4.1. Now, based on each parameter transmitted in the message, total number of bits required for each message can be calculated, and then the sum for all messages sent during each protocol run are computed. The total number of bits in all messages for UMTS-AKA, ES-AKA, and other AKA protocols are as follows:

UMTS-AKA Protocol: Phase-1: (1)+(2)+(3) =128+128+608*n =256+608*n bits, where AUTN=(48⊕48)+48+64=160 bits and AV (n values)=(128+64+128

+128+160)*n = 608*n

Phase-2: (for n values) ((4)+(5))*n= ((128+160)+64)*n= 352*n bits

Total transmitted bits= 256+608*n+352*n = 256+960*n

COCKTAIL-AKA Protocol: Phase-1: (1)+(2)+(3) =(128+128)+(128+128+128)+560 =1200 bits, where PAUTN=128+48+64= 240 bits and PAV=240+64+128+128 = 560 bits

Phase-2: (for n values) ((4)+(5))*n= ((128+240+64)*n = 432*n bits

Total transmitted bits = 1200+432*n bits

AP-AKA Protocol: It is assumed that the maximum idx= 32000= 2^{28} [84]

Phase-1: (1)+(2)+(3)+((4) for n values) = 128+ (128+128+128+64)+ (128+64)+ (128+64+128+ (28+64+64))*n = 768+604*n

Phase-2: (for n values) ((5)+ (6))*n = ((128+156)+64)*n= 348*n

Total transmitted bits = 768+604*n+348*n = 768+952*n

S-AKA Protocol: Phase-1: (1)+(2) = (128+24+8+64)+(128+24+8+64+40)= 488 bits, where AUTN = (64+128+128) = 240 bits

Phase-2: (3)+(4)+(5) = (240+128)+(392)+ 64 = 824 bits

Phase-3: (1)+(2)+(3) = (128+24+8+64)+(392)+(64) = 680 bits

Total transmitted bits = (Phase-1 + Phase-2)+ (Phase-3)*n = 1312+680*n

EURASIP-AKA Protocol: AUTN = 48+48+64 = 160 bits Total transmitted bits (for n values) = ((1)+(2)+(3))*n = ((128+8+40+128+160)+ (128+8+40+128+160) + 64)*n = 992*n bits

X-AKA Protocol: Phase-1: (1)+(2)+(3) = (128+64+64)+(128+64+64)+(240+128) = 880 bits, where AUTH$_h$ = 64+128+48 = 240 bits

Phase-2: (for n values) ((4)+(5))*n = (368+64)*n = 432*n, where MAC$_s$ = 64, AUTH$_s$ = 64+128+128+48 = 368 bits

Total transmitted bits = 880+432*n *EXT-AKA Protocol: Phase-1:* (1)+(2)+(3) = (128+128+64)+(128+128+64) + (64+128+48+128) = 1008 bits, where AUTN$_{hn}$ = 64+128+48 = 240 bits

Phase-2: (for n values) ((4)+(5))*n = ((240+128)+(64))*n = 432*n, where AUTN$_{sn}$ = 64+128+48 = 240 bits

Total transmitted bits = 1008+432*n bits

ES-AKA Protocol: Phase-1: (1)+(2)+(3)+(4)+(5) = (128+24)+(24)+(24+ 64) + (128+ 24+64+40)+ (64+64+48+128+128+128) = 1080 bits

Phase-2: (for n values) ((6)+(7))*n = ((64+64+64+48+28)+64)*n = 332*n

Total transmitted bits = 1080+332*n

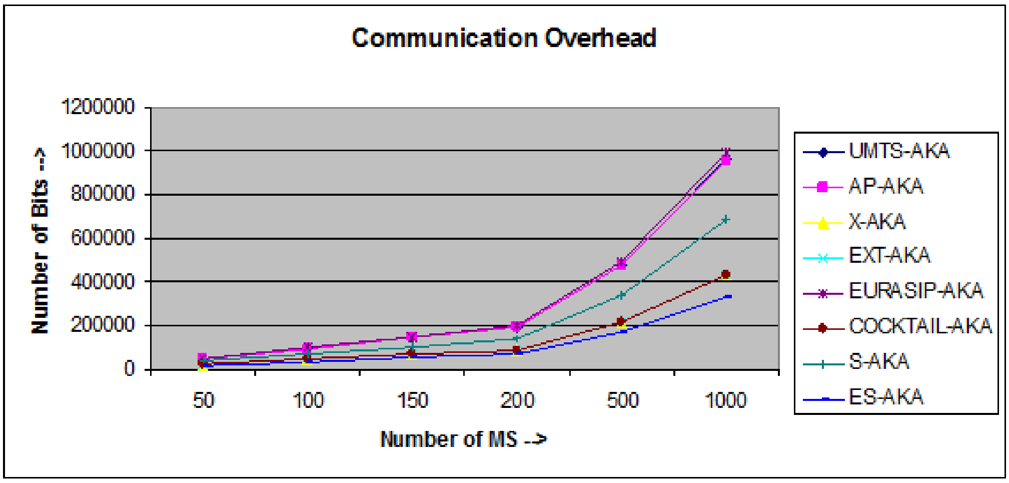


Figure 4.3: Communication overhead of various AKA protocols

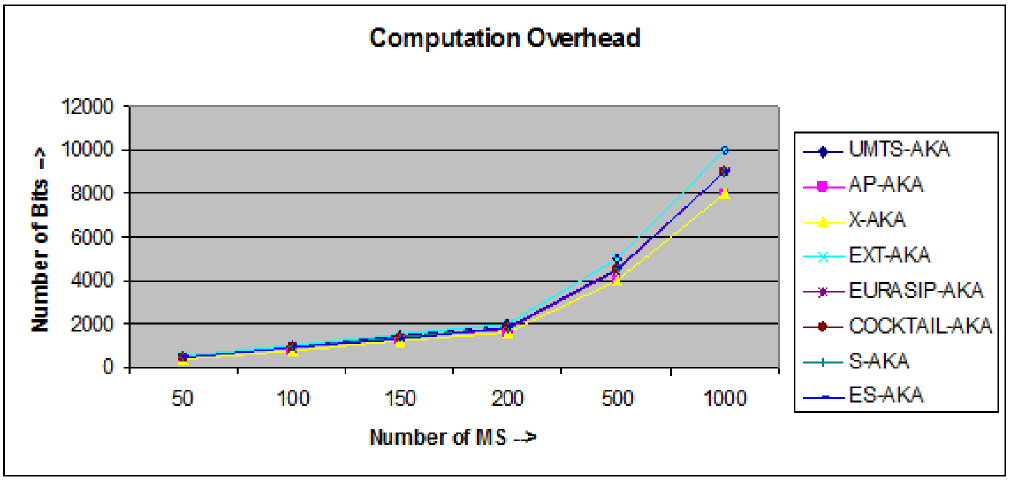


Figure 4.4: Computation overhead of various AKA protocols

In Figure 4.3, a graph is mapped between the number of MS and the number of bits transmitted by different AKA protocols to evaluate the communication overhead of each AKA protocol. It can be easily observed that the ES-AKA protocol generates lesser communication overhead as compared to other AKA protocols.

4.5.4 Computation Overhead

In order to compare the computation of each protocol with the global values, the computational overhead of all security functions are considered a unit value (for the global analysis of all AKA protocols). Further, we calculate that how many functions are required to compute total computation in each protocol.

Below are the statistics to calculate the total number of functions that are used in UMTS-AKA, ES-AKA, and other AKA protocols.

UMTS-AKA Protocol: Phase-1: (for AV, n values) {f_1, f_2, f_3, f_4, f_5}*n = 5*n

Phase-2: f_1, f_2, f_3, f_4, f_5 = 5

Total functions used = Phase-1 + (Phase-2)*n= 5*n +5*n = 10*n

COCKTAIL-AKA Protocol: Phase-1: f_1, f_5, f_3, f_4, f_2, f_2, f_2, f_3, f_4, f_3, f_4, f_2, f_5 = 13

Phase-2: f_1, f_1, f_1, f_2, f_3, f_4, f_2, f_3, f_4= 9

Total functions used = Phase-1 + (Phase-2)*n= 13+9*n

AP-AKA Protocol: Phase-1: {Hk, Fk, Fk, (Fk, Gk, Hk, Fk)*n} = 3+4*n

Phase-2: {Hk, Fk, Fk, Fk} = 4

Total functions used= Phase-1 + (Phase-2)*n = 3+4*n+4*n = 3+8*n

S-AKA Protocol: Phase-1: f_1, f_6, f_1, f_1, f_6 = 5

Phase-2: f_1, f_1, f_1, f_3, f_4, f_2, f_7, f_2, f_3, f_4, f_7 = 11

Phase-3: {f_1, f_6, f_1, f_1, f_1, f_1, f_1, f_2, f_2}*n = 9

Total functions used= (Phase-1 + Phase-2) + (Phase-3)*n = 16+9*n

EURASIP-AKA Protocol: Total functions used (for n values) = {f_5, f_1, f_1, f_2, f_2, f_3, f_4, f_3, f_4}*n = 9*n

X-AKA Protocol: Phase-1: f_1, f_1, f_x, f_1, f_1 = 5

Phase-2: (for n values) {f_1, f_1, f_2, f_2, f_3, f_4, f_3, f_4}*n = 8*n

Total functions used = 5+8*n

EXT-AKA Protocol: Phase-1: f_1, f_{99}, f_1, f_1 = 4

Phase-2: (for n values) {f_1, f_{99}, f_1, f_1, f_2, f_2, f_3, f_4, f_3, f_4}*n = 10

Total functions used= Phase-1 + (Phase-2)*n = 4+10*n

ES-AKA Protocol: Phase-1: f_1, f_1, f_1, f_3, f_4, f_6, f_3, f_4, f_6 = 9

Phase-2: (for n values) {f_1, f_1, f_1, f_2, f_2, f_3, f_4, f_3, f_4}*n = 9

Total functions used = 9+9*n

In Figure 4.4, a graph is plotted between the number of MS and the number of transmitted bits in order to evaluate the computation overhead of each AKA protocol. From Figure 4.4, it is clear that ES-AKA protocol generates less computation overhead than UMTS-AKA, EXT-AKA, COCKTAIL-AKA, and S-AKA but not X-AKA and AP-AKA. Since, the communication and computation overheads are some sort of trade-off, which means that if one computes more at MS or HLR, then the transmitted information in the sent messages can

Table 4.4: Bandwidth utilization of various protocols

No. of AVs	AP-AKA/ UMTS-AKA	X-AKA/ UMTS-AKA	EXT-AKA/ UMTS-AKA	EURA-SIP-AKA/ UMTS-AKA	COCK-TAIL-AKA/ UMTS-AKA	S-AKA/ UMTS-AKA	ES-AKA/ UMTS-AKA
5	1.09	0.6	0.62	0.98	0.66	0.93	0.54
10	1.04	0.52	0.54	1	0.56	0.82	0.44
50	1	0.46	0.46	1.02	0.47	0.73	0.36
100	0.99	0.45	0.45	1.03	0.46	0.72	0.35
200	0.99	0.45	0.45	1.03	0.45	0.71	0.35
500	0.99	0.45	0.45	1.03	0.45	0.71	0.34
1000	0.99	0.45	0.45	1.03	0.45	0.7	0.34
Avg.	1.01	0.48	0.48	1.01	0.5	0.76	0.38

Table 4.5: Reduction ratio for message exchange

No. of AVs	AP-AKA/ UMTS-AKA	X-AKA/ UMTS-AKA	EXT-AKA/ UMTS-AKA	EURA-SIP-AKA/ UMTS-AKA	COCK-TAIL-AKA/ UMTS-AKA	S-AKA/ UMTS-AKA	ES-AKA/ UMTS-AKA
5	0.86	0.9	1.08	0.9	1.16	1.12	1.08
10	0.83	0.85	1.04	0.9	1.03	1.06	0.99
50	0.8	0.81	1	0.9	0.92	0.93	0.91
100	0.8	0.8	1	0.9	0.91	0.91	0.9
200	0.8	0.8	1	0.9	0.9	0.9	0.9
500	0.8	0.8	1	0.9	0.9	0.9	0.9
1000	0.8	0.8	1	0.9	0.9	0.9	0.9
Avg.	0.81	0.82	1.01	0.9	0.96	0.96	0.94

be reduced, and vice versa. Since, the ES-AKA protocol provides lesser communication overhead, thus, it may not be the case that it could also provide lower computation overhead. The X-AKA and AP-AKA used few cryptographic functions, thus, provides less computation overhead than ES-AKA, however, both generate large communication overhead (both require more number of transmitted bits during the run of their protocols) in comparison to the ES-AKA protocol.

4.5.5 Bandwidth Consumption

To analyze the bandwidth consumption in each AKA protocol, we assume that n-authentication vectors are transmitted from the VLR/HLR to the MS. The

Table 4.6: Protocols vs. different parameters

Parameters	UMTS-AKA	AP-AKA	X-AKA	EXT-AKA	EURASIP-AKA	COCKTAIL-AKA	S-AKA	ES-AKA
Mutual Authentication	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
User Traffic Confidentiality	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Signaling Data Integrity	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Reduction in Bandwidth	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Reduction of VLR Storage	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Solve Synchronization	No	Yes	Yes	Yes	Yes	No	Yes	Yes
Communication Overhead	256+960*n	768+952*n	880+432*n	1008+432*n	992*n	1200+432*n	1312+680*n	1080+332*n
Computation Overhead	10*n	3+8*n	5+8*n	4+10*n	9*n	13+9*n	16+9*n	9+9*n

bandwidth consumption in each AKA protocol varies with the number of AVs transmitted from the HLR to the VLR. Since, the total communication bits required in each protocol (for both phases, where second phase can be executed n times) are known, it is easy to calculate bandwidth consumption based on number of authentication requests (n), when n = 5, 10, 50, 100, 200, 500, 1000 (assumption). Now, we calculate the consumed bandwidth for each protocol and then compare the bandwidth consumption of ES-AKA with all other AKA protocols. Similarly, the computation result of each protocol are known, it is easy to calculate total message exchanged (in terms of cryptographic functions′ computation) during each protocol execution by considering n = 5, 10, 50, 100, 200, 500, 1000.

Table 4.4 represents the total bandwidth consumption by each AKA protocol, when the number of authentications within the same VLR (n) = 5, 10, 50, 100, 200, 500, 1000. One can clearly observe that on an average, the ES-AKA protocol lowers 62% of the total bandwidth, which is the utmost reduction of bandwidth by any AKA protocol from the literature. Similarly, Table 4.5 shows that the ES-AKA reduces 6% of the messages exchanged ratio (average of message exchanged ratios of proposed protocol/UMTS-AKA) during authentication, when n = 5, 10, 50,100, 200, 500, 1000.

4.5.6 Summary of Comparative Analysis

This section represents the summary of comparative analysis of all AKA protocols. The analysis of different protocols with respect to fulfillment of various security requirements is presented in Table 4.6.

1. *Mutual Authentication* (HLR authenticates the MS and vice versa), *User Traffic Confidentiality* (fresh generation of CK key) and *Signaling Data Integrity* (fresh generation of IK key) are provided by all existing and proposed AKA protocols.

2. *Reduction of Bandwidth Consumption between the VLR and the HLR:* The n-sets of authentication vectors are transmitted in UMTS-AKA and AP-AKA from the HLR to the VLR. The X-AKA and COCKTAIL-AKA protocols overcome the problem of bandwidth consumption by sending a

temporary key and one-set of PAV to the VLR. Once the HLR authenticates the MS successfully, it sends a DK key to the visited VLR for the subsequent authentications. Such a scheme results in the traffic reduction between the HLR and the VLR, which lowers the bandwidth consumption. Similar to the concept in [29], the ES-AKA protocol also reduces the bandwidth consumption between the HLR and the VLR by using a delegation key, *i.e.*, DK.

3. *Reduction of VLR Storage:* The VLR needs a larger space to store n-sets of authentication vectors transmitted from the HLR to the VLR in the UMTS-AKA and AP-AKA protocols. However, the X-AKA and COCKTAIL-AKA protocols do not need it, and only TK and PAV are required to store at VLR [29]. In ES-AKA, the VLR uses one-time T_1, DK, and TAV to authenticate the MS for the subsequent authentications and hence, reduces the storage space at VLR.

4. *Solve Synchronization Problem:* The AP-AKA and X-AKA protocols use different methods to solve this problem, except UMTS-AKA, however, the COCKTAIL-AKA does not solve this problem [34]. The ES-AKA protocol is also free from the synchronization problem.

4.6 Simulation of ES-AKA Protocol

This section presents simulated results of functions that are used in the ES-AKA protocol, total execution time of ES-AKA, and the computation results of a modified algorithm, *i.e.*, MES-128.

4.6.1 Functions Computation and Execution Time Estimation

Since, it is very hard to have an open source software for UMTS network, where developers/researchers can modify the actaul protocol stack of UMTS-AKA, thus, ES-AKA protocol is simulated in Java environment. The protocol has considered as client server paradigm where MS is a client and, VLR and HLR are servers. The functions $f'()$ and $f_1()$ are implemented as function of XOR and

HMACSHA1 respectively. However, we recommend to use complex function of XOR. Further, functions $f_2()$, $f_3()$, $f_4()$, and $f_6()$ are considered as HMAC-SHA256. However, all these functions are network operator specific. The output of $f'()$ is truncated to finally 24 bits as the output of this functions is ACC_3 of 24 bits in size. Similarly, the output of $f_1()/f_2()$, and $f_3()/f_4()/f_6()$ are truncated to 64 and 128 bits respectively. In all tables, the unit of time is milliseconds and memory/space is measured in bytes. Please note that the total execution time for the ES-AKA protocol is the time to transmit all messages in the network plus the computation time to perform various functions at MS, VLR, and HLR. Since, we simulated this protocol in Java environment on a PC, thus, the message transmission time observed on PC will be different as compared to run on real UMTS network, hence, the transmission time is calculated according to speed of UMTS network. However, the computation time for each function will be almost same on both platforms. The data speed of UMTS network is considered from 384 kbps to 2 Mbps. Thus, the ES-AKA is simulated and the transmission time for each message is calculated with a speed of 384 kbps as well as 2 Mbps. Here, Ext = Execution Time (milliseconds), PCPUT = Process CPU Time (milliseconds), TUM = Total Used Memory (bytes)

ES-AKA Protocol (384 kbps): Total ES-AKA messages transmission time = (1) IMSI, ACC_1 + (2) ACC_2 + (3) ACC_3, MAC_{ms} + (4) IMSI, ACC_3, MAC_{ms}, LAI + (5) MAC_h, T_1, AMF, PCK, PIK, DK + (6) MAC_s, T_2, T_1, AMF, count + (7) RES = 386536 + 61032+ 223784 + 651008 + 1424080 + 681524 + 162752 = 3590716 nanoseconds = 3.5 milliseconds

ES-AKA Protocol (2 Mbps): Total ES-AKA messages transmission time = (1) IMSI, ACC_1 + (2) ACC_2 + (3) ACC_3, MAC_{ms} + (4) IMSI, ACC_3, MAC_{ms}, LAI + (5) MAC_h, T_1, AMF, PCK, PIK, DK + (6) MAC_s, T_2, T_1, AMF, count + (7) RES = 72352 + 11424 + 41888 + 121856 + 266560 + 127568+ 30464 = 672112 nanoseconds = 0.6 milliseconds

Table 4.7: Computations for each function used in ES-AKA protocol

$f'()$			$f_1()$			$f_2()/f_3()/f_4()/f_6()$		
ExT	PCPUT	TUM	ExT	PCPUT	TUM	ExT	PCPUT	TUM
0.84	93.60	12968288	221.60	296.40	15211840	273.41	296.40	15204024

The execution time, process CPU time, and total memory usage for $f'()$, $f_1()$, and $f_2()/f_3()/f_4()/f_6()$ can be observed from Table 4.7. Since, the transmission time for all sent messages along with the computation time for each function that is used in the ES-AKA protocol are known, thus, it is easy to calculate the total time required for the execution of the ES-AKA protocol.

Total execution time for ES-AKA = Total messages transmission time + 2*f′() + 2*$f_1()$ for MAC_{ms} + 2*$f_1()$ for MAC_h + 2*$f_1()$ for MAC_s + 2*$f_3()$ for PCK + 2*$f_4()$ for PIK + 2*$f_6()$ for DK +2*$f_3()$ for CK + 2*$f_4()$ for IK + 2*$f_2()$ for RES

Total execution time for ES-AKA Protocol (384 kbps) = 3.5 + 2*0.84 + 2*221.60 + 2*221.60 + 2*221.60 + 2* 273.41 + 2* 273.41 + 2* 273.41 + 2* 273.41 + 2* 273.41 +2* 273.41 = 3.5 + 156 + 443.20 + 443.20 + 443.20 + 546.82 + 546.82 + 546.82 + 546.82 + 546.82 + 546.82 = 4459.70 milliseconds

Total execution time for ES-AKA protocol (2 Mbps) = 0.6 + 2*0.84 + 2*221.60 + 2*221.60 + 2*221.60 + 2* 273.41 + 2* 273.41 + 2* 273.41 + 2* 273.41 + 2* 273.41 +2* 273.41 = 4456.80 milliseconds

Since, TCK, TIK, and TEK are generated by XOR functions, thus their generation time can be neglected. Similarly, the time estimation for each subsequent authentication request during the session can be calculated as:

Subsequent Authentications (Phase-2: Step 6 and 7):

Total execution time for ES-AKA = Total messages transmission time (during Phase 6 and 7) + $f_1()$ for MAC_h + 2*$f_1()$ for MAC_s + 2*$f_3()$ for CK + 2*$f_4()$ for IK + 2*$f_2()$ for RES

Total execution time for ES-AKA phase-2 protocol (384 kbps) = 0.84+ 221.60 + 2*221.60+ 2* 273.41 + 2* 273.41 + 2* 273.41 = 2306.10 milliseconds

Total execution time for ES-AKA phase-2 protocol (2 Mbps) = 0.15 + 221.60 + 2*221.60+ 2* 273.41 + 2* 273.41 + 2* 273.41 = 2305.41 milliseconds

Since, the execution time for f′() is 0.84 milliseconds, and the transmission time for message (3) from the MS to the VLR is 0.04 milliseconds and 0.22 milliseconds with the 3G speed of 2Mbps and 384 kbps respectively, thus, on an average, one can calculate the timeout period = 0.84 + 0.04 = 0.88 milliseconds or 0.84 + 0.22 = 1.06 milliseconds. This timeout period helps to prevent the flood of authentication requests-based DoS attack.

4.6.2 Improved Secure Encryption Algorithm with ETK Key

Cryptographically weak GSM cipher algorithms A5/1 and A5/2 are replaced by A5/3 algorithm, which is based on the block cipher called KASUMI [85]. The KASUMI is a modified version of the MISTY cryptosystem [86]. The related key attack on full KASUMI algorithm can be simulated within two hours [87]. Nowadays, AES cipher is considered one of the secure algorithms and has 128-bit block size with key size of 128, 192, and 256 bits. It is noticed that going from 128 bits key to 192 bits causes increase in power and time consumption about 8% and for 256 bit key causes an increase of 16% [88]. Biryukov A. et al. [89] describe several attacks, which can break with practical complexity variants of AES-256, while with the best of our knowledge, neither AES-128 nor AES-256 have been directly broken till now. Further, it is extensively believed that ignoring the last round MixColumns of AES has no security implications, thus, many attacks are based on reduced-round variants of AES (free from the MixColumns). However, researchers in [90] refused this faith and presented that the exclusion of MixColumn influences the security of (reduced-round) AES. We have modified the original AES, and implemented Modified Advance Encryption Algorithm-128 bits (MAES-128) along with original AES and KASUMI algorithms.

A1	C7	F2	8E
C3	6D	3A	4A
F5	89	45	64
DE	F7	E4	B1

21	AB	C3	12
5D	F3	24	68
1E	BC	7C	A2
53	A6	5E	6D

21	AB	C3	12
F3	24	68	5D
7C	A2	1E	BC
6D	53	A6	5E

. **Round, SubByte, and ShiftRow in AES Algorithm**

A1	C7	F2	8E
C3	6D	3A	4A
F5	89	45	64
DE	F7	E4	B1

A1	C7	F2	8E
6D	3A	4A	C3
45	64	F5	89
B1	DE	F7	E4

21	AB	C3	12
F3	24	68	5D
7C	A2	1E	BC
6D	53	A6	5E

. **Round, ShiftRow, and SubByte in MAES-128 Algorithm**

$$M = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \qquad M^{-1} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \tag{4.1}$$

$$M_1 = M_1^{-1} = \begin{bmatrix} 01 & 02 & 04 & 06 \\ 02 & 01 & 06 & 04 \\ 04 & 06 & 01 & 02 \\ 06 & 04 & 02 & 01 \end{bmatrix} \tag{4.2}$$

Table 4.8: KASUMI vs. AES: key set time (milliseconds)

KASUMI				AES			
SKT	UPMS	USSS	CPUT	SKT	UPMS	USSS	CPUT
0.04	11090079	16912814	76.44	0.08	11671695	17547345	78.00

Table 4.9: MAES-128 vs. AES key set time (milliseconds)

MAES-128				AES			
SKT	UPMS	USSS	CPUT	SKT	UPMS	USSS	CPUT
0.06	11038857	16606650	78.00	0.08	11671695	17547345	78.00

Table 4.10: AES: encryption and decryption time (milliseconds)

AES Encryption				AES Decryption			
ET	UPMS	USSS	CPUT	DT	UPMS	USSS	CPUT
0.51	11671908	17549778	79.56	0.49	11672051	17550815	79.56

This algorithm is proposed to implement on mobile SIM as well as at VLR/ HLR. This MAES-128 with ETK key is used to encrypt the transmitted message during the authentication process of ES-AKA. For the best assembly code combinations, the order of SubByte and ShiftRow can be exchanged in order to minimize the number of times reads and writes in the memory, and boost the computation rate without compromising the actual outcome [91]. Hence, the order of SubByte and ShiftRow are swapped in the MAES-128 algorithm. In AES, the MixColumns step is defined as a multiplication of the columns with

Table 4.11: KASUMI: encryption and decryption time (milliseconds)

KASUMI Encryption				KASUMI Decryption			
ET	UPMS	USSS	CPUT	DT	UPMS	USSS	CPUT
1.26	11097403	16912912	76.44	3.79	11097477	16912912	76.44

Table 4.12: MAES-128: encryption and decryption time (milliseconds)

MAES-128 Encryption				MAES-128 Decryption			
ET	UPMS	USSS	CPUT	DT	UPMS	USSS	CPUT
0.27	11508939	16606650	78.00	0.25	11038989	16606665	78.00

the matrix M. The matrix M and its inverse matrix M^{-1} used in AES, are different. We propose an alternative matrix M_1 where this new matrix $\mathrm{M}_1 = \mathrm{M}_1{}^{-1}$. The generation of the inverse of the matrix is not required, hence it maintains the computation. To find an efficient approach, it is required to optimize the memory usage during key generation and setup. Thus, all round keys are stored in the memory until they are revoked (within the expiry time of session), the keys can be extracted in a sequence without regenerate them.

The time elapsed to encrypt and decrypt the SMS are measured separately. The execution time is measured by System.currentTimeMillis(), which is available in Java environment. A fixed 128-bit data is passed as input to AES, KASUMI and MAES-128, and various computations with these algorithms have been observed. Various parameters such as set key time, physical memory size, swap space size, CPU time, encryption time, and decryption time are considered with accumulating the results mentioned in Table 4.8, 4.9, 4.10, 4.11, and 4.12. Table 4.8 and Table 4.9 show the key set time for KASUMI, AES, and MAES-128. Out of these algorithms, KASUMI takes less time for setting up key. Table 4.10, Table 4.11, and Table 4.12 conclude the execution time in encryption and decryption of AES, KASUMI, and MAES-128 respectively, and out of which MAES-128 takes lesser time than other algorithms. The CPU time for key setup is same for all algorithms, and KASUMI takes less CPU time for encryption and decryption. KASUMI algorithm utilizes less physical as well as swap space memory for all operations. Since, the ES-AKA protocol generates its own ETK key during its execution, thus, it does not require to generate the first key (only needs to generate subsequent keys for rounds), which saves the key set time and CPU time for setting up the key. Thus, MAES-128 is the best cipher algorithm for the ES-AKA protocol as it requires lower time to cipher and decipher the transmitted messages.

4.6.3 Formal Proof of ES-AKA Protocol

We use the BAN-Logic symbols to formally proof the authentication process of ES-AKA protocol. Various notations used in BAN-Logic are described in section 3.7.4.

1. *The Formal Messages in ES-AKA Protocol:*

   (1) MS → VLR: ID, Na

   (2) VLR → MS: Nb

   (3) MS → VLR: Nc, f_1(Nc, LAI)$_{SK}$, MS $\overset{SK}{\leftrightarrow}$ HLR

   (4) VLR → HLR: ID, Nc, f_1(Nc, LAI)$_{SK}$

   (5) HLR → VLR: Ta, f_1(Ta, AMF)$_{SK}$, VLR $\overset{DK}{\leftrightarrow}$ HLR, VLR $\overset{PCK}{\leftrightarrow}$ HLR, VLR $\overset{PIK}{\leftrightarrow}$ HLR, f_3(Nc)$_{SK}$, f_4(Nc)$_{SK}$, f_6(Nc)$_{SK}$

   (6) VLR → MS: Ta, Tb, f_1(f_1(Ta, AMF)$_{SK}$ Ta, Tb, Td)$_{DK}$, f_3(Tb)$_{DK}$, f_4(Tb)$_{DK}$

   (7) MS → VLR: f_2(Tb)$_{DK}$, TCK = PCK $\oplus$ CK, TIK = PIK $\oplus$ IK, TEK = CK $\oplus$ IK, MS $\overset{TCK}{\leftrightarrow}$ VLR, MS $\overset{TIK}{\leftrightarrow}$ VLR, MS $\overset{TEK}{\leftrightarrow}$ VLR

2. *Security Assumptions:*

   a) It is assumed that SK key is shared between the MS and its HLR.

   (1) MS has the secure key SK and MS $|\equiv$ MS $\overset{SK}{\leftrightarrow}$ HLR

   (2) HLR has the secure key SK and HLR $|\equiv$ MS $\overset{SK}{\leftrightarrow}$ HLR

   b) It is assumed that the VLR trusts the HLR.

   (1) VLR $|\equiv$ HLR $|\Rightarrow$ MS $\overset{SK}{\leftrightarrow}$ HLR,

   (2) $\frac{VLR \mid\!\sim P, HLR \triangleleft P}{HLR|\equiv VLR|\equiv P}$

   (3) $\frac{HLR \mid\!\sim P, VLR \triangleleft P}{VLR|\equiv HLR|\equiv P}$

   c) It is assumed that communication between the HLR and the VLR is secure.

   (1) VLR $|\equiv$ VLR $\overset{P}{\Leftrightarrow}$ HLR, P is conveyance message between the VLR and the HLR.

   (2) HLR $|\equiv$ VLR $\overset{P}{\Leftrightarrow}$ HLR, P is conveyance message between the VLR and the HLR.

3. *Security Analysis:*

   (1) MS → VLR: MS $|\equiv$ #(Na), VLR $\lhd$ Na, ID

   (2) VLR → MS: VLR $|\equiv$ #(Nb)

   (3) MS → VLR: MS $|\equiv$ #(Nc) $\wedge$ VLR $|\equiv$ #(Nc), derive Nc = f′(Na, Nb) and compare with the received Nc; VLR $\lhd$ Nc, $\mathrm{f_1(Nc, LAI)}_{SK}$

   (4) VLR → HLR: HLR $\lhd$ Nc, $\mathrm{f_1(Nc, LAI)}_{SK}$, on receiving derive $\mathrm{f_1(Nc, LAI)}_{SK}$ and compared with $\mathrm{f_1(Nc, LAI)}_{SK}$

   (5) HLR → VLR: VLR $\lhd$ Ta, AMF, $\mathrm{f_1(Ta, AMF)}_{SK}$,HLR $|\equiv \forall$ (VLR $\overset{DK}{\leftrightarrow}$ MS)

   (6) VLR → MS: MS $|\equiv$ #(Tb) $\wedge$ VLR $|\equiv$ #(Tb), MS $\lhd$ Ta,Tb, $\mathrm{f_1(f_1(Ta, AMF)}_{SK}\mathrm{, Ta, Tb, Td)}_{DK}$

   (7) MS → VLR: VLR $\lhd$ $\mathrm{f_2(Tb)}_{DK}$, on receiving derives $\mathrm{f_2(Tb)}_{DK}$ and compares with the received $\mathrm{f_2(Tb)}_{DK}$, HLR $|\equiv \forall$ (MS $\overset{TCK}{\leftrightarrow}$ VLR), HLR $|\equiv \forall$ (MS $\overset{TIK}{\leftrightarrow}$ VLR), HLR $|\equiv \forall$ (MS $\overset{TEK}{\leftrightarrow}$ VLR)

4. *Message Meaning Rule:*

   (1) $\frac{MS|\equiv(MS\overset{SK}{\leftrightarrow}HLR)\wedge(VLR\overset{DK}{\leftrightarrow}MS),MS\lhd f_1(Nc,LAI)_{SK}}{MS|\equiv HLR\upharpoonright f_1(Nc,LAI)_{SK}}$

   (2) $\frac{VLR|\equiv f_1(Ta,AMF)_{SK}\wedge(VLR\overset{DK}{\leftrightarrow}MS),VLR\lhd f_1(f_1(Ta,AMF)_{SK},Ta,Tb,Td)_{DK}}{VLR|\equiv HLR\upharpoonright f_1(f_1(Ta,AMF)_{SK},Ta,Tb,Td)_{DK}}$

5. *Nonce/Timestamp Verification Rule:*

   (1) $\frac{MS|\equiv\#(Na)\wedge\#(Nc),MS|\equiv HLR\upharpoonright f_1(Nc,LAI)_{SK}}{MS|\equiv HLR|\equiv f_1(Nc,LAI)_{SK}}$

   (2) $\frac{VLR|\equiv\#(Nb)\wedge\#(Nc),VLR|\equiv HLR\upharpoonright f_1(f_1(Ta,AMF)_{SK},Ta,Tb,Td)_{DK}}{MS|\equiv HLR|\equiv f_1(f_1(Ta,AMF)_{SK},Ta,Tb,Td)_{DK}}$

6. *Jurisdiction Rule:*

   (1) $\frac{MS|\equiv HLR\Rightarrow f_1(Nc,LAI)_{SK},MS\lhd VLR\upharpoonright f_1(Nc,LAI)_{SK}}{MS|\equiv VLR|\equiv HLR}$

   (2) $\frac{VLR|\equiv HLR\Rightarrow f_1(f_1(Ta,AMF)_{SK},Ta,Tb,Td)_{DK},VLR\lhd VLR\upharpoonright f_1(f_1(Ta,AMF)_{SK},Ta,Tb,Td)_{DK}}{VLR|\equiv HLR|\equiv MS}$

7. *Protocol Goals:*

   a) Mutual authentication between the MS and the VLR

   b) Key agreement between the MS and the VLR

   c) Key freshness between the MS and the VLR

   d) Confidentiality between the MS and the VLR

   e) Resistance replay attack between the MS and the VLR

   f) Resistance man-in-the-middle Attack between the MS and the VLR

   g) Resistance redirection attack between the MS and the VLR

   h) Resistance impersonation attack (corrupt network) between the MS and the VLR

i) Resistance DoS attack between the MS and the VLR

a) Mutual authentication between the MS and the VLR:

MS $|\equiv$ HLR $|\equiv$ VLR $\wedge$ VLR $|\equiv$ HLR $|\equiv$ MS $\rightarrow$ MS $|\equiv$ VLR $\wedge$ VLR $|\equiv$ MS. Thus, mutual authentication holds.

b) Key agreement between the MS and the VLR:

There is a DK key between the VLR and the MS to provide agreement.

MS $|\equiv$ DK $\wedge$ #(Tb), since DK = $\mathrm{f}_6(\mathrm{Tb})_{SK}$

VLR $|\equiv$ DK $\wedge$ #(Tb), since HLR $\rightarrow$ VLR $\upharpoonleft$ DK

c) Key freshness between the MS and the VLR:

HLR $|\equiv$ #(Ta) $\wedge$ VLR $|\equiv$ #(Ta), VLR $|\equiv$ #(Tb) $\wedge$ MS $|\equiv$ #(Tb), DK = $\mathrm{f}_6(\mathrm{Tb})_{SK}$, Thus key freshness between the MS and the VLR holds.

d) Confidentiality between the MS and the VLR:

$\frac{MS|\equiv(MS\overset{TEK}{\leftrightarrow}VLR), MS\triangleleft f_7(Msg)_{TEK}}{MS|\equiv VLR\upharpoonleft Msg} \wedge \frac{VLR|\equiv(VLR\overset{TEK}{\leftrightarrow}MS), VLR\triangleleft f_7(Msg)_{TEK}}{VLR|\equiv MS\upharpoonleft Msg}$;

e) Resistance replay attack between the MS and the VLR:

If the attacker gets #Na from message (1) or #Nb from message (2), he/she cannot forge message (1) and (2) because he/she does not know how to derive #Nc. If the attacker gets Nc from message (3) or (4), he/she cannot forge message because he/she does not know SK key. If the attacker gets Ta from message (5), he/she is unable to forge message (5) and (6) because he/she does not know DK key. If the attacker gets Tb from message (6), he/she is unable to forge message (6) and (7) because he/she also does not know DK key. Since, Tb will be changed at the next time hence, the goal of resistance replay attack between the MS and the VLR holds.

f) Resistance man-in-the-middle attack between the MS and the VLR:

Since, the attacker knows neither TEK key nor f_7 encryption algorithm, thus, it prevents the communication from being eavesdropped.

g) Resistance redirection attack between the MS and the VLR:

Since, $\mathrm{f}_1(\mathrm{Nc, LAI})_{SK}$ is used to maintain the integrity of LAI, thus, it prevents from the redirection attack.

h) Resistance impersonation attack between the MS and the VLR:

(1) Adversary tries to impersonate the MS: Since, $\mathrm{f}_1(\mathrm{Nc, LAI})_{SK}$ is computed at MS and compared at HLR, this prevents the impersonation at-

tack. Additionally, adversary must reply with a valid $f_2(Tb)_{DK}$ but, he/she neither has DK or TEK key, where $f_6(Tb)_{SK}$ and TEK = CK ⊕ IK.
(2) Adversary tries to impersonate the VLR: The integrity value f_1(Nc, LAI)$_{SK}$ at MS and at HLR will be violated. Additionally, if the MS receives $f_1(f_1(\text{Ta, AMF})_{SK}, \text{Ta, Tb, Td})_{DK}$ at any time, then the connection will be terminated because the MS has not sent any request to the VLR.
i) Resistance DoS attack between the MS and the VLR: Here, MS $|\equiv \#(\text{Na}) \wedge \#(\text{Nc})$ and VLR $|\equiv \#(\text{Nb}) \wedge \#(\text{Nc})$, the VLR checks timeout and Nc = f′(Na, Nb), which can mitigate the DoS attack. When an attacker will perform DoS attack, the VLR will not receive the correct Nc.

4.7 More Focus on Attack Model

More on DoS Attack: There are some other reasons for the possibilities of DoS attack such as black-hole attack and dropping ACK signal.
Black-hole attack: An intruder with false BTS equipment follows close to its target victim. In the presence of this attack, all the active mobile terminals in that area are diddled towards the false BTS for the connection, if the signal from the malicious BTS is stronger than the legitimate BTS. When the victim is connected to the fake BTS, the intruder drops each transmitted packet towards the MS. This scenario of black hole is visualized as the radio jamming.

Table 4.13: Definitions of functions used in Secure-AKA protocol

Function	Definition
f′	Function to generate TID
f″	Function to generate IIK
f_1	Function to generate MAC/XMAC
f_2	Key generation function for DK
f_3	Function to generate RES/XRES
f_4	Key generation function for CK
f_5	Key generation function for IK
f_6	Key generation function for EK
g()	Hash generation function for α
H	Hash function
Solu_1	Function to compute Solution_1
Solu_2	Function to compute Solution_2
EK{}, DK{}	Function to cipher/decipher message
||	Concatenation

Dropping ACK Signal: The protection of IMSI is considered a very important issue in the UMTS network. Instead, the temporary identity TMSI is transmitted to the MS just after the activation of cipher mode, and then the TMSIs are used for the signaling communication in the network. A new TMSI is allotted each time a mobile user moves to a roaming area observed by other SGSN. When a TMSI is encrypted and transmitted to the MS, it does not link to corresponding IMSI by the SGSN until allocation complete message is reached to the SGSN from the MS. If it is not the case, then, both sets {IMSI, TMSIold} and {IMSI, TMSInew} are believed correct by the SGSN, and TMSI allocation command messages are examined by the attacker, which immediately drops those messages. This process creates a cause where a new TMSI is generated repeatedly, which is expressed as dropping ACK signal-based DoS attack to all users entering in particular routing area.

4.8 A Solution: Tour Puzzle for DoS Attack

This section focuses on DoS attack that may exist in the UMTS network, whereas the prevention from other attacks are discussed in the later section. A tour puzzle scheme is presented, which prevents the UMTS network from the DoS attack. Various cryptographic puzzle schemes have been introduced as a defense mechanism to prevent the DoS attack, however, the proposed scheme overcomes the various limitations of existing tour puzzle schemes. Table 4.13 defines various functions used in the proposed protocol.

4.8.1 Security Requirements for a Puzzle Scheme

A cryptographic puzzle scheme should satisfy certain requirements in order to qualify as a better defense mechanism [92]. The primary attribute of a client puzzle is that it should be neither impossible nor too easy to solve. Several cryptographic puzzles are based on the inversion of hash function [93], [94]. It is important to note that the puzzle generation and its solution verification must produce minimal computation and memory overhead at the server. Otherwise, there may be a chance that the puzzle mechanism itself becomes a reason for the DoS attack, when an attacker transmits bulky fake requests to generate puzzles or their solutions. Additionally, a protocol with strong client puzzle scheme may not be secure enough against DoS attack, if the required security is not provided to the puzzle. Another important point is that before a server wishes to perform some high computation operations, it must ensure that the client has committed some of its own resources for solving a puzzle. The initial model for client puzzles was described by Jules et al. [94]. Then after, Rivest R. L. et al. [95] proposed a RSA modulus factorization-based modular exponentiation puzzle. However, these puzzles require a demanding server to execute computationally intensive exponentiation to verify their solutions. More recently, Stebila D. et al. [96] proposed a security model for DoS resistance of key exchange protocols. Recently, some other client puzzle-based schemes have been proposed to overcome DoS attack [97], [98]. The following are the requirements for a puzzle scheme:

1. *Computation Assurance:* The puzzle should be able to provide computation assurance in terms of the number of cryptographic operations carried out by the client to solve the puzzle. The puzzle should be hard enough for the malicious user to solve in reasonable time [93].

2. *Efficiency:* The performance efficiency of server in terms of speed, storage capacity, and bandwidth must be high for puzzle generation, distribution, and client verification. All operations carried out must add minimal server overhead and be good enough to tackle DoS attack [99].

3. *Puzzle Granularity:* Puzzle granularity ensures that the degree of difficulty can be increased or decreased based on the assessment to observe the behavior of a user.

4. *Correlation-free:* Applying inference knowledge from other puzzle solution should not strong enough to easily solve the new puzzle.

5. *Tamper-resistance:* The puzzle scheme should be able to resist the replay attack over time. One time puzzle generation can be used to avoid reusability of the answer of a puzzle at multiple clients.

6. *Non-parallelizability:* A strong coupling between puzzle modules limits its parallel implementation. This prevents parallel computation by malicious client for puzzle solution.

7. *Puzzle fairness:* An approximate time must be spent in solving the puzzle by authentic client irrespective of resources used. This property helps in preventing the network from DoS attack to an acceptable level.

The proposed scheme stresses on tour-based puzzle scheme because these puzzles require a minimum amount of time to solve them. It is considered that the size of puzzle and its solution are of 128 bits.

4.8.2 Drawbacks in the Existing Tour Puzzle Schemes

The existing tour puzzles are the efficient solutions against both, the resource consumption DoS attack and the protocol related DoS attacks, but, still they have vulnerabilities that can be exploited to increase the load on server even more. These types of abnormal requests generated by the clients can cause the computation in the puzzle itself to be done repeatedly and cause stress on server. Some of the major drawbacks found in the existing tour puzzle scheme are as follows:

1. *Multiple Requests for the Puzzle by a Client:* In the tour puzzle [92], the server replies to all clients with a service restricted message, indicating that a puzzle needs to be solved before receiving a service, when the server is under attack. The server also allows a single client to experience different tours (n_x) by providing the server with different n_x without performing any other check before giving a tour. However, it may cause congestion or heavy load on the server, if multiple clients send their requests to the server for resource allocation (DDoS attack). Thus, we find that an exhausted server′s condition can be aggravated with sending fake requests by the malicious clients.

2. *No Communication between Server and Nodes:* In the tour puzzle [92], after receiving a puzzle from the server, the client traverses each node and reaches at last node from where it sends the final puzzle answer to the server. A legitimate client follows the correct path of execution; however, a malicious client can request a puzzle and sends an answer after visiting only few nodes in the guided tour or even after visiting the first node. Then the server validates the outcome sent by the client that must be correct

in order to proceed further to calculate more outcomes, however, it can find one of answers to be false. But, by the time the server realizes that the answer was false and sufficient resources are spent by it during this process. The problem can be much more severe, if a client uses different nonce to take new puzzle and sends fake answers to the server only after visiting the first node.

3. *Excessive Computation at Server:* The tour puzzle [92] involves a lot of computation in evaluating the answer sent by the client. A major part of computation performed by the server increases the load on it. The client does not need to do any major calculations but, only waits for the node to give the next address and traverse it in order to complete the puzzle solution.

4.8.3 Improvements in Tour Puzzle

This subsection proposes some improvements in the existing tour puzzle scheme and incorporates them while developing new tour puzzle scheme, which works with the proposed AKA protocol (explained later). The key point in new tour puzzle scheme is that the VLR maintains $\{A_x, C_x, t_x, P_x, i_x\}$ for each MS, where A_x is the address of each MS, C_x is a counter, t_x is the last time access of each A_x, P_x is a random number, and i_x is a flag value. In new scheme, a tour puzzle contains only two fixed nodes, one is VLR and other is proxy connected to the respective VLR. The following are the proposed improvements in the tour puzzle scheme:

1. *Pre-Puzzle Allocation Check:* The first proposition is to place an algorithm before the VLR (server) starts allocating the puzzle and in turn resources to the MS (client). The VLR keeps track of unique individual MS′s A_x and number of times it has requested within a fixed time T by a counter C_x. It also stores t_x to check, if the MS is requesting within predefined time interval T by comparing it to current time. The t_x is updated each time A_x requests the service, and sends answer or its puzzle timeout. If an MS fails to solve puzzle in time or sends incorrect answer, the value of C_x is incremented and thereby, the complexity of puzzle given to that MS is increased exponentially. In case the MS is already given a puzzle and

requests service interim, the current tour is terminated and the MS′s C_x is incremented.

The VLR periodically decrements the counter value after fixed time t_r for each MS, thus, the MS that does not send request rapidly lowers value of C_x, thereby, receiving less difficult puzzles. The frequent requests by an MS increases the value of C_x faster and the MS has to wait longer in order to get easier puzzle. The (t_r * C_x) is the amount of time it takes for C_x value to drop to zero. Once, the value of C_x is dropped to zero, the entry in the database is deleted and memory is freed. However, the VLR must check if the MS is in puzzle phase, then the VLR should wait for the answer or for the timeout of puzzle before deleting the entries from the database. Also if the MS successfully completes the puzzle within given time, then the database entry is deleted.

2. *Randomized Answer:* A random number P_x is associated with every tour puzzle (after solving the first part of puzzle) assigned to an MS. The VLR stores and sends a random value P_x to the proxy, which is used as a parameter to compute the final answer of the puzzle. The MS follows the trail of tour guides and reaches the proxy. The proxy sends P_x to the MS and the MS sends final answer (Solu2′) of given puzzle to the VLR.

3. *Completion of Tour:* The proxy sends a yes/no flag notification to the VLR, which indicates that the A_x has sent the correct solution for phase-1. If the final solution (Solu2′ ?= Solu2) is correct, the VLR changes the value of corresponding MS′s i_x from 0 to 1. The VLR accepts the answer from the MS only if the value of i_x is 1. Afterwards, the MS sends the answer (whether correct or incorrect) to the VLR and the value of i_x is again changed to zero. The same is the case with timeout of puzzle. By doing so, the VLR ensures that the MS is legitimate. This ensures that no MS sends the answer in the middle or halfway through a tour or sends the answer multiple times after visiting the proxy.

4.8.4 Proposed Tour Puzzle Scheme

This subsection presents a new tour puzzle scheme based on the improvements discussed in the previous subsection. Here, t_s is the start time, at which the MS request is generated while t_x is the last time access of each MS. An algorithm for the new tour puzzle scheme presents the behavior of the proposed approach.

4.9 Focus on Proposed Protocol

This section presents a new AKA protocol namely Secure-AKA for the UMTS network. First, the security goals for the proposed protocol are set and then the proposed protocol is explained in detail.

4.9.1 Goals for the Proposed Protocol

In order to maintain the proper security in the UMTS network, it is required to define goals for the proposed protocol that must be fulfilled before the actual design of the protocol. The security and performance goals for the proposed protocol are as follows:

1. The protocol must be able to provide mutual authentication between the MS and the VLR, and between the MS and the HLR.

2. The protocol should reduce the storage overhead at VLR as in UMTS-AKA all the requested authentication vectors from the HLR to the VLR are stored in the storage space of VLR when user is in roaming area.

3. The actual identity of each MS, *i.e.*, IMSI, should be protected from being eavesdropped.

4. The exchange of messages during the authentication process should be reduced in terms of its size.

5. The protocol must effectively use the bandwidth in order to complete the authentication process successfully.

6. The protocol should generate communication and computation overhead as less as possible.

Algorithm 1 Proposed tour puzzle scheme

```
if MS requests a service without answer then
    if $A_x$ entry does not exists in database then
        create database entry ($A_x$, $C_x$= 0, $t_x$=$t_s$, $P_x$=Random number, $i_x$ =0);
        sends a puzzle to MS;
    else
        if $A_x$ entry exists in database then
            if MS is running a tour, i.e., H exists then $C_x$ = $C_x$ +1;
                terminate current connection of $A_x$;
            else
                if MS is not running a tour then
                    if ($t_s$ - $t_x \leq$ T then) $C_x$ = $C_x$ +1; send a new puzzle;
                    else
                        if ($t_s$ - $t_x >$ T) then send a new puzzle;
                        end if
                    end if
                end if
            end if
        end if
    end if
else
    if MS requests service with answer then
        if $A_x$ entry does not exists in database then ignore;
        else
            if $A_x$ entry exists in database then
                if ($i_x$ == 0) then ignore;
                else
                    if ($i_x$ == 1) then verify the answer (H, $\alpha$) for MS at Proxy;
                        if answer is correct then send a number $P_x$ to MS;
                            if Solu2$'$ is correct (Solu$_2'$ ?= Solu$_2$) at VLR then
                                allow access to resources;
                                $i_x$ = 0;
                                tour of MS is completed;
                            end if
                        else
                            if answer is invalid then $C_x$ = $C_x$ +1;
                                $i_x$ = 0;
                                terminate tour of MS;
                            end if
                        end if
                    end if
                end if
            end if
        end if
    end if
end if
```

7. The protocol must be able to resist various possible attacks in the UMTS networks.

4.9.2 Proposed Secure-AKA Protocol

This section proposes and presents a more secure authentication and key agreement protocol called Secure-AKA, which prevents the network from various attacks. Similar to ES-AKA, this protocol follows the framework of original 3GPP UMTS-AKA protocol and resolves the synchronization problem between the MS and the HLR. In the Secure-AKA protocol, each MS and its HLR share a SK key that is stored at AuC of the HLR and onto the SIM card of MS at the time of manufacturing. The cryptographic functions f′, f″, f_1, f_2, f_3, f_4, f_5, f_6, g, H, $Solu_1$, $Solu_2$, Ek{}, and DK{} used in this protocol are defined in Table 4.13.

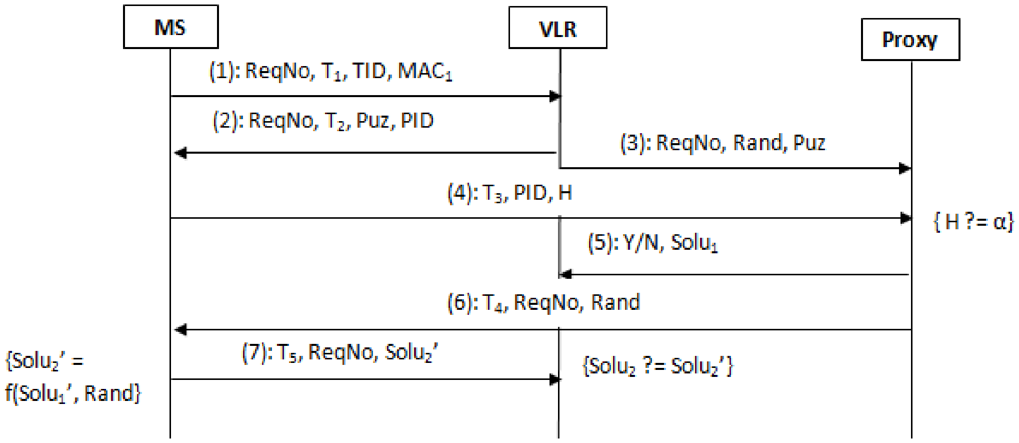


Figure 4.5: Proposed tour puzzle-based approach

The proposed approach based on tour puzzle scheme for DoS prevention has shown in Figure 4.5, which consists of 7 steps. These 7 steps are incorporated in the proposed Secure-AKA protocol, which prevents the UMTS network from various attacks including DoS attack. The Secure-AKA protocol is shown in Figure 4.6. The Secure-AKA protocol is divided in the following two phases:

Phase-1: Initially, the MS who wishes to make a request to the VLR to create a connection for authentication, generates a temporary identity (TID) using f′ function with the original IMSI and a timestamp T_1, where TID=f′(IMSI, T_1). The MS sends (ReqNo, TID, T_1, MAC_1) to the VLR (message (1)), where ReqNo is the request number and $MAC_1 = f_1(T_1, LAI)_{SK}$.

MS to VLR: ReqNo, T_1, *TID,* MAC_1

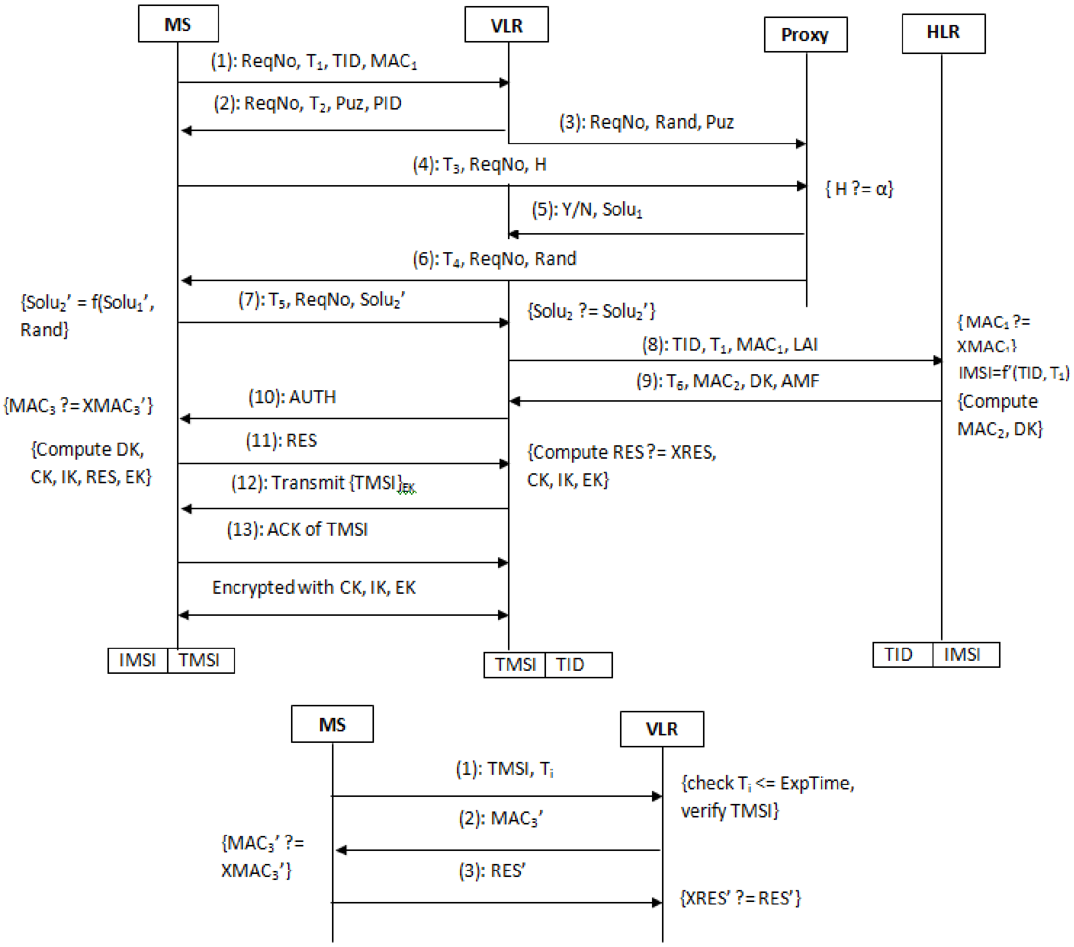


Figure 4.6: Proposed Secure-AKA protocol (a) phase-1, (b) phase-2

Additionally, an integrity key IIK is generated, which prevents the modification of Radio Resource Control (RRC) messages during the communication before the protocol cipher mode is set, where IIK = $f''(T_1)_{SK}$. The steps 2 to 7, shown in Figure 4.6, only execute when the protocol analyzes that there might be DoS attack in the network. The analysis includes the network behavior, number of requests per time, reliability, number of connection failures etc. If it is the case, then the VLR sends a message back to the MS with ReqNo, T_2, a puzzle Puz, and the proxy node identity PID (message (2)).

VLR to MS: ReqNo, T_2, Puz, PID

Simultaneously, the VLR sends ReqNo, a random number Rand, and puzzle Puz to the proxy server/node, which is directly connected to the VLR (message (3)).

VLR to Proxy: ReqNo, Rand, Puz

After receiving the message from the VLR, the proxy waits for the answer from any MS within a timeout period. In the meanwhile the MS sends (T_3, ReqNo, H) to the proxy (message (4)). Here, H is the hash code of the solution generated for the puzzle, *i.e.*, Solu'_1.

MS to Proxy: T_3, ReqNo, H

On receiving the message from the MS, the proxy first compares ReqNo, and if the message is not received within the threshold time, then the request from that MS will be dropped. The proxy sends this notification to the VLR and then the VLR terminates the connection. If all is correct, then the proxy generates the answer of puzzle, *i.e.*, Solu_1, and computes α that is a hash code of Solu_1. The proxy compares α with the received H, if both do not match, then proxy sends a notification message to the VLR with flag value *no* and the connection is released otherwise, proxy sends a notification message to the VLR with *yes* flag with the Solu_1 (message (5)).

Proxy to VLR: Y/N, $Solu_1$

The proxy node waits for a random amount of time before sending the notification to the VLR, only if it finds the duration between the messages received from the VLR and the MS is less than the expected or threshold time duration. Please note that the proxy sends Solu'_1 to the VLR, only if the flag value is *yes* (which means that connection is alive). If the VLR finds the value of flag *yes*, then it sends a message with (T_4, ReqNo, Rand) to the MS (message (6)).

Proxy to MS: T_4, ReqNo, Rand

After receiving the message from the VLR, the MS computes the final solution Solu$'_2$ as Solu$'_2$=f(Solu$'_1$,Rand), where function f is used to provide solution to different puzzle problems. Then, the MS sends (T$'_5$, ReqNo, Solu$'_2$) to the VLR (message (7)).

MS to VLR: T_5, ReqNo, Solu$'_2$

On getting this message, the VLR computes Solu$_2$ and compares it with the received Solu$'_2$. If it does not hold then connection is simply terminated otherwise, the VLR sends (TID, T$_1$, MAC$_1$) to the HLR along with the LAI of MS (message (8)).

VLR to HLR: TID, T_1, MAC_1, LAI

After receiving such information, the HLR derives XMAC$_1$ (XMAC$_1$ = f$_1$(T$_1$, LAI)$_{SK}$ and compares it with the received MAC$_1$. If both are equal then the MS is a legitimate user. The HLR generates IMSI by passing the TID and T$_1$ to the f$'$ function (IMSI = f$'$(TID, T$_1$). The working of function f$'$ is in such a way that if one passes TID to it then he/she receives the output as IMSI, and vice versa. The function can be understood like an XOR operation, however, it is recommended to use a complex function. Then the HLR generates (T$_6$, DK, MAC$_2$, AMF), and passes to the VLR (message (9)) where MAC$_2$ = f$_1$(T$_6$, AMF)$_{SK}$.

HLR to VLR: T_6, DK, MAC_2, AMF

On receiving, the VLR generates T$_7$ and computes MAC$_3$, where MAC$_3$ = f$_1$(MAC$_2$, T$_6$, T$_7$)$_{DK}$. The VLR transmits AUTH to the MS that includes MAC$_3$, T$_6$, T$_7$, and AMF (message (10)).

VLR to MS: MAC_3, T_6, T_7, AMF

On receiving AUTH, the MS generates DK key (DK= f$_2$(T$_1$)$_{SK}$), computes XMAC$_3$ and compares it with the received MAC$_3$ (MAC$_3$?= XMAC$_3$). If it holds, then the MS computes RES, CK, IK, and EK keys, where RES=f$_3$(T$_7$)$_{DK}$, CK=f$_4$(T$_6$)$_{DK}$, IK=f$_5$(T$_6$)$_{DK}$, EK=f$_6$(T$_6$)$_{DK}$. Then, the MS sends RES to the VLR (message (11)).

MS to VLR: RES

On receiving, the VLR computes XRES as XRES=f$_3$(T$_7$)$_{DK}$ and compares it with the received RES (RES ?= XRES). If both are equal, then the VLR generates CK, IK, and EK keys. After the successful mutual authentication, the VLR assigns a new TMSI and sends it to the MS in cipher mode using EK key (message (12)).

VLR to MS: $\{TMSI\}_{EK}$

After receiving the message (12) from the VLR, the MS decrypts TMSI and acknowledges the receipt of TMSI to the VLR (message (13)).

MS to VLR: ACK of TMSI

Here, the size of ACK is of 16 bits. Finally, the MS, VLR, and HLR store the pairs (IMSI, TMSI), (TMSI, TID), and (TID, IMSI) in their storage space respectively. Functions $\text{Solu}_1()$ and $\text{Solu}_2()$ are used to generate the solution_1 (Solu_1) and solution_2 (Solu_2) respectively for the given puzzle. At the completion of authentication process, the TID will be recognized between the VLR and the HLR, while TMSI will be shared between the MS and the VLR.

Phase-2: The MS sends an authentication request to the VLR including the TMSI and current timestamp T_i (message (1)).

MS to VLR: TMSI, T_i

On receiving, the VLR verifies TMSI by checking whether it is stored in the storage space or not. If it is there that means the MS is not requesting for the first time, but, is requesting for the subsequent authentications. The VLR then checks whether $T_i <= \text{ExpTime}$, where ExpTime is the maximum expiry time after that any request for the subsequent authentication is discarded. If both are valid, then the VLR computes MAC'_3 ($\text{MAC}'_3 = f_1(\text{MAC}_2, T_6, T_i)_{DK}$) and sends to the MS (message (2)).

VLR to MS: MAC'_3

Then, the MS computes XMAC'_3 ($\text{XMAC}'_3 = f_1(\text{MAC}_2, T_6, T_i)_{DK}$ and checks whether MAC'_3 ?= XMAC'_3. If it holds then the MS computes RES′ ($\text{RES}' = f_3(T_i)_{DK}$) and sends to the VLR (message (3).

MS to VLR: RES'

On receiving, the VLR computes XRES′ ($\text{XRES}' = f_3(T_i)_{DK}$) and compares it with the received RES′. If both are equal, then the authentication is successful. Now, the MS and the VLR can use CK, IK, and EK keys for the purpose of Confidentiality, Integrity, and Encryption.

The limitation of the Secure-AKA protocol can be considered in terms of storage overhead, since the MS, VLR, and HLR, each needs to store one identity pair as: MS → (IMSI, TMSI), VLR → (TMSI, TID), and HLR → (TID, IMSI).

But in fact, these identity pairs take very less memory space to store them onto the MS, VLR, and HLR.

4.10 Analysis of Secure-AKA Protocol

This section discusses the security and performance analysis of the Secure-AKA protocol in terms of resistance to various attacks, communication and computation overheads, and bandwidth utilization.

4.10.1 Resistance to Attacks

This subsection describes the behavior of Secure-AKA protocol against various possible attacks in the UMTS network.

1. *Redirection Attack and Black-hole Attack:* In Secure-AKA, the message authentication code is used to maintain the integrity of LAI and thereby prevents the UMTS networks from the redirection attack. In the proposed protocol, the MS involves the LAI in MAC_1 and sends MAC_1 to the VLR in message (1). The authentication request is denied, if the HLR fails to match the LAI sent by the VLR in message (8) with the LAI embedded in MAC_1. Such a technique solves the problem of mischarged billing and provides prevention against the black-hole attack in the UMTS networks.

2. *Replay Attack:* The Secure-AKA protocol is free from this attack by sending timestamp T_1, T_2, T_3, T_4, T_5, T_6, T_7, and T_i with the messages during the transmission of information over the network.

3. *Man-in-the-middle Attack:* In Secure-AKA, the EK key is used between the MS and the VLR. The EK key prohibits the communication from being eavesdropped.

4. *Impersonate Networks:* Following are some scenarios, in which an adversary can try to impersonate the UMTS network:

   Case-1: Let us consider that a malicious VLR is present in the network. In order to impersonate MS, the intruder must send valid response RES to the VLR, but the intruder does not get correct RES, since RES was sent between the MS and the correct VLR only.

Case-2: When the adversary tries to impersonate corrupted VLR, the adversary sends MAC_1 and LAI to the HLR in message (8), where MAC_1 was previously sent by the MS to the VLR. When the adversary asks for AV (in place of the actual user), the HLR rejects the request by verifying MAC_1.

Case-3: If the adversary tries to impersonate uncorrupted network, it fails as the MS did not previously ask for any AV.

5. *DoS Attack:* The attacker MS′ floods victim VLR for authentication by spoofing the IMSI/TMSI, T_1, ReqNo, and MAC_1. Then, T_2, ReqNo, Puz, PID are returned back from the VLR to the sender MS.

Case-1: The attacker MS′ floods the victim VLR by self IMSI: If the malicious MS does not respond within the threshold time duration to the proxy, then the connection is simply terminated. Then, the VLR resets authentication request and makes free the resources. Additionally, if it is a malicious user, then the proxy will not get the hash of Solu'_1 or will receive invalid hash. However, even if the attacker′s hash matches with α, he/she cannot compute the same Solu'_2, which requires the knowledge of a secret function f′ along with Rand.

Case-2: The attacker MS′ floods the victim VLR by spoofing IMSI: There are two different scenarios as:

Scenario-1: If the actual MS that gets a puzzle message Puz is inactive, then the proxy will not receive any information from the MS. This process is similar to case-1. The proxy waits for a threshold time to hear from the MS. After timeout, the connection is terminated.

Scenario-2: If the MS is alive, it asks to the VLR to refuse such request and then the connection is terminated. It is also proposed that the timeout to be in random nature, which controls the rate of attack. Further, if the MS′ modifies message (1), the message can be detected by the HLR on receipt of message (8), and that can also be observed by the VLR with DK key.

Thus, the Secure-AKA protocol resists such DoS attacks (DDoS and LDoS). In fact, in this protocol, the HLR is supposed to receive a TID from the MS, which is neither an actual IMSI of the MS nor a TMSI for the network. Thus, an actual IMSI or a fake TMSI with the f′ function are not used to extract the correct IMSI of the MS; hence, the connection will be terminated. Thus, there is no chance that the attacker will be able to generate the same TID from a victim MS′s IMSI.

6. *Dropping ACK Signal:* Since, the Secure-AKA protocol uses TID instead of IMSI or TMSI during the initial phase, thus, it is able to solve the dropping ACK signal issue. There is no confusion in the form of the transmitted identity of the MS. The reception of a new TID by VLR/HLR indicates the generation of new TMSI.

4.10.2 Communication Overhead

This subsection evaluates the communication overhead with respect to UMTS-AKA, Secure-AKA, and other AKA protocols. Total number of transmitted bits can be calculated with the help of values specified in Table 4.1. The total number of transmitted bits by each protocol is as follows:

UMTS-AKA Protocol: Phase-1: (1)+(2)+(3)= 256+608*n bits

Phase-2: ((4)+(5))*n= 352*n bits

Total transmitted bits = 256+960*n

EURASIP-AKA Protocol: Total transmitted bits = ((1)+(2)+(3))*n = 992*n bits

AP-AKA Protocol: It is assumed that the maximum idx= 32000= 2^{28} [84]

Phase-1: (1)+(2)+(3)+((4)*n) = 768+604*n

Phase-2: ((5)+(6))*n = 348*n

Total transmitted bits = 768+952*n

S-AKA Protocol: Phase-1: (1)+(2)+(3)+(4)+(5) = 1312 bits

Phase-2: ((1)+(2)+(3))*n = 680*n bits

Total transmitted bits = 1312+680*n

COCKTAIL-AKA Protocol: Phase-1: (1)+(2)+(3) = 1200 bits

Phase-2: ((4)+(5))*n = 432*n

Total transmitted bits = 1200+432*n

X-AKA Protocol: Phase-1: (1)+(2)+(3) = 880 bits

Phase-2: ((4)+(5))*n = 432*n

Total transmitted bits = 880+432*n

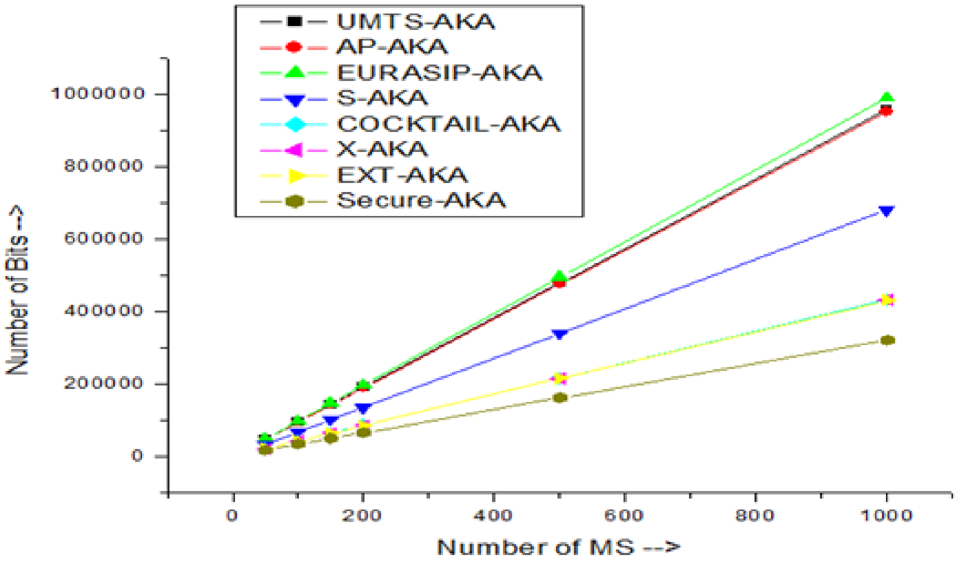


Figure 4.7: Communication overhead of various AKA protocols

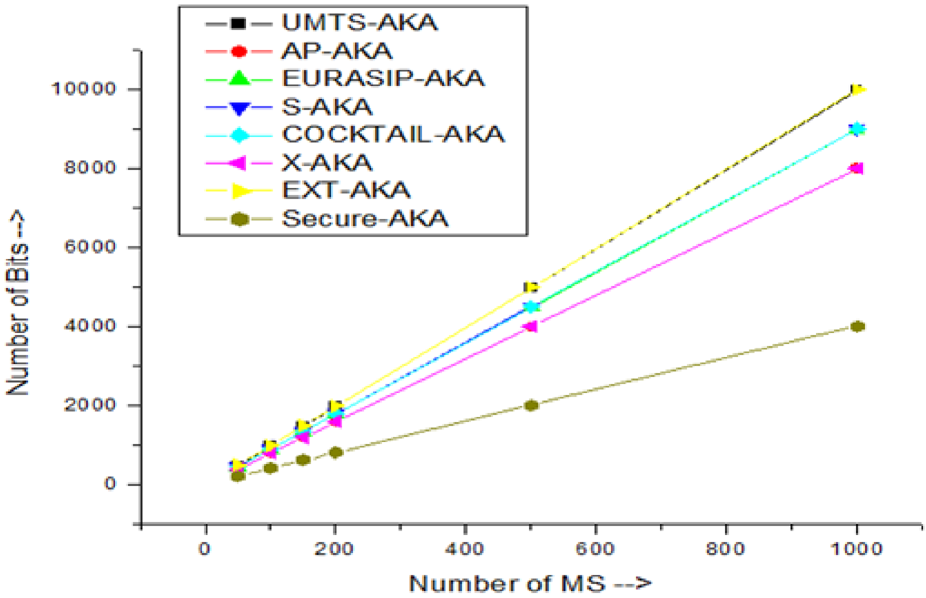


Figure 4.8: Computation overhead of various AKA protocols

EXT-AKA Protocol: Phase-1: (1)+(2)+(3) = 1008 bits

Phase-2: ((4)+(5))*n = 432*n

Total transmitted bits = 1008+432*n

Secure-AKA Protocol: Phase-1: ((1)+(2)+(3)+(4)+(5)+(6)+(7)+(8)+(9)+(10) +(11)+(12)+(13)) = 3289 bits

Phase-2: ((1)+(2)+(3))*n = 320*n

Total transmitted bits = 3289+320*n

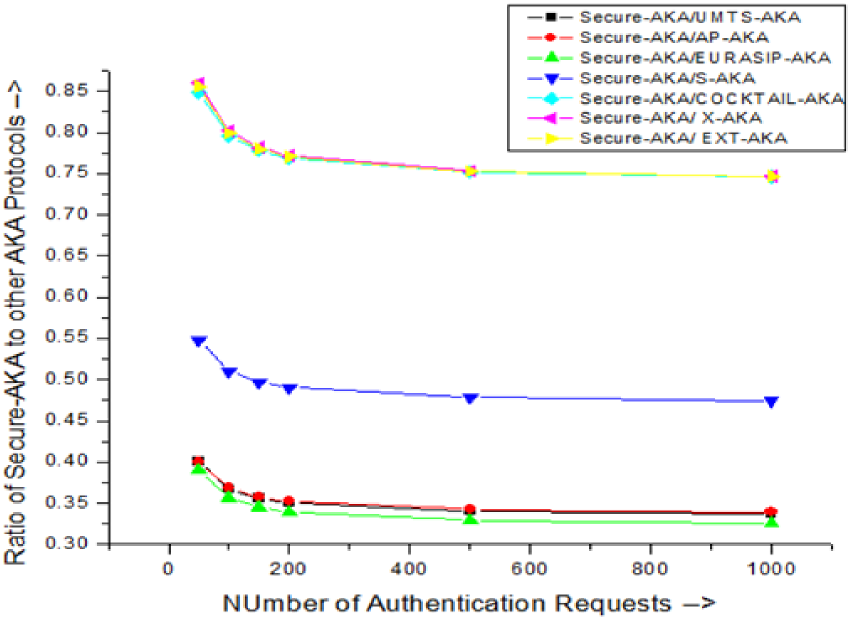


Figure 4.9: Communication overhead (Secure-AKA/other protocols)

Figure 4.7 represents the communication overhead generated by various existing AKA protocols by varying the number of mobile stations for authentication requests. It is clear that Secure-AKA protocol produces lesser communication overhead as compared to existing AP-AKA, S-AKA, EURASIP-AKA, UMTS-AKA, COCKTAIL-AKA, X-AKA, and EXT-AKA protocols.

4.10.3 Computation Overhead

A unit value is considered in order to measure the computational overhead for all the security functions used (without knowing the structure) in various AKA protocols, however, practically these functions may be differ and are operator specific. To evaluate the computation overhead, we need to calculate the total number of functions used in each AKA protocol.

UMTS-AKA Protocol: Phase-1: {f_1, f_2, f_3, f_4, f_5}*n = 5*n

Phase-2: {f_1, f_2, f_3, f_4, f_5}*n = 5*n

Total functions used= 10*n

EURASIP-AKA Protocol: Total functions used = {f_5, f_1, f_1, f_2, f_2, f_3, f_4, f_3, f_4}*n = 9*n

AP-AKA Protocol: Phase-1: Hk, Fk, Fk, {Fk, Gk, Hk, Fk}*n = 3+4*n

Phase-2: {Hk, Fk, Fk, Fk}*n = 4*n

Total functions used= 3+8*n

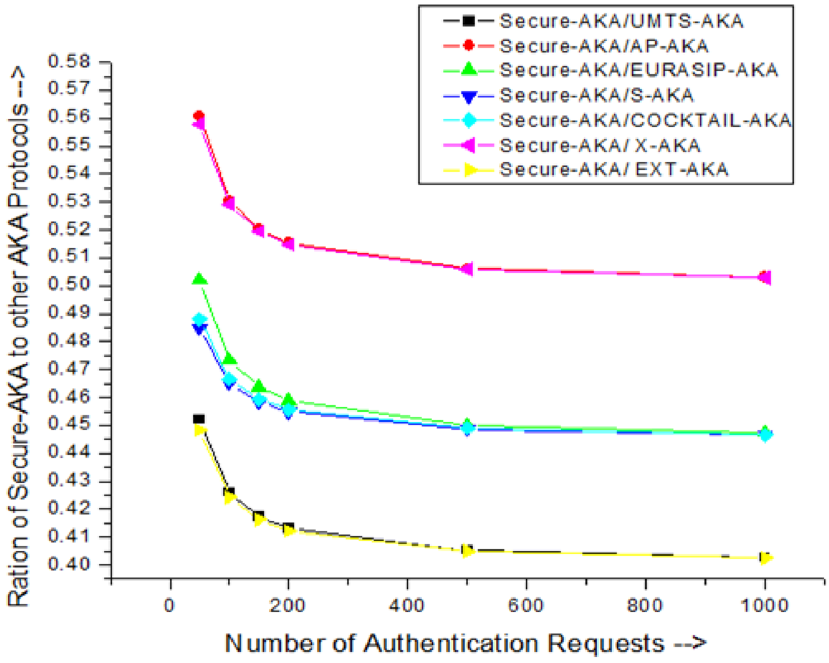


Figure 4.10: Computation overhead (Secure-AKA/other protocols)

S-AKA Protocol: Phase-1: {f_1, f_6, f_1, f_1, f_6, f_1, f_1, f_1, f_3, f_4, f_2, f_7, f_2, f_3, f_4, f_7} = 16

Phase-2: {f_1, f_6, f_1, f_1, f_1, f_1, f_1, f_2, f_2}*n = 9*n

Total functions used= 16+9*n

COCKTAIL-AKA Protocol: Phase-1: {f_1, f_5, f_3, f_4, f_2, f_2, f_2, f_3, f_4, f_3, f_4, f_2, f_5} = 13

Phase-2: {f_1, f_1, f_1, f_2, f_3, f_4, f_2, f_3, f_4}*n = 9*n

Total functions used= 13+9*n

X-AKA Protocol: Phase-1: {f_1, f_1, f_x, f_1, f_1} = 5

Phase-2: {f_1, f_1, f_2, f_2, f_3, f_4, f_3, f_4}*n = 8*n

Total functions used= 5+8*n

EXT-AKA Protocol: Phase-1: {f_1, f_{99}, f_1, f_1} = 4

Phase-2: {f_1, f_{99}, f_1, f_1, f_2, f_2, f_3, f_4, f_3, f_4}*n = 10*n

Total functions used= 4+10*n

Secure-AKA Protocol: Phase-1: {f′, f_1, g, H, $Solu_1$, $Solu_1$, $Solu_2$, $Solu_2$, f′, f_1, f_1, f_2, f_1, f_1, f_1, f_2, f_4, f_5, f_3, f_6, f_4, f_5, f_3, f_6, $\{\}_{EK}$, $\{\}_{DK}$} = 26

Phase-2: {f_1, f_1, f_3, f_3}*n = 4*n

Total functions used= 26+4*n

Figure 4.8 illustrates the computation overhead generated by each AKA protocol. It shows that the Secure-AKA protocol generates least computation

Table 4.14: Bandwidth consumption analysis

No. of AVs	AP-AKA/ UMTS-AKA	EURASIP-AKA/ UMTS-AKA	S-AKA/ UMTS-AKA	COCKTAIL-AKA/ UMTS-AKA	X-AKA/ UMTS-AKA	EXT-AKA/ UMTS-AKA	Secure-AKA/ UMTS-AKA
50	1	1.02	0.73	0.47	0.46	0.46	0.4
100	0.99	1.03	0.72	0.46	0.45	0.45	0.36
200	0.99	1.03	0.71	0.45	0.45	0.45	0.35
500	0.99	1.03	0.71	0.45	0.45	0.45	0.34
1000	0.99	1.03	0.7	0.45	0.45	0.45	0.33
Average	0.99	1.02	0.71	0.45	0.45	0.45	0.35

Table 4.15: Message exchanged analysis

No. of AVs	AP-AKA/ UMTS-AKA	EURASIP-AKA/ UMTS-AKA	S-AKA/ UMTS-AKA	COCKTAIL-AKA/ UMTS-AKA	X-AKA/ UMTS-AKA	EXT-AKA/ UMTS-AKA	Secure-AKA/ UMTS-AKA
50	0.8	0.9	0.93	0.92	0.81	1	0.45
100	0.8	0.9	0.91	0.91	0.8	1	0.42
200	0.8	0.9	0.9	0.9	0.8	1	0.41
500	0.8	0.9	0.9	0.9	0.8	1	0.4
1000	0.8	0.9	0.9	0.9	0.8	1	0.4
Average	0.8	0.9	0.9	0.9	0.8	1	0.41

overhead than UMTS-AKA, AP-AKA, EURASIP-AKA, S-AKA, COCKTAIL-AKA, X-AKA, and EXT-AKA protocol. An individual 1:1 comparison for Secure-AKA protocol is carried out in Figure 4.9 and Figure 4.10 for the communication and computation overheads respectively with respect to already existing AKA protocols, where number of authentication requests n = 50,100, 200, 500, 1000 (assumption). The utmost efficiency of the Secure-AKA protocol is observed in terms of maximum reduction of communication and computation overheads against EURASIP-AKA, EXT-AKA, and UMTS-AKA protocols respectively.

4.10.4 Bandwidth Consumption

Table 4.14 represents the bandwidth consumption of each AKA protocol, when n = 50, 100, 200, 500, 1000. It is clear that on an average, the Secure-AKA

Table 4.16: Average overhead analysis

Parameters	Secure-AKA/ UMTS-AKA	Secure-AKA/ AP-AKA	Secure-AKA/ EURASIP-AKA	Secure-AKA/ S-AKA	Secure-AKA/ COCKTAIL-AKA	Secure-AKA/ X-AKA	Secure-AKA/ EXT-AKA
Avg. B/w Consumption	0.35	0.36	0.34	0.49	0.78	0.78	0.78
Avg. Message Exchange	0.41	0.52	0.46	0.45	0.46	0.52	0.41

protocol is able to reduce the bandwidth consumption by 65%, which is the utmost reduction of bandwidth by any AKA protocol from the literature in comparison to the UMTS-AKA protocol. Similarly, Table 4.15 represents that Secure-AKA protocol diminishes 59% of the messages exchanged ratio (in terms of computations) for the authentication, when n = 50, 100, 200, 500, and 1000. Table 4.16 represents the average bandwidth consumption and average message exchanged ratio for the Secure-AKA protocol with respect to the existing AKA protocols. Table concludes that on an average, the Secure-AKA protocol lowers 65%, 64%, 66%, 51%, 22%, 22%, and 22% of the bandwidth consumption as compared to UMTS-AKA, AP-AKA, EURASIP-AKA, S-AKA, COCKTAIL-AKA, X-AKA, and EXT-AKA protocols respectively. Similarly, on an average the Secure-AKA protocol is able to lower 59%, 48%, 54%, 55%, 54%, 48%, and 59% of the messages exchanged ratio during the authentication in comparison to UMTS-AKA, AP-AKA, EURASIP-AKA S-AKA, COCKTAIL-AKA, X-AKA, and EXT-AKA protocols respectively.

Table 4.17: Protocols vs. prevention from different attacks

Attacks	UMTS-AKA	AP-AKA	X-AKA	EXT-AKA	EURA-SIP-AKA	COCK-TAIL-AKA	S-AKA	Secure-AKA
Replay	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Corrupt N/w	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
MITM	No	No	No	No	No	Yes	Yes	Yes
Redirection	No	Yes	No	No	No	Yes	Yes	Yes
DoS	No	No	No	No	No	No	Partial	Yes

4.10.5 Summary of Attack Resistance

Table 4.17 summarizes the resistance to various attacks by different existing AKA protocols for the UMTS networks. All the existing protocols mentioned in Table 4.17 are free from replay and active attacks in corrupt networks. The AP-AKA prevents the network from redirection attack but, penetrable to man-in-the-middle attack. The X-AKA, EXT-AKA, and EURASIP-AKA are not able to prevent the UMTS network from redirection and man-in-the-middle attacks; however, COCKTAIL-AKA and S-AKA are able to defeat such attacks. The Secure-AKA protocol prevents the UMTS network from all such attacks including DoS attack, while only the S-AKA protocol is able to resist DoS attack partially.

4.11 Secure-AKA Protocol Simulation

Similar to ES-AKA protocol, the simulation of this protocol is performed in Java environment. The function $f_1()$ is implemented as HMACSHA1. Further, functions $f_2()$, $f_3()$, $f_4()$, $f_5()$, and $f_6()$ are considered as HMACSHA256. However, all these functions are network operator specific. The output of $f_1()$ is truncated to 64 bits as the output of this functions is $MCA_1/MAC_2/MAC_3/MAC'_3$ of 64 bits in size. Similarly, the output of $f_2()/f_4()/f_5()/f_6()$, and $f_3()$ are truncated to 128 and 64 bits respectively. In all tables, the time is measured in milliseconds while memory/space is in bytes. The message transmission time is calculated similar to the assumption in ES-AKA with data speed of 384 kbps and 2 Mbps.
Secure-AKA Protocol (384 kbps): Total Secure-AKA messages transmission time = (1) ReqNo, TID, T_1, MAC_1 + (2) ReqNo, T_2, Puz, PID + (3) ReqNo, Rand, Puz + (4) ReqNo, T_3, H + (5) Y/N, $Solu_1$ + (6) T_4, ReqNo, Rand + (7) T_5, ReqNo, $Solu'_2$ + (8) TID, T_1, MAC_1, LAI + (9) T_6, MAC_2, DK, AMF + (10) MAC_3, T_6, T_7, AMF + (11) RES + (12) $\{TMSI\}_{EK}$ + (13) ACK = 3289*2543/1000*1000 = 8.3 milliseconds
Secure-AKA Protocol (2 Mbps): Total Secure-AKA messages transmission time = (1) ReqNo, TID, T_1, MAC_1 + (2) ReqNo, T_2, Puz, PID + (3) ReqNo, Rand, Puz + (4) ReqNo, T_3, H + (5) Y/N, $Solu_1$ + (6) T_4, ReqNo, Rand + (7) T_5, ReqNo, $Solu'_2$ + (8) TID, T_1, MAC_1, LAI + (9) T_6, MAC_2, DK, AMF

+ (10) MAC_3, T_6, T_7, AMF + (11) RES + (12) $\{TMSI\}_{EK}$ + (13) ACK = 3289*476/1000*1000 = 1.6 milliseconds

Table 4.18: Computation for f′() and g()/H in Secure-AKA

Function f′()			g()/H		
ExT	PCPUT	TUM	ExT	PCPUT	TUM
0.84	93.60	12968288	78.0	109.20	12990648

Table 4.19: Computation for other functions in Secure-AKA

$f_1()$			$f_2()/f_3()/f_4()/f_5()/f_6()$		
ExT	PCPUT	TUM	ExT	PCPUT	TUM
221.60	296.40	15211840	273.41	296.40	15204024

The execution time, process CPU time, and total memory usage for f′(), g()/H, $f_1()$, and $f_2()/f_3()/f_4()/f_6()$ can be observed from Table 4.18 and Table 4.19. A simple XOR-based function is implemented for f′(), however it is recommended to use a complex function. For the encryption and decryption of TMSI in step-12 of the Secure-AKA protocol, we prefer to use AES algorithm, which is considered one of the best algorithms. It has been found that AES takes 40 milliseconds to encrypt ({}EK) and 18 milliseconds to decrypt ({}DK) a single message [100]. Further, as an example, it has been observed that the puzzle solution time is 44.88 seconds (44880 milliseconds) to generate and solve a puzzle [101]. The security level for the puzzle is considered as 2^{40}, which is very difficult level. The solution to puzzle is generated using Smart Dust simulator, which is a Java simulator. However, the puzzle generation and solution time differs from puzzle to puzzle and their complexity levels. Since, the transmission time for all messages sent along with the computation time for each function is known, thus, it is easy to calculate the total time required for the execution of Secure-AKA protocol.

Total execution time for Secure-AKA = Total messages transmission time + 2*f′() for IMSI/TMSI + 2*g()/H for hash + 2*$f_1()$ for MAC_1 + 2*$f_1()$ for MAC_2 + 2*$f_1()$ for MAC_3 + 2*$f_3()$ for RES + 2*$f_4()$ for CK + 2*$f_6()$ for EK +2*$f_5()$ for IK + 2*$f_2()$ for DK + $\{\}_{EK}$ + $\{\}_{DK}$ + Puzzle solution time [Puzzle generation + 2*$Solu_1()$ for Puzzle + 2*$Solu_2()$ for Puzzle]

Total execution time for Secure-AKA protocol (384 kbps) = 8.3 + 2*0.84 +

2*78.0 + 2*221.60 +2*221.60 + 2*221.60 + 2* 273.41 + 2* 273.41 + 2* 273.41 + 2* 273.41 + 2* 273.41 + 40 + 18 + Puzzle solution time = 8.5 + 1.68 + 156 + 443.20 + 443.20 + 443.20 + 546.82 + 546.82 + 546.82 + 546.82 + 546.82 + 40 + 18 + Puzzle solution time = 4287.68 + Puzzle solution time (in milliseconds)
Total execution time for Secure-AKA protocol (2 Mbps) = 1.6 + 2*0.84 + 2*78.0 + 2*221.60 +2*221.60 + 2*221.60 + 2* 273.41 + 2* 273.41 + 2* 273.41 + 2* 273.41 + 2* 273.41 + 40 + 18 + Puzzle solution time = 8.5 + 1.68 + 156 + 443.20 + 443.20 + 443.20 + 546.82 + 546.82 + 546.82 + 546.82 + 546.82 + 40 + 18 + Puzzle solution time = 4281.18 + Puzzle solution time (in milliseconds)

Since, the puzzle solution time is 44880 milliseconds, thus, total execution time for Secure-AKA with 384 kbps speed = 4287.68 + 44880 = 49167.68 milliseconds, and total execution time for Secure-AKA with 2Mbps speed = 4281.18 + 44880 = 49161.18 milliseconds. Similarly, the estimate time for each subsequent authentication (phase-2) during the session can be calculated as:
Total execution time for Secure-AKA phase-2 protocol = Total messages transmission time (Phase-2) + $f_1()$ for MAC$'_3$ + 2*$f_3()$ for RES$'$
Total execution time for Secure-AKA phase-2 protocol (384 kbps) = (320*2543/ 1000*1000) + 2*221.60+ 2* 273.41 = 990.83 milliseconds
Total Execution Time for Secure-AKA Phase-2 Protocol (2 Mbps) = (320*476/ 1000*1000) + 2*221.60+ 2* 273.41 = 990.17 milliseconds

4.12 Formal Proof of Secure-AKA Protocol

BAN-Logic symbols are used to formally proof the authentication process of Secure-AKA protocol. Various notations used in BAN-Logic are explained in section 3.7.4 (chapter 3).

1. *The Formal Messages in Secure-AKA Protocol:*

   Phase-1: (1) MS → VLR: Ta, ReqNo, f$'$(IMSI, Ta), f_1(Ta, LAI)$_{SK}$; MS $\overset{SK}{\leftrightarrow}$ HLR

   (2) VLR → MS: Tb, ReqNo, Puz, PID

   (3) VLR → Proxy: ReqNo, Rand, Puz

   (4) MS → Proxy: Tc, ReqNo, H

   (5) Proxy → VLR: Flag, Solution$_1$

   (6) Proxy → MS: Td, ReqNo, Rand

(7) MS → VLR: Te, ReqNo, Solution$_2$

(8) VLR → HLR: TID, Ta, LAI, f$_1$(Ta, LAI)$_{SK}$

(9) HLR → VLR: Tf, f$_2$(Ta)$_{SK}$, AMF, f$_1$(Tf, AMF)$_{SK}$; VLR $\overset{DK}{\leftrightarrow}$ HLR and f$_2$(Ta)$_{SK}$

(10) VLR → MS: Tf, Tg, AMF, f$_1$(f$_1$(Tf, AMF)$_{SK}$, Tf, Tg)$_{DK}$

(11) MS → VLR: f$_3$(Tg)$_{DK}$; f$_4$(Tf)$_{DK}$, f$_5$(Tf)$_{DK}$, f$_6$(Tf)$_{DK}$, MS $\overset{CK}{\leftrightarrow}$ VLR, MS $\overset{IK}{\leftrightarrow}$ VLR, MS $\overset{EK}{\leftrightarrow}$ VLR

(12) VLR → MS: {TMSI}$_{EK}$

(13) MS → VLR: ACK{TMSI}

Phase-2: (1) MS → VLR: T$_i$, TMSI

(2) VLR → MS: f$_1$(f$_1$(Tf, AMF)$_{SK}$, Tf, T$_i$)$_{DK}$

(3) MS → VLR: f$_3$(T$_i$)$_{DK}$

2. *Security Assumptions:*

a) It is assumed that SK key is shared between the MS and its HLR.

(1) MS has the secure key SK and MS $|\equiv$ MS $\overset{SK}{\leftrightarrow}$ HLR

(2) HLR has the secure key SK and HLR $|\equiv$ MS $\overset{SK}{\leftrightarrow}$ HLR

b) It is assumed that the VLR trusts the HLR.

(1) VLR $|\equiv$ HLR $|\Rightarrow$ MS $\overset{SK}{\leftrightarrow}$ HLR,

(2) $\frac{VLR|\!\sim P, HLR \lhd P}{HLR|\equiv VLR|\equiv P}$

(3) $\frac{HLR|\!\sim P, VLR \lhd P}{VLR|\equiv HLR|\equiv P}$

c) It is assumed that communication between the HLR and the VLR is secure.

(1) VLR $|\equiv$ VLR $\overset{P}{\Leftrightarrow}$ HLR, P is conveyance message between the VLR and the HLR.

(2) HLR $|\equiv$ VLR $\overset{P}{\Leftrightarrow}$ HLR, P is conveyance message between the VLR and the HLR.

3. *Security Analysis:*

Phase-1: (1) MS → VLR: MS $|\equiv$ #(Ta), VLR $\lhd$ Ta, ReqNo, f′(IMSI, Ta), f$_1$(Ta, LAI)$_{SK}$

(2) VLR → MS: VLR $|\equiv$ #(Tb), MS $\lhd$ Tb, ReqNo, Puz, PID

(3) VLR → Proxy: Proxy $\lhd$ ReqNo, Rand, Puz

(4) MS → Proxy: MS $|\equiv$ #(Tc), Proxy $\lhd$ ReqNo, H; on receiving, the proxy compares the α with the received H. If it does not hold, then con-

nection is terminated.

(5) Proxy → VLR: VLR $\triangleleft$ Flag, Solution$_1$

(6) Proxy → MS: Proxy $|\equiv$ #(Td), MS $\triangleleft$ Td, ReqNo, Rand; The MS computes Solu$'_2$

(7) MS → VLR: MS $|\equiv$ #(Te); VLR $\triangleleft$ Te, ReqNo, Solution$_2$; on receiving the VLR computes Solu$_2$ (from Solu$_1$ and Rand using a function f) and compares with the received Solu$'_2$

(8) VLR → HLR: HLR $\triangleleft$ Ta, LAI, f$_1$(Ta, LAI)$_{SK}$, f$'$(IMSI, Ta); on receiving, the HLR derives IMSI as f$'$(TID, LAI); computes f$_1$(Ta, LAI)$_{SK}$ and compares with the received f$_1$(Ta, LAI)$_{SK}$

(9) HLR → VLR: HLR $|\equiv$ #(Tf), VLR $\triangleleft$ Tf, f$_2$(Ta)$_{SK}$, AMF, f$_1$(Tf, AMF)$_{SK}$; HLR $|\equiv \forall$ (VLR $\overset{DK}{\leftrightarrow}$ MS)

(10) VLR → MS: MS $\triangleleft$ Tf, Tg, AMF, f$_1$(f$_1$(Tf, AMF)$_{SK}$, Tf, Tg)$_{DK}$; MS computes f$_1$(f$_1$(Tf, AMF)$_{SK}$, Tf, Tg)$_{DK}$ and compares with the received f$_1$(f$_1$(Tf, AMF)$_{SK}$, Tf, Tg)$_{DK}$; MS generates f$_2$(Ta)$_{SK}$, f$_3$(Tg)$_{DK}$, f$_4$(Tf)$_{DK}$, f$_5$(Tf)$_{DK}$, and f$_6$(Tf)$_{DK}$

(11) MS → VLR: VLR $\triangleleft$ f$_3$(Tg)$_{DK}$; on receiving, the VLR derives f$_3$(Tg)$_{DK}$ and compares with the received f$_3$(Tg)$_{DK}$; the VLR generates f$_4$(Tf)$_{DK}$, f$_5$(Tf)$_{DK}$, f$_6$(Tf)$_{DK}$; HLR $| \equiv \forall$ (MS $\overset{CK}{\leftrightarrow}$ VLR), HLR $| \equiv \forall$ (MS $\overset{IK}{\leftrightarrow}$ VLR), HLR $| \equiv \forall$ (MS $\overset{EK}{\leftrightarrow}$ VLR)

(12) VLR → MS: E{TMSI}$_{EK}$

(13) MS → VLR: On receiving, the MS computes D{TMSI}$_{EK}$, then MS → VLR: VLR $\triangleleft$ ACK{TMSI}

Phase-2: (1) MS → VLR: MS $| \equiv$ #(T$_i$), VLR $\triangleleft$ T$_i$, TMSI

(2) On receiving, the VLR checks TMSI and T$_i$ <= ExpT; VLR → MS: f$_1$(f$_1$(Tf, AMF)$_{SK}$, Tf, T$_i$)$_{DK}$

(3) On receiving, the MS computes f$_1$(f$_1$(Tf, AMF)$_{SK}$, Tf, T$_i$)$_{DK}$ and compares with the received f$_1$(f$_1$(Tf, AMF)$_{SK}$, Tf, T$_i$)$_{DK}$; then MS → VLR: f$_3$(T$_i$)$_{DK}$; If the VLR finds that the calculated f$_3$(T$_i$)$_{DK}$ and the received f$_3$(T$_i$)$_{DK}$ are equal then only the authentication is successful.

4. *Message Meaning Rule:*

(1) $\frac{MS|\equiv(MS\overset{SK}{\leftrightarrow}HLR)\wedge(VLR\overset{DK}{\leftrightarrow}MS),MS\triangleleft f_1(Ta,LAI)_{SK}}{MS|\equiv HLR\upharpoonright f_1(Ta,LAI)_{SK}}$

(2) $\frac{VLR|\equiv f_1(Tf,AMF)_{SK}\wedge(VLR\overset{DK}{\leftrightarrow}MS),VLR\triangleleft f_1(f_1(Tf,AMF)_{SK},Tf,Tg)_{DK}}{VLR|\equiv HLR\upharpoonright f_1(f_1(Tf,AMF)_{SK},Tf,Tg)_{DK}}$

5. *Nonce/Timestamp Verification Rule:*

   (1) $\frac{MS|\equiv\#(Ta),MS|\equiv HLR\upharpoonright f_1(Ta,LAI)_{SK}}{MS|\equiv HLR|\equiv f_1(Ta,LAI)_{SK}}$

   (2) $\frac{HLR|\equiv\#(Tb)\wedge VLR|\equiv\#(Tc),VLR|\equiv HLR\upharpoonright f_1(f_1(Tf,AMF)_{SK},Tf,Tg)_{DK}}{MS|\equiv HLR|\equiv f_1(f_1(Tf,AMF)_{SK},Tf,Tg)_{DK}}$

6. *Jurisdiction Rule:*

   (1) $\frac{MS|\equiv HLR\Rightarrow f_1(Ta,LAI)_{SK},MS\triangleleft VLR\upharpoonright f_1(Ta,LAI)_{SK}}{MS|\equiv VLR|\equiv HLR}$

   (2) $\frac{VLR|\equiv HLR\Rightarrow f_1(f_1(Tf,AMF)_{SK},Tf,Tg)_{DK},VLR\triangleleft VLR\upharpoonright f_1(f_1(Tf,AMF)_{SK},Tf,Tg)_{DK}}{VLR|\equiv HLR|\equiv MS}$

7. *Confidentiality between the MS and the VLR:*

   $\frac{MS|\equiv(MS\overset{EK}{\leftrightarrow}VLR),MS\triangleleft(Msg)_{EK}}{MS|\equiv VLR\upharpoonright Msg} \wedge \frac{VLR|\equiv(VLR\overset{EK}{\leftrightarrow}MS),VLR\triangleleft(Msg)_{EK}}{VLR|\equiv MS\upharpoonright Msg}$

8. *Protocol Goals:*

   a) Mutual authentication between the MS and the VLR:

   MS $|\equiv$ HLR $|\equiv$ VLR $\wedge$ VLR $|\equiv$ HLR $|\equiv$ MS $\rightarrow$ MS $|\equiv$ VLR $\wedge$ VLR $|\equiv$ MS. Thus, mutual authentication holds.

   b) Key agreement between the MS and the VLR:

   There is a DK key between the VLR and the MS to provide agreement.

   MS $|\equiv$ DK $\wedge$ #(Ta), since DK = $\mathrm{f}_2(\mathrm{Ta})_{SK}$

   VLR $|\equiv$ DK $\wedge$ #(Tf), since HLR $\rightarrow$ VLR $\upharpoonright$ DK

   c) Resistance replay attack between the MS and the VLR:

   If the attacker gets #Ta, #Tb, #Tc, #Td, #Te, and #Tf from messages (1-9) in phase-1 or #Ti from message (1) in phase-2, he/she cannot forge messages because he/she does not know how to derive these values. Since, Ta, Tb, Tc, Td, Te, Tf, and T_i will be changed at the next time, thus, goal of resistance replay attack between the MS and the VLR holds.

   d) Resistance man-in-the-middle Attack between the MS and the VLR:

   Since, the attacker does not knows EK key or encryption algorithm/function, thus, it prevents the communication from being eavesdropped.

   e) Resistance redirection attack between the MS and the VLR:

   Since, $\mathrm{f}_1(\mathrm{Ta, LAI})_{SK}$ is used to maintain the integrity of LAI, thus, it prevents from the redirection attack.

   f) Resistance impersonation attack between the MS and the VLR:

   (1) Adversary tries to impersonate the MS: Since, $\mathrm{f}_1(\mathrm{Ta, LAI})_{SK}$ is computed at MS and compared at HLR, this prevents from impersonation attack. Additionally, adversary must reply with a valid $\mathrm{f}_3(\mathrm{Tg})_{DK}$ but he/she neither has DK nor EK key, where DK = $\mathrm{f}_2(\mathrm{Ta})_{SK}$ and EK =

$f_6(Tf)_{DK}$.

(2) Adversary tries to impersonate the VLR: The integrity value $f_1(Ta, LAI)_{SK}$ at MS and at HLR will be violated. Additionally, if the MS receives $f_1(f_1(Tf, AMF)_{SK}, Tf, Tg)_{DK}$ at any time, then the connection will be terminated because the MS has not sent any request to the VLR.

g) Resistance DoS attack between the MS and the VLR: Each MS has to solve a puzzle Puz into two phases by computing $Solu_1$ and $Solu_2$, and allows access to the resources only after the successful verification of $Solu_2$, where $Solu_1$ (128 bits) = $Solu_1(Puz, T_2)$ and $Solu_2$ (128 bits) = $Solu_2(Solu_1, Rand)$. Additionally, each MS has to wait for a threshold time before computing the $Solu_2$. All this prevents the network from DoS attack.

4.13 Chapter Summary

This chapter has shown that the ES-AKA protocol with MAES-128 cipher algorithm offers an efficient and secure service as compared to all existing protocols for the UMTS network. The ES-AKA protocol is free from various attacks that exist in the original UMTS-AKA protocol. Further, it does not propose to use a counter for synchronization purpose between the MS and the VLR. On an average, ES-AKA saves 62% of the total bandwidth and reduces 6% messages exchanged ratio (in terms of computations) during the authentication in comparison to the original UMTS-AKA protocol, when number of MS (n) = 5, 10, 50, 100, 200, 500, and 1000. It produces lesser communication overhead as compared to all other AKA protocols discussed. However, it is only better than UMTS-AKA, EXT-AKA, COCKTAIL-AKA and S-AKA protocols in terms of computation overhead. This ES-AKA protocol resists DoS attack, when the power of an adversary and a lagitimate user are equivalent in terms of resource utilization. If it is not the case, then this protocol may not be able to prevent the network from DoS attack. Further, we analyzed such scenario and proposed an improved and more efficient Secure-AKA protocol for the UMTS network with the assumption that the adversary is more powerful as compared to valid user in resource consumption. The Secure-AKA protocol protects the transmission of IMSI over the network. On an average, the Secure-AKA protocol reduces bandwidth consumption by 65%, 64%, 66%, 51%, 22%, 22%, and 22% in comparison to UMTS-AKA, AP-AKA, EURASIP-AKA, S-AKA, COCKTAIL-AKA, X-AKA, and EXT-AKA protocols respectively, when n = 50, 100, 200, 500, and 1000. This 65% drop in bandwidth utilization by Secure-AKA is the maximum reduction of the bandwidth by any UMTS protocol. Further, we have observed that this protocol lowers the ratio of messages exchanged by 59%, 48%, 54%, 55%, 54%, 48%, and 59% during the authentication with respect to UMTS-AKA, AP-AKA, EURASIP-AKA, S-AKA, COCKTAIL-AKA, X-AKA, and EXT-AKA protocol, when n = 50, 100, 200, 500, and 1000. An improved cipher algorithm named MAES-128 has been proposed for the UMTS network, which provides much faster encryption and decryption (0.27 and 0.25 milliseconds) than the existing KASUMI (1.26 and 3.79 milliseconds) and AES (0.51 and 0.49 milliseconds) algorithms.

Chapter 5

AKA Protocol in 4G LTE Network

5.1 Introduction

Long Term Evolution is a radio access technology, which provides very high speed of data upto 100 Mbps. Previously, the existing and proposed 2G GSM-AKA protocols [15], [18], [21], [102] and 3G UMTS-AKA protocols [22], [28], [103], [104] do not fit well in 4G systems due to some weaknesses. 4G networks are heterogeneous networks that are connected with wireless and some unprotected wired parties of 2G/3G networks via IP (Internet Protocol)- based bone networks. Both 2G and 3G AKA protocols do not provide mutual authentication between wired parties. The 4G network has solved these issues of 2G/3G networks and proposed the EPS-AKA protocol. However, this protocol has some major security drawbacks [105], [106] as follows:

1. In the LTE network, the identity of a user, *i.e.*, IMSI, is sent from the User Entity (UE) to the MME in clear text over the air interface (for initial request), which causes man-in-the-middle attack and user-id theft attack.

2. Passing the clear text Key Set Identifier (KSI) for Access Security Management Entity (ASME) from the MME to the UE over the network is another problem in the cellular networks. The protection of IMSI and KSI$_{ASME}$ are very crucial during the communication over the LTE networks as an adversary can misuse these parameters, which may result in

user-id theft and key-id theft attacks.

3. In the EPS-AKA protocol of 4G LTE network, the UE and the HSS, each maintains a counter, which result in synchronization problem.

5.2 EPS-AKA Protocol

In this section, we briefly outline the EPS-AKA protocol along with EPS key hierarchy. Various symbols and abbreviations used in the chapter are represented in Table 5.1, while Table 5.2 describes the role of each cryptographic function.

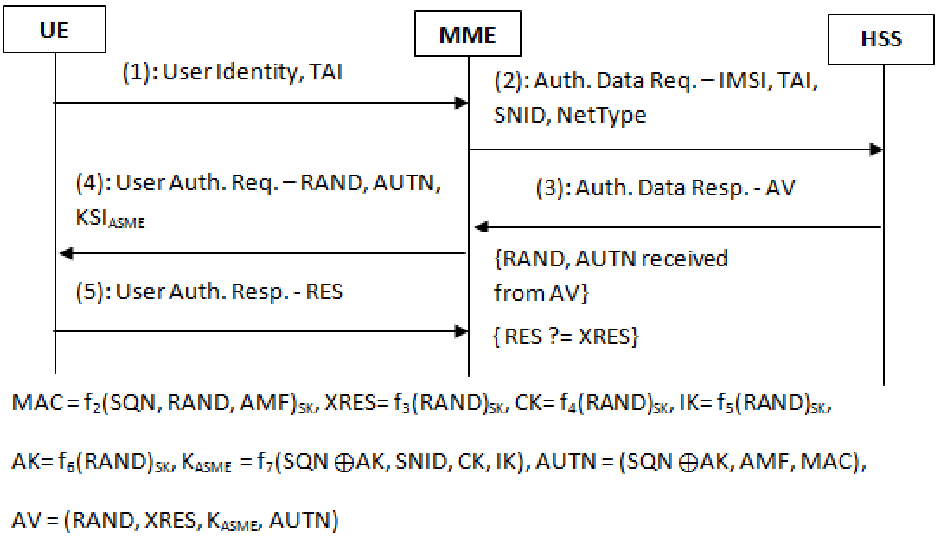

UE
MME
HSS
(1): User Identity, TAI
(2): Auth. Data Req. – IMSI, TAI, SNID, NetType
(3): Auth. Data Resp. - AV
(4): User Auth. Req. – RAND, AUTN, KSI_ASME
{RAND, AUTN received from AV}
(5): User Auth. Resp. - RES
{RES ?= XRES}
MAC = f_2(SQN, RAND, AMF)_SK, XRES= f_3(RAND)_SK, CK= f_4(RAND)_SK, IK= f_5(RAND)_SK,
AK= f_6(RAND)_SK, K_ASME = f_7(SQN ⊕AK, SNID, CK, IK), AUTN = (SQN ⊕AK, AMF, MAC),
AV = (RAND, XRES, K_ASME, AUTN)

Figure 5.1: EPS-AKA protocol for 4G LTE network

5.2.1 Overview of EPS-AKA Protocol

The execution of EPS-AKA protocol generates keys for Radio Resource Control (RRC) signaling, Non-Access Stratum (NAS) signaling and User Plane (UP). Figure 5.1 shows the working of the EPS-AKA protocol. The EPS-AKA protocol begins by sending a service request with an attach request NAS message from the UE to the MME. The MME verifies the identity of the UE and asks to send its IMSI or GUTI from TAU procedure based on whether the UE is requesting first time or it is not a new user for this network. Then, the MME sends GUTI with TAU message to old MME over S10 interface to extract the actual IMSI of the UE. If the UE was earlier in the roamed 2G/3G network, then the new MME makes connection to old MME through SGSN via S3 interface to extract

Table 5.1: Symbols and abbreviations

Symbol	Definition	Bits
IMSI/ID	International Mobile Subscriber Identity	128
TID/GUTI	Temporary Identity/ Global Unique TID	128
DMSI	Dynamic Mobile Subscriber Identity	166
CID	MME/ASME EPS Context Identity	48
SQN	Sequence Number	48
TAI	Tracking Area Identity	64
SNID	Serving Network Identity	128
AV-req	Authentication Vector Request	8
NetType	Network Type	3
PI	Protocol Identifier	4
AMF	Authentication Management Field	48
RAND/MSR/NSR	Random Number	128
AUTN	Authentication Token	Variable
AV	Authentication Vector	Variable
AK/XAK	Anonymity key	128
CK/XCK	Cipher key	128
IK/XIK	Integrity key	128
SK/K	Secret key shared b/w MS and HLR	128
$KSI_{ASME}/XKSI_{ASME}$	Key Set Identifier for each K_{ASME}	3
ACK	Acknowledgement	3
K_{ASME}	MME Intermediate key	256
K_{NASint}	Integrity key for NAS signaling	256
K_{NASenc}	Cipher key for NAS signaling	256
K_{eNB}	Intermediate key b/w MME and UE	256
K_{UPenc}	Cipher key for User Plane	256
K_{RRCenc}	Cipher key for RRC signaling	256
K_{RRCint}	Integrity key for RRC signaling	256
MAC/XMAC	Message Authentication Code	64
RES/XRES	Response/Expected Response	64
T/Actcode	Timestamp/Activation code for USIM	64

the IMSI of the UE. After that the MME connects to the HSS via S6a interface and verifies the IMSI of the UE through a permission message [107].

The HSS generates a new AV and sends it to the MME. This generated authentication vector consists of a random number RAND, authentication token AUTN, signed response XRES, and K_{ASME} key. Both, the HSS and the UE, maintain a counter for the synchronization purpose. A major improvement in EPS-AKA compared to UMTS-AKA is that cipher key CK and integrity key IK are never actually sent by the HSS over the network. The UE sends signals to the MME with the type of access network it uses in the initial message (for Evolved-Universal Terrestrial Radio Access Network (E-UTRAN), set AMF = 1). The K_{ASME} is stored on the MME. The benefit of doing this is that it does

Table 5.2: Cryptographic functions definition

Function	Definition
f_1	Function to generate TID
f_2	Function to generate MAC/XMAC
f_3	Function to generate RES/XRES
f_4	Key generation function for CK
f_5	Key generation function for IK
f_6	Key generation function for AK
f_7	Key generation function for K_{ASME}
f'	Function to generate $XKSI_{ASME}$
fun()	Function to generate K_{ASME}
EK{}/DK{}	Functions to cipher/decipher message
$\|\|$	Concatenation

not require to execute full AKA protocol again, when re-synchronization of the UE is required. Next, the MME sends RAND and AUTN to the UE, and it waits for the response. The MME also sends a key set identifier eKSI to the UE through Evolved NodeB (eNB). There are two eKSI types, one is KSI_{ASME} and other is KSI_{SGSN}. The KSI_{ASME} is used to indicate a native EPS security context while the KSI_{SGSN} is used to indicate a mapping security context. The EPS-AKA protocol provides mutual authentication between the UE and the network. On receipt of the message, the UE computes RES and sends it to the MME. Thereafter, the MME compares this RES with the computed XRES. The UE gets authenticated, only if both are same. The HSS sends initial keys to the MME and the eNB, which are then used by these entities to derive actual keys for NAS signaling, user plane, and RRC signaling.

5.2.2 EPS Key Hierarchy

The EPS key hierarchy in 4G LTE network provides the security protection over the different traffic and consists of a set of symmetric secret keys, which are organized according to the security context of the network. The K key is shared between the UE and the HSS. This key is stored onto the Universal Subscriber Identity Module (USIM) at the time of manufacturing and onto the AuC of the HSS. The K key is used to generate two keys: cipher key CK and integrity key IK. There are two deriving keys, one is the master key K_{ASME} and other is over the air root key K_{eNB}. All the derived keys are 256 bits in size, however, the session keys use only least significant bits (LSB) of size 128.

The K$_{ASME}$ is derived from CK and IK by the HSS and is sent to the MME, which is used to derive different keys for the protection of three different traffic flows. The confidentiality and integrity protection in NAS signaling between the UE and the MME is provided by K$_{NASenc}$ and K$_{NASint}$ keys respectively. The traffic confidentiality and integrity protection between the UE and the eNB is covered by K$_{RRCenc}$ and K$_{RRCint}$ keys respectively, while the user plane data between the UE and the eNB is protected by K$_{UPenc}$ key. Here, the NAS and the RRC signaling integrity protection are provided by AES or SNOW 3G algorithm [108]. Similarly, NAS signaling, RRC signaling, and User Plane ciphering are also provided by AES and SNOW algorithm.

5.3 Proposed Protocol

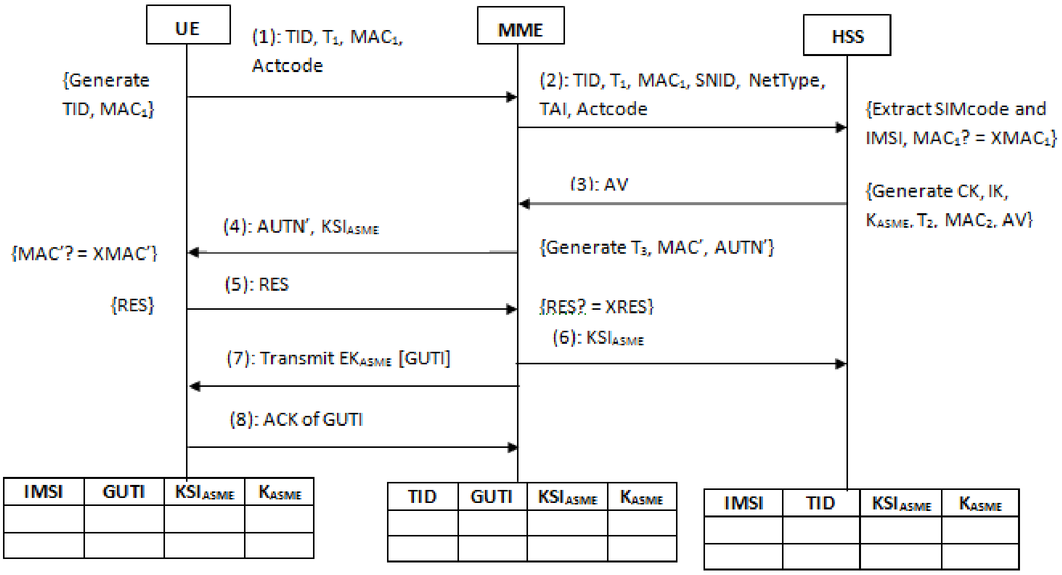
This section presents an improved AKA protocol, which looks after the shortcomings of the EPS-AKA protocol and prevents various threats and attacks of 4G LTE network. The purpose of eNB node and other elements in proposed protocol are similar to the original LTE EPS-AKA protocol.

5.3.1 Security Goals

In order to resist various threats and attacks, numerous alternative approaches have been designed. There are many research groups, which are working on the design of security architectures for 4G networks including Y-Comm and Hokey. Major security challenges in 4G heterogeneous networks [109] are inherited from the internet security threats and IP security vulnerabilities including user-id theft, man-in-the-middle attack, key-id theft, and impersonation attack. Therefore, it is recommended to design a security solution in terms of an improved AKA protocol, which should be independent of the network providers and end user devices.

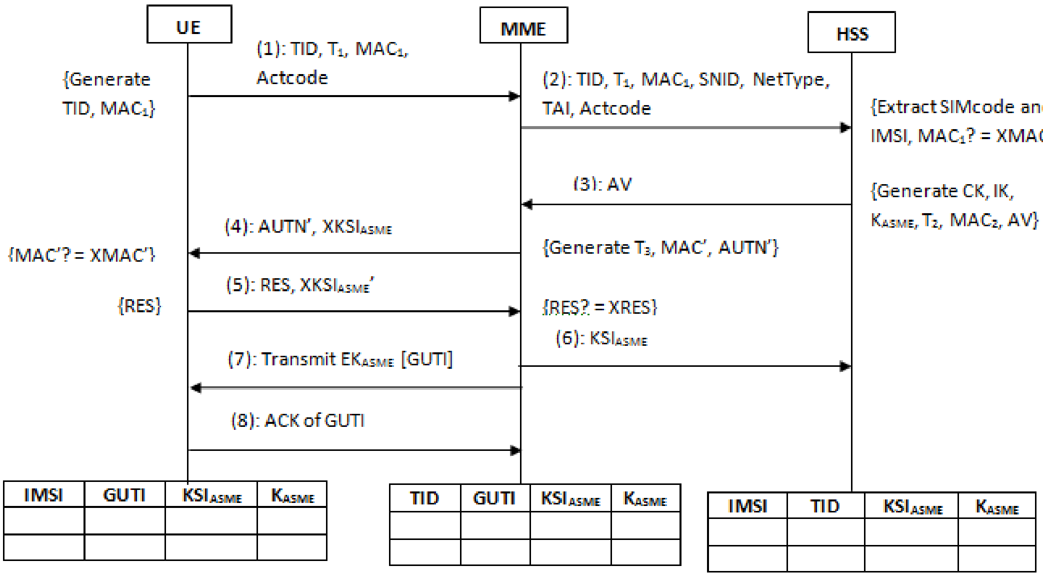
5.3.2 Attack Model

The problems of clear text transmission of IMSI and KSI$_{ASME}$ may result in various possible attacks. Here, some of these scenarios are considered in order to throw the light on various possible attacks in the LTE network. A man-in-the-middle attack can occur when a UE try to connect to an eNB/MME and also in the case that requires the transmission of the IMSI in the first initial request. An adversary puts itself in between the target user and a genuine network, and can capture, modify, eavesdrop, and spoof signaling and data exchanged between both the parties. Nowadays, due to access availability of phone number catcher in the market, it becomes easy to catch the phone number or IMSI over the air by this attack. This vulnerability opens the door for a man-in-the-middle attack that can take place once the IMSI is captured. Other vulnerability is that the EPS-AKA protocol lacks in Perfect Forward Secrecy (PFS) support, which assures the secrecy of a set of session keys, even if a previous set of keys has been compromised. User-identity transmitted in clear text during the initial request procedure of EPS-AKA compromises the entire system. Creating security gap exploitation allows an eavesdropper to track the user location, which results in user-id theft attack. The attacker can send signaling and/or user data to the receiver in order to makes the receiver believe that it is originating from a genuine source, which leads to the impersonation attack. The attacker can also capture the subsequent or derived keys with the help of parent key or based on some information about the parent key and its identity, which causes key-id theft attack.


UE
MME
HSS
{Generate TID, MAC₁}
(1): TID, T₁, MAC₁, Actcode
(2): TID, T₁, MAC₁, SNID, NetType, TAI, Actcode
{Extract SIMcode and IMSI, MAC₁? = XMAC₁}
(3): AV
{Generate CK, IK, K_ASME, T₂, MAC₂, AV}
(4): AUTN′, KSI_ASME
{Generate T₃, MAC′, AUTN′}
{MAC′? = XMAC′}
(5): RES
{RES}
{RES? = XRES}
(6): KSI_ASME
(7): Transmit EK_ASME [GUTI]
(8): ACK of GUTI
IMSI | GUTI | KSI_ASME | K_ASME
TID | GUTI | KSI_ASME | K_ASME
IMSI | TID | KSI_ASME | K_ASME

$Actcode = LCS_n(T_1 \text{ XOR } TAI \text{ XOR } SIMcode)$, $SIMcode = RCS_n(T_1 \text{ XOR } TAI \text{ XOR } Actcode)$, $TID = f_1(IMSI, T_1)_{SK}$, $MAC_1 = XMAC_1 = f_2(T_1, TID, TAI, Actcode)_{SK}$, $AV = (AUTN, K_{ASME})$, $AUTN = (T_2, AMF, MAC_2)$, $MAC_2 = XMAC_2 = f_2(T_2, AMF)_{CK}$, $AUTN' = (T_3, MAC', AMF)$, $MAC' = f_2(MAC_2, T_3, AMF, KSI_{ASME})K_{ASME}$, $XMAC' = f_2(f_2(T_2, AMF)_{CK}, T_3, AMF, KSI_{ASME})K_{ASME}$, $RES = XRES = f_3(T_3)K_{ASME}$

Figure 5.2: Proposed AKA protocol for 4G LTE network (case-1)


UE
MME
HSS
{Generate TID, MAC₁}
(1): TID, T₁, MAC₁, Actcode
(2): TID, T₁, MAC₁, SNID, NetType, TAI, Actcode
{Extract SIMcode and IMSI, MAC₁? = XMAC₁}
(3): AV
{Generate CK, IK, K_ASME, T₂, MAC₂, AV}
(4): AUTN', XKSI_ASME
{Generate T₃, MAC', AUTN'}
{MAC'? = XMAC'}
(5): RES, XKSI_ASME'
{RES}
{RES? = XRES}
(6): KSI_ASME
(7): Transmit EK_ASME [GUTI]
(8): ACK of GUTI
IMSI | GUTI | KSI_ASME | K_ASME
TID | GUTI | KSI_ASME | K_ASME
IMSI | TID | KSI_ASME | K_ASME

Actcode = LCS_n(T_1 XOR TAI XOR SIMcode), SIMcode = RCS_n(T_1 XOR TAI XOR Actcode), TID = f_1(IMSI, T_1)$_{SK}$, MAC_1 = $XMAC_1$ = f_2(T_1, TID, TAI, Actcode)$_{SK}$, AV = (AUTN, K_{ASME}), AUTN = (T_2, AMF, MAC_2), MAC_2 = $XMAC_2$ = f_2(T_2, AMF)$_{CK}$, AUTN' = (T_3, MAC', AMF), MAC' = f_2(MAC_2, T_3, AMF, $XKSI_{ASME}$)K_{ASME}, XMAC' = f_2(f_2(T_2, AMF)$_{CK}$, T_3, AMF, $XKSI_{ASME}$)K_{ASME}, RES = XRES = f_3(T_3, $XKSI_{ASME}$')K_{ASME}, KSI_{ASME} = f'($XKSI_{ASME}$, $XKSI_{ASME}$')

Figure 5.3: Proposed AKA protocol for 4G LTE network (case-2)

5.3.3 Protocol Design

The proposed protocol solves the problem of clear text transmission of IMSI and KSI$_{ASME}$ over the LTE network. The proposed protocol considers two cases, one where KSI$_{ASME}$ is not protected over the network (in Figure 5.2), and other where this protection is available (in Figure 5.3) and the actual KSI$_{ASME}$ is not sent over the network. Here, both the cases of the proposed protocol are shown in Figure 5.2 and Figure 5.3, which are described as follows:

Case-1: IMSI is not sent but KSI$_{ASME}$ is sent over the network: Initially, the UE sends a request (message (1)) to the MME to establish a connection by transmitting the TID, T$_1$, Actcode, and MAC$_1$, where TID is the temporary ID, T$_1$ is timestamp, Actcode is the generated activation code for the UE, and MAC$_1$ is the calculated integrity code (MAC$_1$ = f$_2$(T$_1$, TID, TAI, Actcode)$_{SK}$).
UE to MME: TID, T$_1$, MAC$_1$, Actcode

The UE first generates TID from a function f$_1$ (secret and shared between the UE and the HSS only) with SK key with input as IMSI and T$_1$ (TID=f$_1$(IMSI, T$_1$)$_{SK}$). This f$_1$ is a symmetric function such that we can retrieve the original IMSI, when we pass TID and T$_1$ to as input this function with SK key

(IMSI=f_1(TID, T_1)$_{SK}$). Here, SK key is the secret key shared between the USIM and the HSS, stored on to the USIM at the time of manufacturing and in the database of authentication center of the HSS. The Actcode is the one time activation code, which is sent to the HSS. The purpose of this code is to retrieve and verify the SIMcode. It is assumed that the SIMcode is generated and stored onto the USIM and on the authentication center, when a SIM card gets activated. This SIMcode is attached attaches as a label to the SK key in the authentication center of the HSS.

The generation of Actcode at UE and SIMcode at HSS are not publically available and are secret in nature. This can be achieved as follows: *at UE:* Actcode = LCS$_n$(T_1 ⊕ TAI ⊕ SIMcode), and *at HSS:* SIMcode = RCS$_n$(T_1 ⊕ TAI ⊕ Actcode), where LCS$_n$ = Left Circular Shift by n, RCS$_n$ = Right Circular Shift by n, n = value of first digit of TID (in decimal, convert TID from bits to decimal and pick first leftmost decimal digit), however, generation of such code is operator specific.

The MME passes the message received from the UE to the HSS with the additional information Serving Network ID (SNID) and the Network Type (Net-Type), *i.e.*, E-UTRAN (message (2)).

MME to HSS: TID, T_1, MAC_1, SNID, NetType, TAI, Actcode

On receiving the message, the HSS first checks whether $T_1 < T_{current}$, if yes, then the HSS checks the integrity of the received message by computing XMAC$_1$ and compares it with the received MAC$_1$ (MAC$_1$?= XMAC$_1$), where XMAC$_1$ = f_2(T_1, TID, TAI, Actcode)$_{SK}$. If it holds, then the HSS computes SIMcode from the received Actcode and compares it with the stored value of SIMcode. If both SIMcode match, then the SK key corresponding to that SIMcode is retrieved. Thereafter, the HSS retrieves IMSI through function f_1 with SK key. If the computed IMSI is same as the stored IMSI, then the HSS computes authentication vector otherwise, the request is discarded and the connection is terminated. Afterwards, the HSS generates T_2, CK, IK, K$_{ASME}$, AMF, and MAC$_2$ (MAC$_2$ = f_2(T_2, AMF)$_{CK}$) as a part of AV and sends the AV to the MME (message (3)).

HSS to MME: T_2, AMF, MAC_2, K_{ASME}

On receiving the message from the HSS, the MME generates T_3, MAC′, and AUTN′, where MAC′ = f_2(MAC$_2$, T_3, AMF, KSI$_{ASME}$)$_{K_{ASME}}$. Then, the MME sends AUTN′ to the UE along with a generated KSI_{ASME} (message (4)).

MME to UE: T_3*, MAC′, AMF,* KSI_{ASME}

After receiving the message from the MME, the UE generates CK, IK, and K_{ASME}, computes $\text{XMAC}'=\text{f}_2(\text{f}_2(\text{T}_2, \text{AMF})_{CK}, \text{T}_3, \text{AMF}, \text{KSI}_{ASME})_{K_{ASME}}$ and compares whether MAC′ ?= XMAC′. If such a condition does not hold, then the connection is terminated otherwise, the UE computes signed response RES and sends it to the MME (message (5)).

UE to MME: RES

Note that XKSI_{ASME} at UE is same as received KSI_{ASME} from the MME. Next, on receiving the message from the UE, the MME again computes XRES ($\text{XRES} = \text{f}_3(\text{T}_3)_{K_{ASME}}$) and compares it with the received $\text{RES} = \text{f}_3(\text{T}_3)_{K_{ASME}}$. If both are same, then the MME transmits KSI_{ASME} to the HSS (message (6)).

MME to HSS: KSI_{ASME}

All the UE, MME, and HSS store the values of K_{ASME} and KSI_{ASME} in their memory or database space. After completion of authentication proces,s the MME sends a message (7) to the UE with the encrypted GUTI.

MME to UE: $E[GUTI]_{K_{ASME}}$

Finally, the UE acknowledges the receipt of GUTI to the MME (message (8)).

UE to MME: ACK of GUTI

Case-2: IMSI as well as KSI_{ASME} *are not sent over the network:* There is a possibility to prevent the transmission of actual KSI_{ASME} over the network, which improves the drawback of EPS-AKA of lacking in PFS support. Additionally, the keys, based on unique KSI_{ASME}, are generated only once, thus, a new K_{ASME} is generated every time, when there is a request for the AV, which initiates a new session. Hence, the generation of new secret keys is no longer depending upon the previous keys and other parameters. Such a scenario is illustrated in Figure 5.3, where first three steps are same as in the previous protocol discussed in Figure 5.2. The MME sends T_3, MAC′, AMF, and XKSI_{ASME} to the UE (message (4)).

MME to UE: T_3*, MAC′, AMF,* $XKSI_{ASME}$

In such a case, after receiving a message (4) from the MME, the UE also generates a new XKSI'_{ASME} and computes KSI_{ASME} by considering XKSI_{ASME} (previously generated by the MME) and XKSI'_{ASME} as input to a predefined function shared between the MME and the UE. Then UE transmits the message (5) to the MME with a new calculated RES′ and XKSI′$_{ASME}$, where RES′ = $f_3(T_3$, XKSI′$_{ASME})_{K_{ASME}}$.

UE to MME: RES, XKSI′$_{ASME}$

After receiving the message from the UE, the MME checks the integrity of message by computing XRES′ (XRES′=$f_3(T_3$, XKSI′$_{ASME})_{K_{ASME}}$ and compares it with the received RES′. If both are equal, then the MME calculates a new KSI$_{ASME}$ by taking input of XKSI$_{ASME}$ and XKSI′$_{ASME}$ in the shared function f′. Thus, the UE and the MME have the same KSI$_{ASME}$ value without transmitting the actual value over the network. Afterward, the MME transmits the actual value of KSI$_{ASME}$ to the HSS (message (6)).

MME to HSS: KSI$_{ASME}$

All the UE, MME, and HSS store the values of KSI$_{ASME}$ and K$_{ASME}$ in their memory. After completion of authentication process, the MME sends message (7) to the UE having the encrypted GUTI.

MME to UE: E[GUTI]$_{K_{ASME}}$

Finally, the UE acknowledges the receipt of GUTI to the MME (message (8)).

UE to MME: ACK of GUTI

Roaming Tracking Area Update Procedure: First, a TAU request is sent to the new MME via eNB including the RRC parameters (selected networks and old Globally Unique Mobility Management Entity Identifier (GUMMEI)). Then, eNodeB selects the appropriate MME based on the GUMMEI. Thereafter, new MME uses the received GUTI from the UE to search the old MME-SGSN address and sends a context request message to the old MME. After this, the old MME replies a context response to the new MME. If the integrity of message violates at any stage, then it is mandatory to execute the authentication protocol between the UE and the HSS. New MME acknowledges (context ACK) the received response to old MME and sends a TAU accept message to the UE. Finally, the UE acknowledges the TAU complete message to new MME. This TAU procedure can be executed with and without the Serving Gateway (SWG) changes. In both the procedures, modification of bearer request & response and update location request & response also take place. There is one difference in TAU procedure with serving gateway change that it generates session request and session response before any modification takes place in the settings.

5.4 Security and Performance Analysis of Proposed Protocol

This section discusses security analysis and performance evaluation of the proposed protocol on various parameters.

5.4.1 Mutual Authentication between UE-MME and UE-HSS

In the proposed protocol, the HSS authenticates the UE by verifying MAC_1. To authenticate the HSS, the UE checks the received MAC' from the MME. If MAC' is equal to XMAC', then both, the HSS and the MME are authenticated by the UE. This process ensures the mutual authentication between the UE and the HSS. Next, the MME authenticates the UE by verifying RES. After receiving the message, the MME calculates XRES and checks whether RES is equal to XRES or not. The UE is authenticated, if this equality holds. The same procedure takes place in authenticating the UE when the MME receives RES while communicating with only MME. All this ensures the mutual authentication between the UE-HSS and the UE-MME.

5.4.2 Sequence Number Management

The proposed protocol is also able to solve the synchronization problem as this scheme does not use any sequence number (however, the timestamps used in the protocol prevent replay attack, they are not particularly used for synchronization purpose.)

5.4.3 Resistance to Attacks

In this subsection, we justify that the proposed protocol is free from various attacks and provides the resistance to LTE network.

1. *Replay Attack:* The proposed 4G LTE AKA protocol is free from this attack by sending timestamp T_1, T_2, and T_3 with the message information

over the network.

2. *Man-in-the-middle Attack:* Integrity protection between UE-MME and UE-HHS protects the 4G LTE network from MITM attack. Additionally, the privacy of IMSI and KSI$_{ASME}$ over the network also protects the LTE network from the MITM attack. Apart from this, the message encryption in different domains prevent from this attack, like the K$_{ASME}$ encrypts and sends the GUTI from the MME to the UE. Similarly, K$_{NASenc}$, K$_{RRCenc}$, and K$_{UPenc}$ are used in ciphering the NAS signaling, RRC signaling, and user plane respectively. The ciphering of NAS signaling, RRC signaling, and user plane exist between UE-MME, UE-eNB, and UE-eNB respectively. Thus, this whole scenario successfully prevents the network from MITM attack.

3. *Redirection Attack:* The proposed AKA protocol uses MAC to maintain the integrity of TAI. This concept thereby prevents the network from the redirection attack. This attack is easily possible, when the adversary gets the correct user′s UE information. In the proposed protocol, the UE involves TAI of eNB in MAC$_1$ and transmits MAC$_1$ to the MME. An authentication request is discarded when the HSS fails to match TAI sent by the MME and embedded in XMAC$_1$.

4. *User Identity Attack / User-ID Theft:* The UE transmits user-identity, *i.e.*, IMSI, in clear text during the initial attach procedure of EPS-AKA, which compromises the entire system. However, the proposed protocol defeats this attack.

5. *Impersonation Attack:* In the proposed protocol, MAC$_1$ is computed at UE and sends to the HSS. The HSS then computes XMAC$_1$ with the shared secret key SK between the UE and the HSS. Please note that the HSS computes XMAC$_1$ including the received TAI of UE from the MME while the UE calculates MAC$_1$ based on its current TAI. Additionally, the attacker must reply with a valid response to the MME such that RES ?= XRES in order to impersonate the UE. However, this is not possible for an attacker to have the correct RES. Similarly, an attempt to impersonate the MME will be failed as the UE verifies that the authentication was not

requested by the MME but the UE receives AUTN′ from the MME. Thus, the proposed protocol is able to avoid impersonation attack.

Table 5.3: Summary of various 4G LTE AKA protocols

Prevent Parameters	EPS-AKA	Kϕien [41]	Gu [43]	Chou. [47]	Purkh. [46]	Vintila [37]	Proposed AKA
IMSI to be sent over the network	No	No	Yes	Yes	No	No	Yes
KSI$_{ASME}$ to be sent over network	No	No	No	No	No	Yes	Yes
Replay Attack	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Redirection Attack	Yes	Yes	No	No	Yes	No	Yes
MITM Attack	No	No	No	No	No	Yes	Yes
User-ID Theft	No	No	No	Yes	No	No	Yes
Key-ID Theft	No	No	No	No	No	Yes	Yes
Solve Synchronization Problem	No	No	Yes	No	No	Yes	Yes

6. *Key-ID Theft Attack:* In the proposed protocol, XKSI$_{ASME}$ is sent over the network with the integrity protection and the actual key-id KSI$_{ASME}$ is not transmited over the network (in case-2). This prevents the LTE network from the key-id theft attack.

Table 5.3 summarizes a glimpse of original EPS-AKA, proposed AKA, and various existing protocols with respect to several parameters. Only Vintila [37] prevents the actual KSI$_{ASME}$ to be sent over the network, while Gu [43] and Choudhary [47] prevent the IMSI to be sent over the network. The proposed protocol protects both parameters to be sent over the network. All the existing protocols mentioned in Table 5.3 resist the replay attack. Kϕien [41] and Purkhiabani [46] provide resistance to redirection attack, but do not solve the problem of MITM attack, User-id theft, and Key-id theft. Gu [43] does not solve any issue out of these. The Choudhory [47] only solves user-id theft and still suffers with other three security issues. Vintila [37] is able to prevent MITM and key-id attacks but, does not provide resistance from redirection and user-id attacks. Only Gu [43] and Vintila [37] solve the synchronization issue occurred between the UE and MME/HSS. The proposed protocol provides the resistance from all such attacks and overcomes security issues of the LTE network.

5.4.4 Discussion on Functions Used in Proposed Protocol

The generation of Actcode and SIMcode are secret in nature. The function that generate them is completely depends upon the network operators. If an attacker finds Actcode, he/she cannot obtain SIMcode as they do not know the function and cannot retrieve the SK. Function $f_1()$ is used to generate TID=f_1(IMSI, $T_1)_{SK}$ and IMSI=f_1(TID, $T_1)_{SK}$. Since, SK key is secret, thus, if an attack has a knowledge of function $f_1()$, he/she cannot forge the function as he/she does not know SK. Functions $f_2()$, $f_3()$, $f_4()$, and $f_5()$ are based on HMAC with SHA functions. These functions are one way in nature and nowadays considered secure, thus it is almost impossible to break the security of these functions. Hence, the attacker cannot retrieve any confidential information by these functions.

5.4.5 Performance Analysis

This subsection provides the performance analysis of proposed AKA protocol in terms of storage overhead at UE, MME, and HSS, reduction in bandwidth consumption, and efficiency of new system with respect to various parameters.

1. *Storage Overhead at UE, MME, and HSS:* The proposed protocol has some storage overhead in order to prevent the transmission of IMSI and KSI_{ASME} in clear text over the network. The KSI_{ASME} and K_{ASME} are need to be stored in order to generate different subsequent keys for signaling and data channels. The UE and the MME store (IMSI, GUTI, KSI_{ASME}, K_{ASME}) and (TID, GUTI, KSI_{ASME}, K_{ASME}) respectively, while the HSS stores (TID, IMSI, KSI_{ASME}, K_{ASME}).

2. *Communication Overhead:* We calculate the transmitted message size in order to evaluate total communication overhead for EPS-AKA, proposed AKA, and other existing LTE protocols. All protocols are assumed to be of global sizes with respect to various parameters and only single authentication vector is being transmitted from the HSS to the MME. The total number of bits used by each LTE AKA protocol is as follows:

   a) EPS-AKA Protocol: (1)+(2)+(3)+(4)+(5) = (IMSI+TAI) + (IMSI+ TAI + SNID+ NetType) + (RAND+XRES+ K_{ASME} + ((SQN$\oplus$AK)+ AMF +MAC)) + (RAND+((SQN$\oplus$AK) +AMF+MAC)+ KSI_{ASME}) +

(RES) = 1638 bits

b) Kϕien′s Protocol: (1)+(2)+(3)+(4)+(5)+(6)+(7) = (ID+PI+ RAND) +(ID+PI+RAND+SNID+CID)+(RAND+RES+K$_{ASME}$+CID + AMF + RES) + (RAND+CID+AMF+RES)+(RES+RES$_{HSS}$)+(CID+ RES) + (CID) = 1752 bits

c) Purkhiabani et al.′s Protocol: (1)+(2)+(3)+(4)+(5) = (IMSI+TAI+ MSR+ MSMAC)+(IMSI+TAI+MSR+MSMAC+ SNID+NetType+NSR) + (HEAUTN + HERES+ HEK$_{ASME}$) + (RAND+(HEAK$\oplus$NSR)+AMF +MAC+ KSI$_{ASME}$) + (XRES) = 2019 bits

d) Choudhury et al.′s Protocol: (1)+(2)+(3)+(4)+(5)+(6) = (DMSI) + (AV-req+DMSI)+((ERAND+ XRES+CK+IK+AUTN)+DMSI)+(AUTN + ERAND) + (RES) + (GUTI) = 1890 bits, where AUTN = ERAND+ AMF +MAC = 308 bits

e) Proposed AKA Protocol (Case-1): (1)+(2)+(3)+(4)+(5)+(6)+(7)+(8) = (TID+ T$_1$+ MAC$_1$ + Actcode) + (TID+T$_1$+MAC$_1$+SNID+NetType+ TAI+Actcode) + (T$_2$+AMF+ MAC$_2$+ K$_{ASME}$) + (T$_3$ +AMF+MAC′+ KSI$_{ASME}$)+(RES) + (KSI$_{ASME}$) + (GUTI) + (ACK) = 1644 bits

Proposed AKA Protocol (Case-2): (1)+(2)+(3)+(4)+(5)+(6)+(7)+(8) = (TID+ T$_1$+ MAC$_1$ +Actcode) + (TID+T$_1$+ MAC$_1$ +SNID+NetType +TAI+Actcode) + (T$_2$+AMF+MAC$_2$+ K$_{ASME}$) + (T$_3$+AMF+MAC′+ XKSI$_{ASME}$) + (RES + XKSI′$_{ASME}$) + (KSI$_{ASME}$) + (GUTI) + (ACK) = 1647 bits

Note that the size of ACK is considered 3 bits; however, even if we assume it 8 bits, the proposed protocol is still better in terms of every point of discussion than the EPS-AKA protocol. In fact, 3 bits leads to 8 different combinations, which is more than sufficient because there is only one ACK needs to be sent during the authentication. Thus, the proposed protocol with case-2 (Figure 5.3) is more effective as it is able to hide actual KSI$_{ASME}$ between the UE and the MME. Total bandwidth used by proposed AKA protocol during the authentication as compared to original EPS-AKA, Kϕien′s [41], Purkhiabani′s [46] and Choudhury′s [47] protocols are {(1647/1638)*100 = 100.5%}, {(1647/1752)*100 = 94.0%}, {(1647/2019)*100 = 81.5%}, and {(1647/1890)*100 = 87.1%} respectively. Thus, the proposed AKA protocol reduces -0.5% (almost equal), 6%, 18.5%, and 12.9% bandwidth utilization during the authentication process in comparison to EPS-AKA, Kϕien′s [41], Purkhiabani′s [46], and Choudhury′s [47] protocols respectively. There is just 9 bits overhead difference between the EPS-AKA and the proposed AKA protocol, thus, their overheads can be considered equivalent.

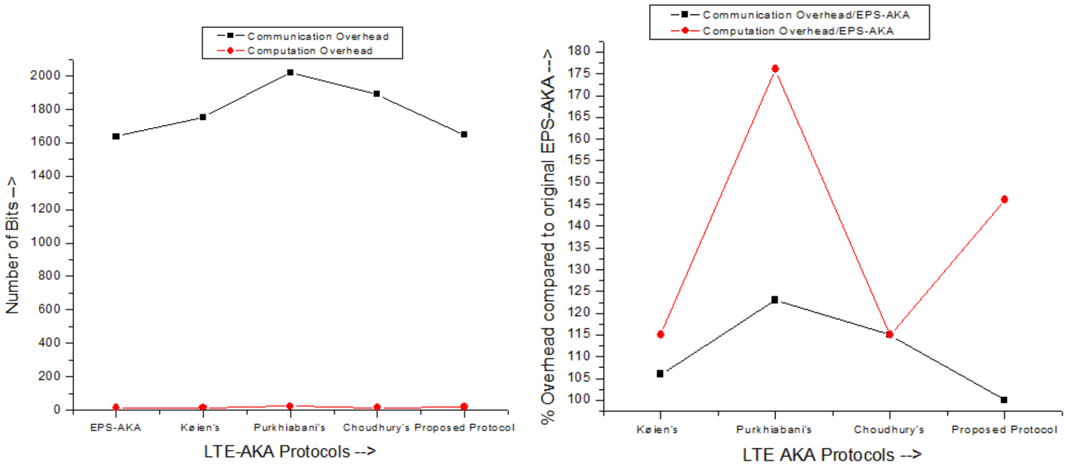


Figure 5.4: Overheads evaluation (a) LTE AKA (b) LTE AKA/EPS-AKA

3. *Computation Overhead:* In order to evaluate computation of each protocol with global values, all the computational overheads of security functions are considered a unit value for ease computation, however, actual weights of these functions may be different.

   a) EPS-AKA Protocol: MAC, AK, CK, IK, XRES, K_{ASME}, (AK⊕SQN), XMAC, (AK⊕SQN), RES, CK, IK, K_{ASME} = 13

   b) Kϕien′s Protocol: $RAND_{UE}$, CID, RAND, K_{ASME}, RES_{UE}, RES, RES_{HSS}, RES_{UE}, RES, RES_{HSS}, K_{ASME} = 11.

   Please note that the K_{ASME} is not derived from CK and IK, thus both keys are not generated in this protocol. An adversary can compromise the security of K_{ASME} due to its generation by non-standard function. If we consider the generation of K_{ASME} by the standard function, then we have (CK, IK) at UE and at MME, thus, the number of functions = 15

   c) Purkhiabani et al.′s Protocol: MSMAC, MSCK, MSIK, MSRES, MSAK, RES, CK, IK, K_{ASME}, HEMAC, HEAK, HECK, HEIK, HERES, HEASME, MAC, XHEMAC, XMAC, XRES, XCK, XIK, MSASME, K_{ASME} = 23

   d) Choudhury et al.′s Protocol: f_i, f_e, f_x, f_n, f_d, f_s, f_m, XRES, CK, IK, K_{ASME}, RES, CK, IK, K_{ASME} = 15 *e) Proposed AKA Protocol:* Actcode, TID, MAC_1, SIMcode, XMAC_1, IMSI, CK, IK, K_{ASME}, MAC_2, K_{ASME}, MAC′, XMAC′, RES, XRES, EK{}, DK{}, CK, IK = 19

Figure 5.4(a) and 5.4(b) represent the communication and computation overheads of all LTE AKA protocols and compare them with the original EPS-AKA protocol.

Table 5.4: Bandwidth consumption by various protocols

Bandwidth Utilization (bits)	EPS-AKA	Kϕien [41]	Purkhi. [46]	Chou. [47]	Proposed-AKA
Between UE-MME	627	676	944	794	697
Between MME-HSS	1011	1076	1075	1096	950

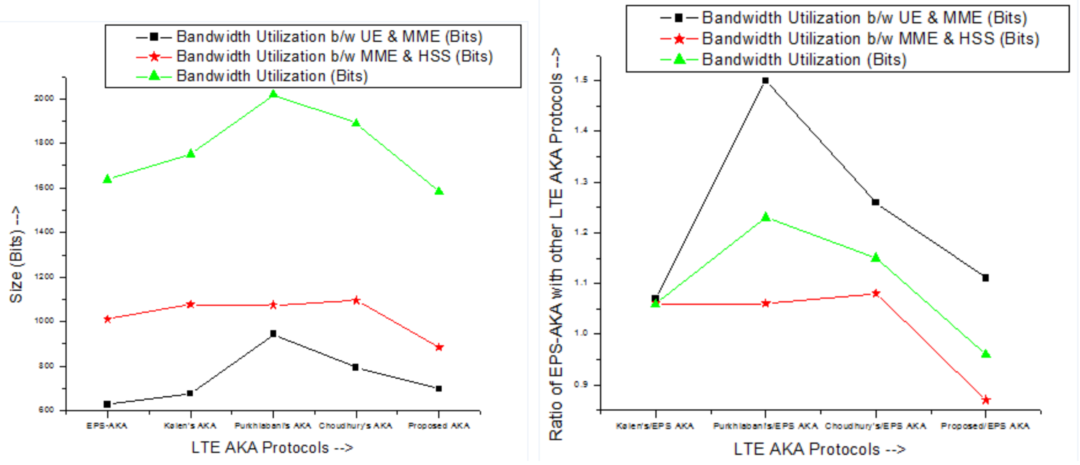


Figure 5.5: Bandwidth consumption (a) LTE AKA (b) LTE AKA/EPS-AKA

4. *Bandwidth Consumption:* The proposed AKA protocol is able to reduce the bandwidth consumption between the MME and the HSS. From Table 5.4, it can be observed that for single authentication, the proposed protocol lowers (100-(950/1011)*100) = 6.1%, 11.8%, 11.7%, and 13.4% of bandwidth utilization between the MME and the HSS as compared to EPS-AKA, Kϕien′s [41], Purkhiabani′s [46], and Choudhury′s [47] protocols respectively.

   Figure 5.5(a) shows bandwidth consumption between UE-MME, MME-HSS, and UE-HSS while Figure 5.5(b) illustrates the bandwidth of other LTE AKA protocols with respect to EPS-AKA. Although, in proposed protocol, the bandwidth utilization between the UE and the MME is increased by about 11% and 3% in comparison to the EPS-AKA and the

Kϕiens [41] protocol (while is reduced by 26.2% and 12.3% with respect to Purkhiabani′s [46] and Choudhury′s [47] protocols), but, we cannot point it out as the limitation because overall in every authentication process, the proposed protocol reduces 105 bits (1752-1647=105 bits) to be transmitted than the Kϕien′s [41] protocol. However, it requires 9 bits (1638-1647=-9 bits) more to be transmitted than EPS-AKA, which is very small value and can be neglected. As per the EPS-AKA protocol, the HSS generates new AV set (or single AV) and sends it to the MME. In both the cases, the proposed protocol provides better service between the HSS and the MME. In every authentication process, the proposed protocol lowers 61 bits and 126 bits with respect to the EPS-AKA and the Kϕiens [41] protocol. The bandwidth consumption of other LTE authentication protocols stated in the literature is not computed because they do not define their parameters clearly.

5.5 Simulation of Proposed Protocol

Java environment is considered in order to simulate the proposed protocol. Functions $f_4()$, $f_5()$, and fun() are implemented as HMACSHA256, while functions $f_2()$ and $f_3()$ are considered as HMACSHA1. However, all these functions are network operator specific. The output of $f_4()$ and $f_5()$ are truncated to 128 bits as these functions are used to generate 128 bits of CK and IK key respectively. Similarly, the output of $f_2()$ and $f_3()$ are also truncated to 64 bits because $f_2()$ produces the output as $MCA_1/MAC_2/MAC'$ of 64 bits in size, while $f_3()$ generates the output as RES of 64 bits. The function to generate Actcode/SIMcode is an XOR function with LCS/RCS. On the other hand, AES is suitable for the encryption/decryption EK{}/DK{} functions with 256 bits K_{ASME} key. The output of function $f_1()$ is 128 bits as the output of this functions is TID/IMSI of 128 bits, thus, we prefer to have $f_1()$ with 128 bits SK key similar to AES encryption (to get TID) and decryption (to retrieve IMSI).

The results mentioned in Tables 5.5 and in Table 5.6 are the average of 30 iterations of each output value. In Table 5.5 and Table 5.6, the unit of time is milliseconds and memory/space is measured in bytes. The message transmission time in the protocol is considered according to the upload and download speed

Table 5.5: Computation for Actcode/SIMcode and cipher functions

Actcode/SIMcode			$f_1()$/EK{}/DK{}			
ExT	PCPUT	TUM	EK{} ExT	TUM	DK{} ExT	TUM
0.89	93.60	12968290	8.8	9329.6	9.1	4816

Table 5.6: Computation for other functions used in proposed protocol

$f_2()/f_3()$			$f_4()/f_5()$/fun()		
ExT	PCPUT	TUM	ExT	PCPUT	TUM
221.60	296.40	15211840	273.41	296.40	15204024

of the LTE network. A post by Graziano D. [110], in June 2013, AT&T was the fastest 4G LTE network that provides average download speed 16.65 Mbps and upload speed 7.43 Mbps, while another company Verizon provides download speed of 10-15 Mbps and upload speed of 4-8 Mbps. Then, as per Hardawar D.′ post [111], in Jan 2014, T-Mobile telecom company has the fastest LTE network in the US that provides on average 17.8 Mbps download speed, while AT&T and Verizon provides 14.7 Mbps and 14.3 Mbps speed respectively. However, UK′s largest mobile company Everything Everywhere Limited (EE) claims to provide world′s fastest 4G LTE network. According to [112], it provides download speed maximum of 120 Mbps, on average 25-30 Mbps in big cities, and 20 Mbps on average over all, while [113] states the 4G EE LTE download speed of 24-30 Mbps and 11 Mbps of upload speed. Thus, in the proposed AKA protocol the transmission time for each message is calculated with upload link speed at 11 Mbps and download link speed at 20 Mbps.

Total messages to be transmitted by the proposed AKA protocol (for case-2): (1)+(2)+(3)+(4)+(5)+(6)+(7)+(8) = (TID+ T_1+ MAC_1+Actcode)+(TID+ T_1+MAC_1+SNID+NetType+TAI+Actcode)+ (T_2+AMF+ MAC_2+ K_{ASME})+ (T_3+AMF+MAC′+ $XKSI_{ASME}$)+($XKSI'_{ASME}$+ RES)+(KSI_{ASME})+ (GUTI) +(ACK)

Proposed protocol at upload link: (1)+(2)+(5)+(6)+(8)=320+515+67+3+3 = 908 bits

Proposed protocol at download link: (3)+(4)+(7) = 432+179+128 = 739 bits

Total time at upload link (speed 11 Mbps, per bit 90.9 nanoseconds) = 908*90.9 = 82537.2 nanoseconds = 0.08 milliseconds

Total time at download link (speed 20 Mbps, per bit 50 nanoseconds) = 739*50

= 36950 nanoseconds = 0.03 milliseconds

Thus, total transmission time for all messages in the proposed AKA protocol = 0.08+0.03 = 0.11 milliseconds. The execution time, process CPU time, and total memory usage for the function to generate Actcode/SIMcode, and encryption/decryption time and memory used for EK{}/DK{} and $f_1()$ are presented in Table 5.5. Similarly, the execution time, process time, and memory usage for functions $f_2()/f_3()/f_4()/f_5()$ can be observed from Table 5.6. For the encryption and decryption of GUTI in proposed protocol, we prefer to use AES algorithm, which is considered one of the best algorithms. It has been found that AES takes 8.8 milliseconds for EK{} and 9.1 milliseconds for DK{} on Java Micro Edition (J2ME) Wireless Toolkit (WTK) platform, while it performs 40 milliseconds to encrypt (EK{}) and 18 milliseconds to decrypt (DK{}) a single message of 1120 bits on JDK1.6. Here, it is preferred to include the execution time obtained on J2ME than on JDK as later is affected by the total memory and the time of other processes running on PC. Since, in the proposed protocol, the transmission time for all the messages sent along with the computation time for each function used are known, thus, it is easy to calculate total time required for the execution of proposed AKA protocol.

Total execution time for proposed protocol = Total messages transmission time + 2*Actcode/SIMcode function + $f_1()$ for TID + $f_1()$ for IMSI + 2*$f_2()$ for MAC_1 + $f_2()$ for MAC_2 + 2*$f_2()$ for MAC' + 2*$f_3()$ for RES + 2*$f_4()$ for CK + 2*$f_5()$ for IK + 2*$f_2()$ for K_{ASME} + EK{} + DK{}

Total execution time for proposed protocol = 0.11 + 2*0.89 + 8.8 + 9.1 + 2*221.60 + 221.60 + 2*221.60 + 2*221.60 + 2*273.41 + 2*273.41 + 2*273.41 + 8.8 + 9.1 = 0.11 + 1.78 + 8.8 + 9.1 + 443.20 + 221.60 + 443.20 + 443.20 + 546.82 + 546.82 + 546.82 + 8.8 + 9.1 = 3229.35 milliseconds = 3.23 seconds

5.6 Formal Proof of Proposed AKA Protocol

We use the BAN-Logic symbols to formally proof the authentication process of proposed protocol. The notations of BAN-Logic are presented in section 3.7.4.

1. *Security Analysis:*

   (1) UE → MME: $UE |\equiv \#(Ta)$, $MME \triangleleft f_1(IMSI, Ta)_{SK}$, $f_2(Ta, TAI, f_1(IMSI, Ta)_{SK})_{SK}$, Actcode

(2) MME → HSS: HSS $\triangleleft$ Ta, TAI, $\mathrm{f}_1(\mathrm{IMSI, Ta})_{SK}$, $\mathrm{f}_2(\mathrm{Ta, TAI, f}_1(\mathrm{IMSI, Ta})_{SK})_{SK}$, Actcode; on receiving, the HSS derives IMSI as $\mathrm{f}_1(\mathrm{TID, Ta})_{SK}$ by first computing SIMcode, then retrieves SK; computes $\mathrm{f}_2(\mathrm{Ta, TAI, f}_1(\mathrm{IMSI, Ta})_{SK})_{SK}$ and compares with the received $\mathrm{f}_2(\mathrm{Ta, TAI, f}_1(\mathrm{IMSI, Ta})_{SK})_{SK}$

(3) HSS → MME: HSS $\mid\equiv \#(\mathrm{Tb})$, MME $\triangleleft$ Tb, AMF, K_{ASME}, $\mathrm{f}_2(\mathrm{Tb, AMF})_{CK}$

(4) MME → UE: MME $\mid\equiv \#(\mathrm{Tc})$, UE $\triangleleft$ Tc, AMF, $\mathrm{f}_2(\mathrm{Tc, AMF, XKSI}_{ASME}, \mathrm{f}_2(\mathrm{Tb, AMF})_{CK})_{K_{ASME}}$, XKSI_{ASME}; UE computes $\mathrm{f}_2(\mathrm{Tc, AMF, KSI}_{ASME}, \mathrm{f}_2(\mathrm{Tb, AMF})_{CK})_{K_{ASME}}$ and compares with the received $\mathrm{f}_2(\mathrm{Tc, XKSI}_{ASME}, \mathrm{AMF, f}_2(\mathrm{Tb, AMF})_{CK})_{K_{ASME}}$; UE generates $\mathrm{f}_3(\mathrm{Tc, XKSI}_{ASME}{}')_{K_{ASME}}$

(5) UE → MME: MME $\triangleleft$ $\mathrm{f}_3(\mathrm{Tc, XKSI}_{ASME}{}')_{K_{ASME}}$, $\mathrm{XKSI}_{ASME}{}'$; on receiving the MME derives $\mathrm{f}_3(\mathrm{Tc, XKSI}_{ASME}{}')_{K_{ASME}}$ and compares with the received $\mathrm{f}_3(\mathrm{Tc, XKSI}_{ASME}{}')_{K_{ASME}}$

(6) MME → HSS: HSS $\triangleleft$ XKSI_{ASME}, $\mathrm{f}'(\mathrm{KSI}_{ASME}, \mathrm{KSI}_{ASME}{}')$

(7) MME → UE: $\mathrm{E}\{\mathrm{GUTI}\}_{K_{ASME}}$

(8) On receiving, the UE computes $\mathrm{D}\{\mathrm{GUTI}\}_{K_{ASME}}$, then UE → MME: ACK{GUTI};

2. *Security Assumptions:*

a) It is assumed that SK key is shared between the UE and its HSS.

(1) UE has the secure key SK and UE $\mid\equiv$ UE $\overset{SK}{\leftrightarrow}$ HSS,

(2) HSS has the secure key SK and HSS $\mid\equiv$ UE $\overset{SK}{\leftrightarrow}$ HSS

b) It is assumed that the MME trusts the HSS.

(1) MME $\mid\equiv$ HSS $\mid\Rightarrow$ UE $\overset{SK}{\leftrightarrow}$ HSS,

(2) $\frac{MME \upharpoonright C, HSS \triangleleft C}{HSS \mid\equiv MME \mid\equiv C}$

(3) $\frac{HSS \upharpoonright C, MME \triangleleft C}{MME \mid\equiv HSS \mid\equiv C}$

c) It is assumed that communication between the HSS and the MME is secure.

(1) MME $\mid\equiv$ MME $\overset{C}{\Leftrightarrow}$ HSS, C is conveyance message between the MME and the HSS.

(2) HSS $\mid\equiv$ MME $\overset{C}{\Leftrightarrow}$ HSS, C is conveyance message between the MME and the HSS.

3. *Message Meaning Rule:*

   (1) $\frac{UE|\equiv(UE\overset{SK}{\leftrightarrow}HSS)\wedge(MME\overset{DK}{\leftrightarrow}UE),UE\triangleleft f_1(Ta,TID,TAI)_{SK}}{UE|\equiv HSS\mid\!\sim f_2(Ta,TID,TAI)_{SK}}$

   (2) $\frac{MME|\equiv f_2(Tb,AMF)_{CK}\wedge(MME\overset{DK}{\leftrightarrow}UE),MME\triangleleft f_2(f_2(Tb,AMF)_{CK},Tc,AMF,XKSI_{ASME})_{K_{ASME}}}{MME|\equiv HSS\mid\!\sim f_2(f_2(Tb,AMF)_{CK},Tc,AMF,XKSI_{ASME})_{K_{ASME}}}$

4. *Nonce/Timestamp Verification Rule:*

   (1) $\frac{UE|\equiv\#(Ta),UE|\equiv HSS\mid\!\sim f_2(Ta,TID,TAI)_{SK}}{UE|\equiv HSS|\equiv f_2(Ta,TID,TAI)_{SK}}$

   (2) $\frac{HSS|\equiv\#(Tb)\wedge MME|\equiv\#(Tc),MME|\equiv HSS\mid\!\sim f_2(f_2(Tb,AMF)_{CK},Tc,AMF,XKSI_{ASME})_{K_{ASME}}}{UE|\equiv HSS|\equiv f_2(f_2(Tb,AMF)_{CK},Tc,AMF,XKSI_{ASME})_{K_{ASME}}}$

5. *Jurisdiction Rule:*

   (1) $\frac{UE|\equiv HSS\Rightarrow f_2(Ta,TID,TAI)_{SK},UE\triangleleft MME\mid\!\sim f_2(Ta,TID,TAI)_{SK}}{UE|\equiv MME|\equiv HSS}$

   (2) $\frac{MME|\equiv HSS\Rightarrow f_2(f_2(Tb,AMF)_{CK},Tc,AMF,XKSI_{ASME})_{K_{ASME}},Z}{MME|\equiv HSS|\equiv UE}$, where Z = MME $\triangleleft$ MME $\mid\!\sim$ $\mathrm{f}_2(\mathrm{f}_2(\mathrm{Tb, AMF})_{CK}$, Tc, AMF, $\mathrm{XKSI}_{ASME})_{K_{ASME}}$

6. *Protocol Goals:*

   a) Mutual authentication between UE-MME and MME-HSS:

   UE $|\equiv$ HSS $|\equiv$ MME $\wedge$ MME $|\equiv$ HSS $|\equiv$ UE $\rightarrow$ UE $|\equiv$ MME $\wedge$ MME $|\equiv$ UE; Thus, mutual authentication holds.

   b) Key agreement between the UE and the MME:

   There is a K_{ASME} key between the MME and the UE to provide agreement. UE $|\equiv$ K_{ASME} $\wedge$ #(Tb), since K_{ASME} = fun(CK, IK); MME $|\equiv$ K_{ASME} $\wedge$ #(Tb), since HSS $\rightarrow$ MME $\mid\!\sim$ K_{ASME}

   c) Key freshness between the UE and the MME:

   HSS $|\equiv$ #(Ta) $\wedge$ MME $|\equiv$ #(Ta), MME $|\equiv$ #(Tb) $\wedge$ UE $|\equiv$ #(Tb), K_{ASME} = fun(CK, IK). Thus, key freshness between the UE and the MME holds.

   d) Confidentiality between the UE and the MME:

   For NAS signaling cipher: $\frac{UE|\equiv(UE\overset{K_{NASenc}}{\leftrightarrow}MME),UE\triangleleft fun(Msg)_{K_{NASenc}}}{UE|\equiv MME\mid\!\sim Msg}$ $\wedge$ $\frac{MME|\equiv(MME\overset{K_{NASenc}}{\leftrightarrow}UE),MME\triangleleft fun(Msg)_{K_{NASenc}}}{MME|\equiv UE\mid\!\sim Msg}$

   RRC signaling cipher: $\frac{UE|\equiv(UE\overset{K_{RRCenc}}{\leftrightarrow}MME),UE\triangleleft fun(Msg)_{K_{RRCenc}}}{UE|\equiv MME\mid\!\sim Msg}$ $\wedge$ $\frac{MME|\equiv(MME\overset{K_{RRCenc}}{\leftrightarrow}UE),MME\triangleleft fun(Msg)_{K_{RRCenc}}}{MME|\equiv UE\mid\!\sim Msg}$

   User plane cipher: $\frac{UE|\equiv(UE\overset{K_{UPenc}}{\leftrightarrow}MME),UE\triangleleft fun(Msg)_{K_{UPenc}}}{UE|\equiv MME\mid\!\sim Msg}$ $\wedge$ $\frac{MME|\equiv(MME\overset{K_{UPenc}}{\leftrightarrow}UE),MME\triangleleft fun(Msg)_{K_{UPenc}}}{MME|\equiv UE\mid\!\sim Msg}$

   e) Resistance replay attack between the UE and the MME:

   If the attacker gets # Ta from message (1) or #Tb from message (3) or #Tc from message (4), he/she cannot forge message (1), (3) and (4) because he/she know neither SK nor K$_{ASME}$ key. Since, #Tc will be changed at the next time, hence, the goal of resistance replay attack between the MS and the MME holds.

f) Resistance man-in-the-middle attack between the UE and the MME:
Since attacker neither knows K$_{NASenc}$, K$_{RRCenc}$, K$_{UPenc}$ keys nor encryption algorithm, thus, it prevents the communication from being eavesdropped.

g) Resistance redirection attack between the UE and the MME:
Since, f$_2$(Ta, TAI, f$_1$(IMSI, Ta)$_{SK}$)$_{SK}$ is used to maintain the integrity of TAI, thus, it is safe from the redirection attack.

h) Resistance impersonation attack between the UE and the MME:
(1) Adversary tries to impersonate the UE: Since, f$_2$(Ta, TAI, f$_1$(IMSI, Ta)$_{SK}$)$_{SK}$ is computed at UE and compared at HSS, hence, it avoids the network from impersonation attack. Additionally, the attacker must reply with a valid response f$_3$(Tc, XKSI$_{ASME'}$)$_{K_{ASME}}$ to the MME, but, he/she neither has K$_{ASME}$ nor K$_{NASenc}$, K$_{RRCenc}$ and K$_{UPenc}$.
(2) Adversary tries to impersonate the MME: The integrity value f$_2$(Ta, TAI, f$_1$(IMSI, Ta)$_{SK}$)$_{SK}$ at UE and at HSS is violated. Additionally, if the UE receives f$_2$(Tc, AMF, XKSI$_{ASME}$, f$_2$(Tb, AMF)$_{CK}$)$_{K_{ASME}}$ at any time, then the connection is terminated because the UE had not previously sent any request to the MME.

i) Resistance User-id theft attack between the UE and the HSS/MME:
The UE transmits f$_1$(IMSI, Ta)$_{SK}$ to the MME, *i.e.*, protected user-identity over the network.

j) Resistance Key-id theft attack between the UE and the HSS/MME:
The XKSI$_{ASME}$ is sent over the network with integrity protection and the actual key set-id KSI$_{ASME}$ is not sent over the network.

5.7 Chapter Summary

This chapter proposes an efficient AKA protocol for 4G LTE network.The protocol provides better security to 4G LTE network by resolving the issues of clear text transmission of IMSI from the UE to the MME and transmission of unencrypted KSI$_{ASME}$ from the MME to the UE over the network. The proposed protocol does not propose to maintain a counter at UE as well as at MME, hence, solves the synchronization issue that is present in the original EPS-AKA protocol. This protocol is free from man-in-the-middle attack, user-id theft, key-id theft, replay attack, impersonation attack, and redirection attack. This work reports that for a single authentication vector, the proposed protocol lowers bandwidth consumption between the MME and the HSS by 6.1%, 11.8%, 11.7%, and 13.4% during the authentication process as compared to EPS-AKA, Kϕien′s, Parkhiabani′s, and Choudhury′s protocols. Further, the protocol is able to reduce communication overhead by -0.5% (almost equal), 6%, 18.5%, and 12.9% in comparison to EPS-AKA, Kϕien′s, Parkhiabani′s, and Choudhury′s protocols respectively.

Chapter 6

AKA Protocol for Secure Delivery of VAS using SMS

6.1 Introduction

On December 3, 2013, SMS service completed its 21 years. On December 3, 1992, the world′s first SMS was sent by Neil Papworth from UK through the Vodafone network. The popularity of SMS is increasing day by day as it is being used in many data centric applications including railways enquiry, news alert, mobile banking, health care applications, activation & deactivation of different kind of services, advertisements and much more. The security of SMS is the most important concern during the data transmission over the network. SMS can be transmitted between the SMSC and the MS via SS7 and OTA interface. But unfortunately, SS7 does not provide any security interface to the SMS. The OTA interface provides a wireless security protection with A5/1 or A5/2 algorithm, however, both algorithms A5/1 and A5/2 have already been broken. In this regard, the SMSSec protocol [48] and PK-SIM protocol [49] were proposed by Lo J. L. C. et al. and Rongyu H. et al. respectively, however, it is observed that these protocols generate large communication and computation overheads. Additionally, these protocols fail to justify their performance in terms of bandwidth utilization and do not discuss about the prevention against different threats and attacks. Thus, all these issues must be taken into consideration and a secure and efficient protocol should be developed for end-to-end secure transmission of SMS over the network.

Table 6.1: Symbols and abbreviations

Symbol	Definitions	Bits
U/ID_ME	Cell Phone Number	40
C/ME	Mobile Station	—
S/AS	Authentication Server	—
SAG	Security Access Gateway	—
H/HU/H_{un}	Hash function	64
D/D_n	Parameter to generate Key	64
Q/Q_n	New Session Identifier	28
Rc*/Nc/Ns/Na/Na′	Random Number	128
Pf	Private Port Number	16
M	Message	Variable
SQ/Seq	Sequence Number	28
PK**/PK_PK-SIM	Public key of server	128
SK***/SK_n/ SK_SAG/SK_AS-CA	Symmetric key	128
UAKey	Primary key	128
Expiry/ExpT	Expiry of Primary key	64
C_ME	Certificate of ME	40
CertSAG	Certificate of SAG	40
MAC/MAC_1/MAC_2	Message Authentication Code	64
T_1/T_2/T_i	Timestamp	64
DK	Delegation key	128

6.2 Attack Model

This section discusses the wireless delivery system of SMS in brief along with different scenarios for the possibilities of various attacks. The wireless delivery of SMS takes place through various radio channels. The air interface has divided into two classes of logical channels: Control Channel (CCH) and Traffic Channel (TCH). TCH channel transmits voice traffic once the call setup is done while CCH carries network statistics information and assists in data (SMS) transmission. A message is broadcasted on Paging Channel (PCH) to send a natification to a targeted device regarding the availability of a call or SMS. Afterward, available devices in reachable network inform their status to network and accept incoming communications using slotted ALOHA-based Random Access Channel (RACH) uplink. Then, a Standalone Dedicated Control Channel (SDCCH) is assigned for the targeted device by listening to the Access Grant Channel (AGCH). If an SMS is available for delivery, then base station authenticates the device, enables encryption, and delivers the content of SMS over the

Table 6.2: Cryptographic functions definition

Function	Definition
$\{\}_{PK}$	RSA_OAEP encryption using PK
$\{\}_{SK}/\{\}_{SK\_n}$	Symmetric key (between MS and AS) encryption using AES
$\{\}_{E\_UAKey}$	Encryption with Primary key UAKey
$\{\}_{SK\_SAG}$	Encryption with Symmetric key between the SAG and the CA
$\{\}_{PK\_PK-SIM}$	Encryption with Public key of PK-SIM
$\{\}_{E\_SK}$	Encryption with Symmetric key
$\{\}_{DK}$	Encryption with Delegation key
$\{\}_{SK\_AS-CA}$	Encryption with Symmetric key between AS and CA
f_1	Cryptographic function to generate DK
$\|\|$	Concatenation

assigned SDCCH [70].

When an SMS is sent from the MS to the AS, it follows the path as MS-BTS-BSS-MSC-SMSC-SMSGateway-AuthenticationGateway-AS. Since, the SMS is sent in plain text only, thus, the network operators can easily access the content of SMS during its transmission at SMSC. This leads to SMS disclosure and SMS spoofing attacks. The OTA interface between the MS and the BTS is protected by a weak encryption algorithm, thus an attacker can compromise these algorithms to capture the information contained in the SMS or can alter the SMS message. The attacker can also try to cryptanalyze the generated cryptographic keys used in the authentication protocol. The attacker can also perform replay and man-in-the-middle attacks. The SMS messages are unencrypted when sent over the SS7 networks. The cryptographic security protection is not available in Short Message Peer to Peer protocol (SMPP) over the Internet. All these above stated attacks are discussed in section 6.4.2 with their prevention by the proposed SecureSMS protocol.

6.3 Focus on the Proposed Protocol

In this section, various aspects concerning the design of proposed protocol are discussed. These aspects include the choice of cryptographic algorithms and implementation issues. Various symbols and abbreviations used in this chapter are listed in Table 6.1, while all the cryptographic functions used are defined in Table 6.2., where in SMSSec protocol, *Considered 64 bits, **Considered 2048 bits, and ***Considered 256 bits.

6.3.1 Basic Requirements for a Secure Protocol

There are some challenges that must be fulfilled in order to develop an efficient protocol for secure SMS. These challenges include the processing time and computation speed of processor, physical memory size of the MS, SMS structure, and single SMS length. Further, the strong cryptographic algorithms need to ensure the confidentiality and integrity of the SMS messages. Cryptographic computations are costly to perform, mainly when public key cryptography is involved. Although, there are some existing protocols proposed by various researchers based on public key cryptography [50], [52], [54], but, we propose a symmetric key-based protocol because symmetric key algorithms are 1000 times faster than asymmetric key algorithms [114]. We prefer security over efficiency, hence, a careful investigation is required for the selection and deployment of cryptographic algorithms for end-to-end SMS security. The cryptographic algorithms should be completely secure, easy to implement, need low computations, and require less storage for cryptographic keys and algorithms. The choice of algorithm should be in such a way that could possibly improve the security and efficiency of the overall system. There exists a trade-off between security and efficiency of the system, and we have prioritized the security over efficiency. Many symmetric key cryptographic algorithms are available at present and can be implemented with various modes of operations. In this way, the symmetric key algorithms with different modes of operations are recommended to provide security for SMS applications. During the execution of the protocol, some parameters may get inconsistent or the connection failure may occur. If any form of failure occurs, then the protocol must be started from the beginning of authentication phase.

6.3.2 System Model and Design of Proposed Protocol

The architecture of proposed SecureSMS protocol has shown in Figure 6.1. The transmission of secure SMS can be within a single SMSC or between SMSCs. Further, we consider both the cases of SMS transmission, *i.e.*, SMS-deliver as well as SMS-submit. In the SMS-submit, the SMS is transmitted from the MS to the SMSC, while SMS-deliver is the case when the SMS is sent from the SMSC to the MS. In the SMS-delivery, the transmitted data either may be an SMS (which

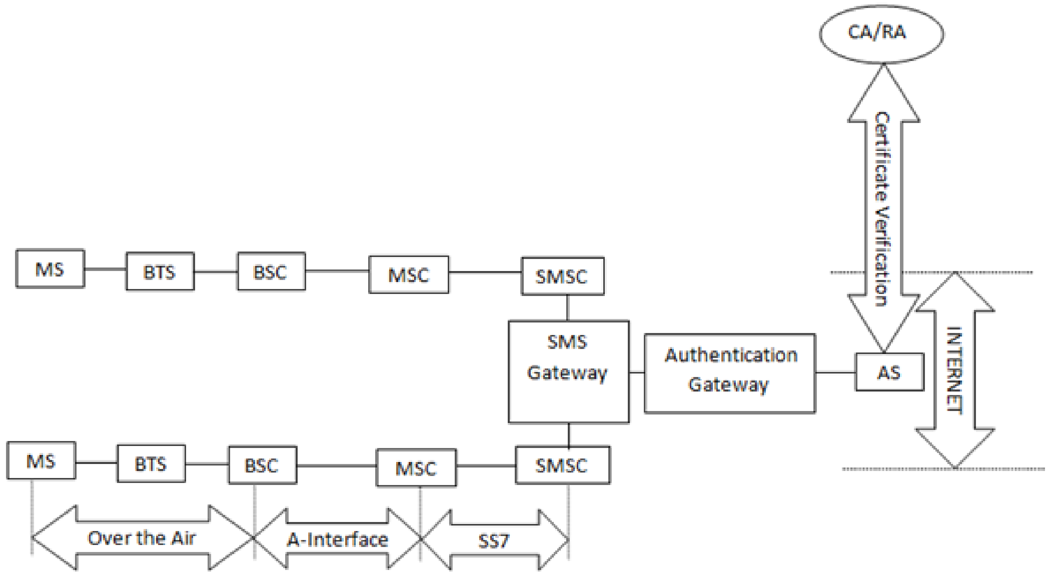


Figure 6.1: Secure transmission of SMS to & fro MS-AS

is being sent to another MS) or may be a delivery report (which is requested by the MS who has sent the SMS and waiting for its acknowledgement). The SMS is moved from the MS to the SMSC over the air interface, A-interface, and through SS7 network. But unfortunately, the security aspect of the transmitted data lacks in the whole path. The SMS is sent from one MS to another MS by one or different SMSCs through SMS gateway and authentication gateway where communication medium is Internet. In this system model, we assume that the identity of each valid and active MS is certified by a Certification Authority/Registration Authority CA/RA.

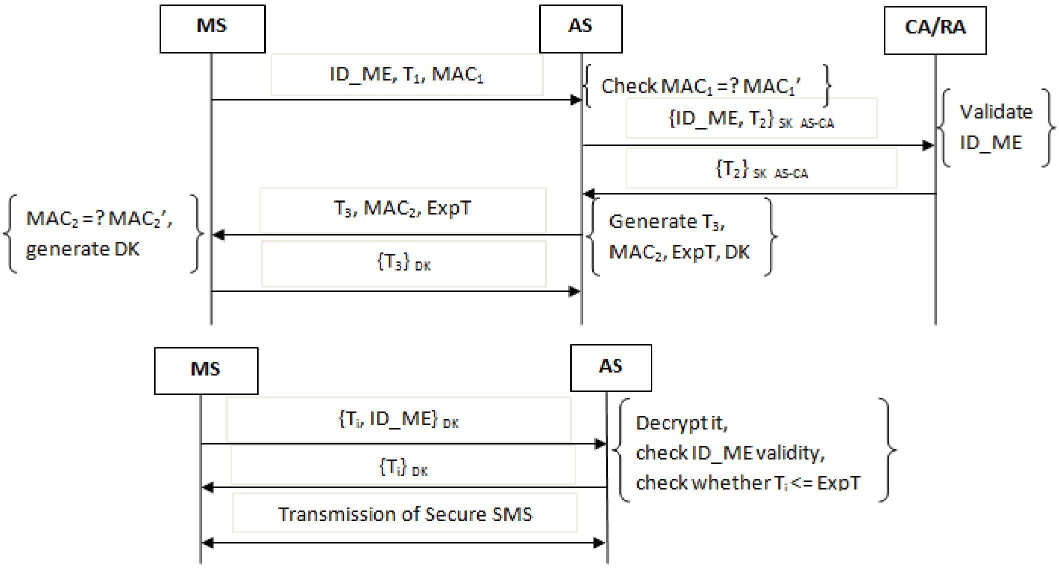

MS
AS
CA/RA
ID_ME, T_1, MAC_1
Check MAC_1 =? MAC_1'
{ID_ME, T_2}$_{SK\ AS\text{-}CA}$
Validate ID_ME
{T_2}$_{SK\ AS\text{-}CA}$
T_3, MAC_2, ExpT
Generate T_3, MAC_2, ExpT, DK
MAC_2 =? MAC_2', generate DK
{T_3}$_{DK}$
MS
AS
{T_i, ID_ME}$_{DK}$
Decrypt it, check ID_ME validity, check whether T_i <= ExpT
{T_i}$_{DK}$
Transmission of Secure SMS

Figure 6.2: SecureSMS protocol (a) phase-1 (b) phase-2

The proposed SecureSMS protocol is shown in Figure 6.2. The protocol is initiated, when any MS wishes to send a message (SMS) to access a value added service. In this process, the very first task is the authentication of MS who wishes to send the SMS. This can be done by the AS with the help of CA/RA authority. The proposed protocol is divided into two phases that are as follows:
Phase-1: The first phase starts with a request to send SMS generated by MS to the AS. In this request the MS sends its identity ID_ME, a timestamp T_1 (the time at which this request is generated), and a message authentication code MAC_1 to maintain the integrity of the information being sent, where MAC_1 = H(T_1, ID_ME).
MS to AS: $\{ID\_ME,\ T_1,\ MAC_1\}$

On receipt of the above message, the AS first calculates MAC1$'$ by the received T_1 and ID_ME, and checks whether MAC_1 is equal to MAC'_1 or not. If not, then the request is discarded otherwise, the AS sends ID_ME and a timestamp T_2 to the CA/RA through the symmetric encryption with shared key between the AS and the CA/RA.
AS to CA/RA: $\{ID\_ME,\ T_2\}_{SK\_AS-CA}$

When the CA/RA receives such message, it checks the validity of ID_ME. If the identity of ME is valid, then the CA/RA sends response $\{T_2\}$ back to the AS otherwise, generates a user invalid message (which is not shown in Figure 6.2, instead of a positive response $\{T_2\}_{SK\_AS-CA}$ is shown).
CA/RA to AS: $\{T_2\}_{SK\_AS-CA}$

If the AS finds ID_ME as a valid response from the CA/RA, it generates a timestamp T_3, a delegation key DK (DK = $f_1(T_1,\ T_3)_{SK}$, an expiry time ExpT for DK key, and MAC_2 (MAC_2 = H(T_1,T_3, ExpT). Now, the AS passes this information to the MS. Here, SK is the secret shared key between the MS and the AS.
AS to MS: $\{T_3,\ ExpT,\ MAC_2\}$

Next, on receipt of above message, the MS checks the integrity of the received information by computing MAC'_2 with the received T_1, T_3, and ExpT. The MS checks whether the computed MAC'_2 is same as the received MAC_2. If not, then simply the request is aborted and the connection is terminated otherwise, the MS computes DK key (DK = $f_1(T_3,\ T_1)_{SK}$), encrypts the response with DK and sends back to the AS. When the AS receives this message, it decrypts the message and checks the stored T_3 with the received T_3. If both are same, then the authentication is successful.

MS to AS: $\{T_3\}_{DK}$

Phase-2: Now, for the subsequent authentication, every time the MS wishes to send SMS within the expiry time of DK key, the following procedure takes place:

The MS sends ID_ME along with the current timestamp T_i symmetrically encrypted by DK key to the AS.

MS to AS: $\{T_i, ID\_ME\}_{DK}$

On receipt of message from the MS, the AS decrypts the message and checks the validity of ID_ME whether the MS was previously authenticated within expiry time duration. If not, then the authentication procedure phase-1 takes place otherwise, the AS checks whether the received timestamp T_i is less than or equal to the expiry time of DK key ($T_i <=$ ExpT). If T_i is within the time limit, then the AS replies with encrypted T_i by DK to the MS.

AS to MS: $\{T_i\}_{DK}$

Thereafter, the secure communication takes place between MS-AS or MS-MS depending upon the nature of SMS application. If T_i is not within ExpT, then the MS has to follow the above procedure (phase-1) for a fresh authentication.

6.4 Security and Performance Analysis of SecureSMS Protocol

This section analyzes the proposed protocol in various aspects including security issues, various threats and attacks, and communication and computation overheads.

6.4.1 Mutual Authentication between the MS and the AS

In the SecureSMS protocol, the AS authenticates the MS by verifying MAC_1, where $MAC_1 = H(T_1, ID\_ME)$. On receiving MAC_1, the AS calculates MAC'_1 by T_1 and ID_ME, and then compares MAC'_1 with the received MAC_1. If it matches, then the authentication of MS is done by the AS. Similarly, on receiving MAC_2, the MS computes MAC'_2 to authenticate the AS, where $MCA_2 = H(T_1,$

T_3, ExpT). If $MAC_2 = MAC'_2$ holds, then the authentication of AS is successful. All this ensures the mutual authentication between the MS and the AS.

6.4.2 Resistance to Attacks

In this subsection, we justify that SecureSMS protocol is free from following attacks:

1. *SMS Disclosure:* Nowadays, the transmission of SMS does not provide end-to-end protection of confidentiality and integrity. Thus, the SMS message can be snooped and intercepted during its transmission. The SecureSMS protocol proposes the use of an encryption approach for transmitting the SMS over the network. This encryption approach (AES/Blowfish) provides security to the message information. Thus, this protocol prevents the system from SMS disclosure.

2. *SMS Spoofing:* A malicious user can spoof the legitimate MS by sending an SMS from Internet. If a malicious user knows the secret authenticating information of a legitimate MS, he/she can perform the operation with the legitimate AS. The SecureSMS protocol provides mutual authentication between the MS and the AS. When both authenticate each other, then only the authentication is successfully completed. Thus, no malicious user or attacker can spoof the SMS with SecureSMS protocol.

3. *Replay Attack:* This attack does not work, if an authentication request number (a unique number or random number) is included in the transmitted message. Thus, the proposed protocol is free from this attack because each time it sends a unique timestamp ($T_1/T_2/T_3$) with the information over the network.

4. *Man-in-the-middle Attack:* In the SecureSMS protocol, a symmetric cryptographic algorithm AES is proposed to use between the MS and the AS. The timestamp T_i and ID_ME is encrypted with DK key in every subsequent authentication. Thus, this protocol prevents the communication from being eavesdropped.

5. *Over the Air Modification in SMS Transmission:* The SMS messages are sent OTA interface between the MS and the BTS with A5/1 or A5/2,

however, both algorithms have already proved vulnerable. The proposed protocol provides end-to-end security to the SMS from the sender to the receiver (AS) including the OTA interface with a strong encryption algorithm AES that solves this problem.

6. *Security Issue in SMS Transmission through SS7:* The SMS messages sent over the SS7 networks are unencrypted and the cryptographic security protection is not available in SMPP protocol. Thus, the malicious user can easily read, alter, or delete the message content. Additionally, the SMS messages are stored as plain text in SMSC before they are successfully delivered to the intended recipient, thus, network operator can also access the content of message, which prohibits the system to be privacy protected. The SecureSMS provides end-to-end security so that no attacker could gain secret information from the message that prevents the message privacy even from network operators.

7. *Other Protocol Attacks:* There are some other protocol attacks that have been discussed by Horn G. et al. [56]. The possible applicable attacks and their impacts are as follows:
   a) Signer and Content Verification Attack: The identity of a user can be compromised, if a malicious user obtains a digital signature generated by that user. The malicious user can apply the known public verification keys successively to the signature for the message recovery. The content verification attack can be used to determine the partial message information from a signature that is unknown to the attacker. Here, the malicious user repeatedly guesses the missing message information and attempts to verify the signature with the known user′s verification key. In the SecureSMS protocol, the algorithm used for the digital signature is ECDSA, which is based on the hardness of elliptic curve discrete logarithm problem. Thus, it is not easy for an attacker to recover the signature. Further, the digital signature is applied after encrypting the message, and even if attacker performs the signature verification he would not be able to get the information from the encrypted message. Thus, the proposed protocol is free from these attacks.
   b) Source Substitution Attack: A source substitution attack involves an

attacker to take the public key of another user and obtains a certificate in the name of the attacker for that public key value. This allows the attacker to masquerade and capture the confidential information. In the proposed protocol, certificate verification is performed in cipher mode with the symmetric key shared between the AS and the CA/RA. Thus, there is no such possibility that the attacker could capture the public key of any user.

c) Time-memory Trade-off Attack: A time-memory trade-off attack can be used to determine the data, for which a hash is calculated. The attacker calculates and stores the all possible values of hash code. During the execution of protocol, attacker compares the generated value with the stored values. A hash algorithm (SHA1), used in SecureSMS, generates a hash code of 160 bits. It would be very difficult for an attacker to calculate and store all the possible values of hash code, *i.e.*, 2^{160}. If during the execution of protocol, attacker compares the generated value with all the stored values, he would not get the plain text matches but the cipher matches, which do not guarantee to be same after decryption. Thus, this attack does not affect the SecureSMS protocol security.

d) Codebook Attack: An attacker keeps a record of the data encrypted with the same symmetric key. If the same data are ever encrypted with the same key, then the attacker can identify the set of similar data from the stored record. During a session, the message (T_i, ID_ME) is sent from the MS to the AS and the reply message (T_i) is sent from the AS to the MS encrypted with DK key. The value of T_i changes every time a request is sent from the MS to the AS. Thus, there is no such transmitted message containing same information as in any previous message.

e) Key Separation Attack: Consider a user X who encrypts the message for user Y during the authentication protocol using a symmetric key. Now, as part of a completely different protocol, user X sends the message to user Z encrypted with the same key that have used between user X and Y. Here, it may be possible that user Z could replace the message sent on to user Y; however user X believes the correct execution of the protocol. The DK key is randomly generated from SK key for every connection established for the communication between the MS and the AS (or other MS). A connection is alive for a session based on the expiry time (ExpT). Thus, there would be no such possibility that any attacker could replace the message encrypted with same DK used for another connection. If an attacker tries the same, then that message would be discarded because of the ID_ME mismatch. Thus, this attack is prevented by the SecureSMS protocol.

f) Known Key Attacks: This attack applies when in an event an old session key is compromised and future session keys can also be compromised. There is no such possibility of the attack because DK key is generated for a session (depends upon expiry time ExpT) and no other session key is required in the SecureSMS protocol. If the session expires, the MS or the user has to establish a fresh connection with the AS for the authentication.

6.4.3 Communication Overhead

This subsection calculates the communication overhead for all three protocols.

1. *SMSSec Protocol:* This protocol is divided into two phases. The size of Rc is considered 128 bits as a global parameter value for all protocols, however, it is of 64 bits in SMSSec. The size of various parameters are listed in Table 6.1. Here, we have considered both cases as:

   Case-1: Where random number Rc is 128 bits (as a standard)

   Phase-1: (1)+(2)+(3)+(4) = (40+64+64+28+128)+(128+16+28)+(28) +(28) = 552 bits

   Phase-2: (for n values) = ((1)+(2)+(3)+(4))*n = ((64+40+64+64+ 28+ 128)+(128+16+28)+(28)+(28))*n = 616*n

   Total transmitted bits = Phase-1 + Phase-2 = 552+616*n

   Case-2: Where random number Rc is 64 bits (as assumed in SMSSec Protocol)

   Phase-1: (1)+(2)+(3)+(4) = (40+64+64+28+64)+(64+16+28)+(28)+ (28) = 424 bits

   Phase-2: (for n values) = ((1)+(2)+(3)+(4))*n = ((64+40+64+64+28 +64)+(64+16+28)+(28)+(28))*n = 488*n

   Total transmitted bits = Phase-1 + Phase-2 = 424+488*n

   Thus, this protocol generates 552+616*n bits of communication overhead where n depends upon the number of authentication requests.

2. *PK-SIM Protocol:* We can calculate the overhead for both phases in the following way:

   Phase-1: (1)+(2)+(3)+(4)+(5) = (40+128+64+28)+(40)+(40+64)+(128 +128+64+64)+(128) = 916 bits

   Phase-2: (for n values) = ((1)+(2))*n = (40+128+64)+(128+64) = 424*n

   Total transmitted bits = Phase-1 + Phase-2 = 916+424*n

   Thus, PK-SIM protocol generates 916+424*n bits of communication overhead.

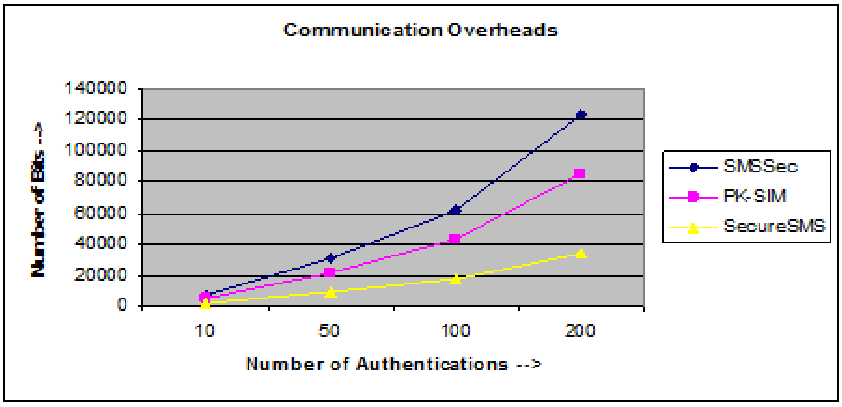


Figure 6.3: Communication overhead for SMSSec, PK-SIM, and SecureSMS

3. *SecureSMS Protocol:* The proposed protocol is also divided into two phases. First phase executes for once, while the second phase executes as many times as the number of authentications are required within the expiry time.

   Phase-1: (1)+(2)+(3)+(4)+(5) = (40+64+64)+(40+64)+(64)+(64+64+ 64+128)+(64) = 656 bits

   Phase-2: (for n values) = (64+40+64)*n = 168*n

   Total transmitted bits = Phase-1 + Phase-2 = 656+168*n

Thus, the SecureSMS protocol generates 656+168*n bits of communication overhead. A comparison of communication overhead among all three protocols is mapped in Figure 6.3. It can be concluded that out of three protocols, the SecureSMS generates lower communication overhead as compared to SMSSec and PK-SIM protocols.

6.4.4 Computation Overhead

This subsection evaluates the computation overhead for all three protocols.

1. *SMSSec Protocol: Phase-1:* [H, $\{\}_{PK}$, $\{\}_{SK}$, $\{\}_{SK}$, $\{\}_{SK}$] = 5

   Phase-2: [H, HU, $\{\}_{SK}$, $\{\}_{SK\_n}$, $\{\}_{SK\_n}$, $\{\}_{SK\_n}$]*n = 5+6*n

   Total functions= Phase-1 + Phase-2 = 5+6*n

   Thus, this protocol generates 5+6*n computation overhead.

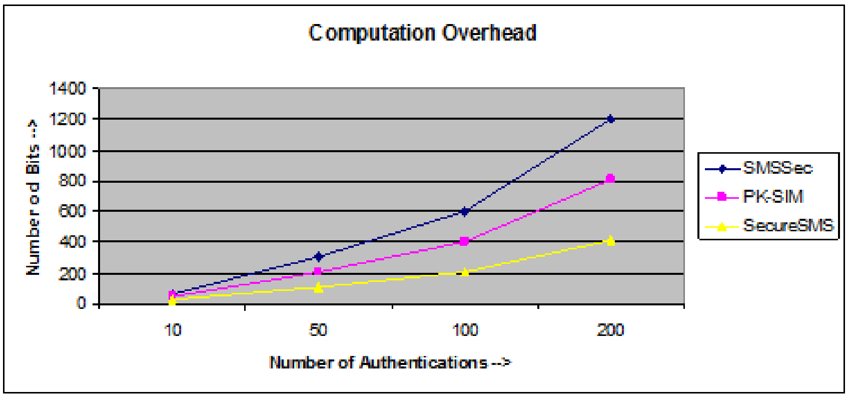


Figure 6.4: Computation overhead for SMSSec, PK-SIM, and SecureSMS

2. *PK-SIM Protocol: Phase-1:* [H(CertSAG), $\{\}_{SK\_SAG}$, H(C_ME), $\{\}_{SK\_SAG}$, H(Ns, Nc, UAKey, Expiry), $\{\}_{SK\_SAG}$, $\{\}_{PK\_PK-SIM}$, $\{\}_{E\_UAKey}$] = 8

   Phase-2: [MAC, {}E_SK, MAC′, {}E_SK]*n = 4*n

   Total functions = Phase-1 + Phase-2 = 8+4*n

   Thus, the PK-SIM protocol generates 8+4*n computation overhead.

3. *SecureSMS Protocol: Phase-1:* [MAC_1, MAC'_1, $\{\}_{SK\_AS-CA}$, $\{\}_{SK\_AS-CA}$, MAC_2, MAC'_2 DK, DK, $\{\}_{DK}$] = 9

   Phase-2: [$\{\}_{DK}$, $\{\}_{DK}$]*n = 2*n

   Total functions = Phase-1 + Phase-2 = 9+2*n

Thus, the SecureSMS protocol generates 9+2*n computation overhead. A comparison of computation overhead among all three protocols is illustrated in Figure 6.4. It can be concluded that out of three protocols, the SecureSMS generates least computation overhead.

6.4.5 Bandwidth Utilization

This subsection calculates the bandwidth utilized by all three protocols and compares them with respect to each other. Table 6.3 presents the bandwidth utilization of SecureSMS with respect to SMSSec and PK-SIM protocols. The number of authentication requests (n) is considered as 10, 50, 100, 200 for both

Table 6.3: Bandwidth utilization of various protocols

No. of Authentication Requests	SecureSMS/ SMSSec	SecureSMS/ PK-SIM
10	0.34	0.45
50	0.28	0.4
100	0.28	0.4
200	0.27	0.39
Average	0.29	0.41

comparisons SecureSMS/SMSSec and SecureSMS/PK-SIM. It is concluded that on an average, the SecureSMS protocol lowers 71% and 59% of the bandwidth utilization during the authentication as compared to SMSSec and PK-SIM protocols respectively.

6.5 Implementation and Results Discussion

This section describes SMS information flow and implementation of SecureSMS protocol.

6.5.1 SMS Information Flow in SecureSMS Protocol

This subsection explains the idea of physical implementation of SecureSMS and the flow of SMS transmission in the GSM network. From Figure 6.2, it is clear that there are three main entities in the SecureSMS protocol, *i.e.*, MS, AS, and CA/RA. For the physical implementation of SecureSMS protocol, we relate the entities presented in the SecureSMS with the real scenario of cellular networks. Here, the MS is a mobile station or mobile user while the AS is referred to as authentication server, which is a part of AuC of the cellular networks. The CA/RA is responsible for verifying the identities of mobile users, while in cellular networks it is handled by the HLR and the VLR. In the proposed protocol, the CA/RA is the connected network of various HLR and VLR. Thus, there is no need of any new entity for implementing the SecureSMS protocol in the cellular networks. All secret keys are stored on the AS in the same way as nowadays the keys are stored in AuC of cellular networks for voice communication. These keys can only be extracted by the process of authentication. No individual (even the network operator) can obtain these secret keys. The network operators can access the SMS at SMSC but they cannot access any content at AS (or AuC). In the cellular network, the flow of SMS from sender to receiver can be understood as follows:

1. The MS′s Short Message Entity (SME) is powered ON.
2. An authentication protocol is executed to verify whether the MS is registered with the network or not (verification is processed through the AS).
3. The MS transfers the SMS to the MSC (through BTS and BSC).
4. The MSC interrogates the VLR to verify that SMS transfer does not violate the control limits forced.
5. The MSC sends SMS to the SMSC.
6. Next, the SMSC interrogates the HLR to verify SMS transfer status.
7. The SMSC sends SMS to the MSC.
8. The MSC transfers SMS to the MS (through BSC and BTS).
9. The SMSC returns a status report indicating delivery of SMS.
10. The SMSC and MSC acknowledge the status of sent SMS to MSC and MS respectively.

6.5.2 Simulation: Secure SMS Communication

The transmission of SMS in the GSM network is proposed with the security services like authentication, confidentiality, integrity, and non-repudiation. The aim of the proposed protocol is to first provide mutual authentication between the MS and the AS, and cipher the SMS, and then digital signature is imposed with the integrity function. (The need of digital signature exists with the use of symmetric key algorithm). In the SecureSMS protocol, the authentication is provided by the AS, and the confidentiality can be maintained by ciphering the message using the symmetric key cryptographic algorithm, like AES or Blowfish. These algorithms have been implemented and simulated using JDK 1.6 and J2ME Wireless Toolkit with Bouncy Castle APIs.

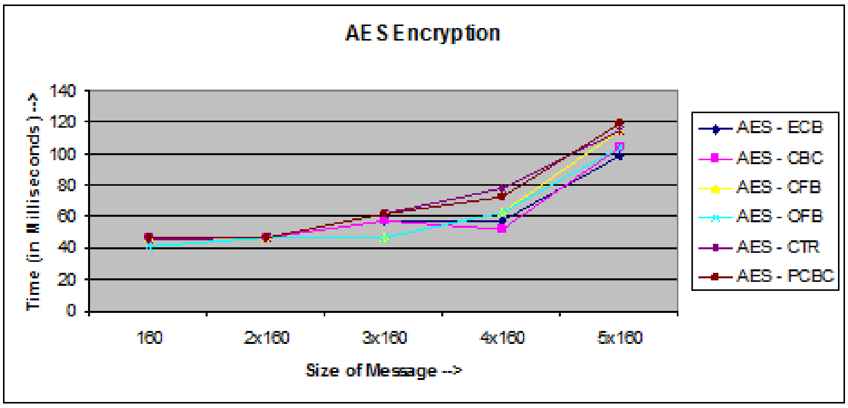


Figure 6.5: AES-encryption with various modes

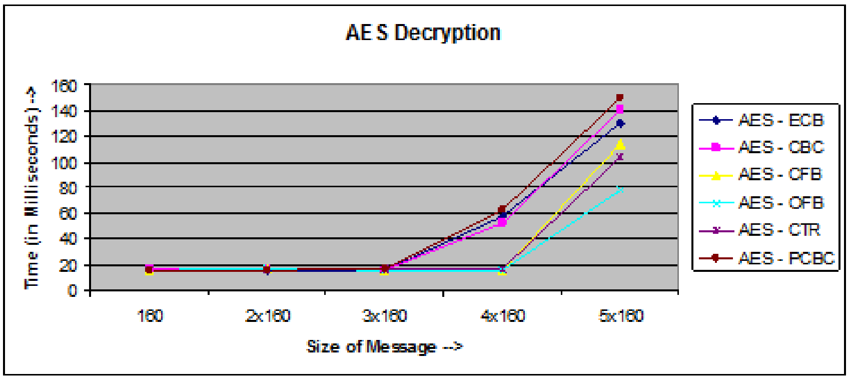


Figure 6.6: AES-decryption with various modes

To analyze and improve the security of the SMS communication system, various modes of operations have been implemented with these algorithms. These various modes of operations are Propagating Cipher Block Chaining (PCBC), Electronic Code Book (ECB), Cipher Block Chaining (CBC), Counter (CTR), Cipher Feedback Block (CFB), and Output Feedback Block (OFB). Figure 6.5, 6.6, 6.7 and 6.8 present different graphs between the execution time to perform encryption/decryption vs. various modes with algorithms. Figure 6.5 and Figure 6.7 represent the encryption with various modes of operations for multiple size SMS by AES and Blowfish respectively. Similarly, Figure 6.6 and Figure 6.8 show the decryption with various modes for multiple size messages by same algorithms. Here, it has considered that the maximum size of transmitted data is 800 characters (700 bytes). Since, the size of a single message (SMS) is 160 characters, thus, this amount of data can be transmitted by concatenating the maximum of 5 messages. The CTR mode is widely accepted and is well suited to the operations performed on a multi-processor machine because the blocks can be encrypted in parallel. After generating the results, it can be clearly

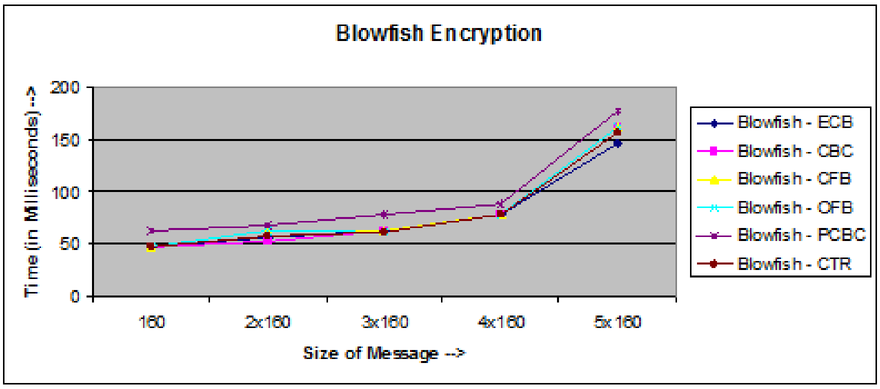


Figure 6.7: Blowfish-encryption with various modes

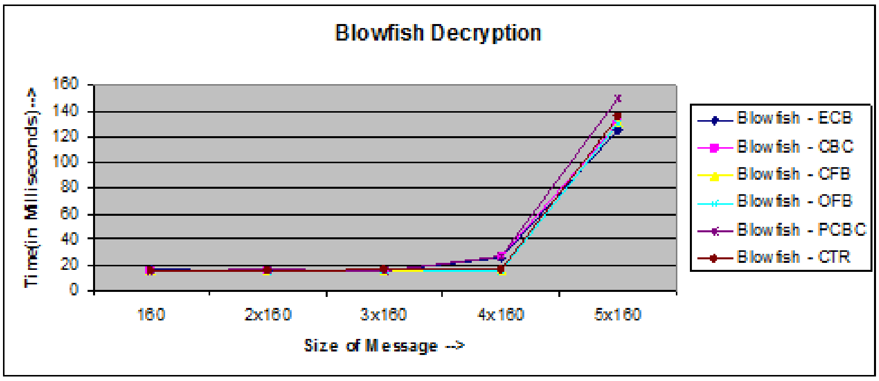


Figure 6.8: Blowfish-decryption with various modes

observed that AES-OFB, AES-ECB, and Blowfish-ECB takes less time for the encryption and decryption. However, these algorithms cannot be considered as a part of the SecureSMS protocol due to lack of security with these modes. On the other hand, AES-CTR mode provides the parallelism and better security for the system. Normally, Blowfish algorithm takes less time to cipher and decipher the message but the main problem with Blowfish is that it consumes more time for key generation. The AES and Blowfish algorithms have been implemented with key size of 128 bits, which are same as the size of DK Key in SecureSMS, *i.e.*, 128 bits. AES with CTR mode algorithm is considered to encrypt/decrypt the messages in the proposed protocol.

Table 6.4 shows the pairs of SMS (original, cipher) having 160 characters in the original message. Here, one can clearly observe that it is able to send exactly 160 characters after ciphering, if the AES or Blowfish with CTR mode is chosen. However, the final selection of an algorithm for the proposed protocol depends on the service providers. The computational efficiency of AES and Blowfish are mentioned in Table 6.5. The AES takes 8.8 milliseconds to encrypt the

Table 6.4: SMS pair size (original, cipher)

Mode/Algorithm	AES	BLOWFISH
PCBC	160, 80	160, 80
ECB	160, 80	160, 80
CBC	160, 155	160, 80
CTR	160, 160	160, 160
CFB	160, 161	160, 164
OFB	160, 82	160, 158

Table 6.5: Comparison of computational efficiency (encryption & decryption)

AES Encryption Time	AES Decryption Time	Blowfish Encryption Time	Blowfish Decryption Time
8.8	9.1	29.1	30.5

message of 160 characters (140 bytes), while decryption takes 9.1 milliseconds for the same. However, the encryption with Blowfish spends 29.1 milliseconds and decryption takes 30.5 milliseconds for single SMS.

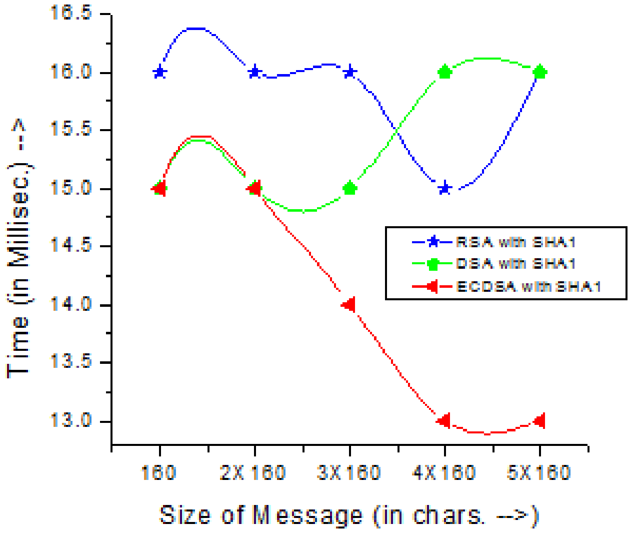


Figure 6.9: Digital signature generation: execution time vs. message size

Table 6.6 describes the memory usage of AES and Blowfish in encryption and decryption. The memory usage of encryption and decryption performed by AES are 9329.6 bytes (9.3 KB) and 4816 bytes (4.8 KB), and by Blowfish are 7816 bytes (7.8 KB) and 4252.8 bytes (4.3 KB) respectively. All these values are calculated on the average of 100 samples each. The integrity of SMS is maintained by using the cryptographic message digest algorithms like MD5 and SHA1. The non-repudiation is provided by imposing the digital signature over the encrypted message. Since, 1024 bits RSA key security is equivalent to the

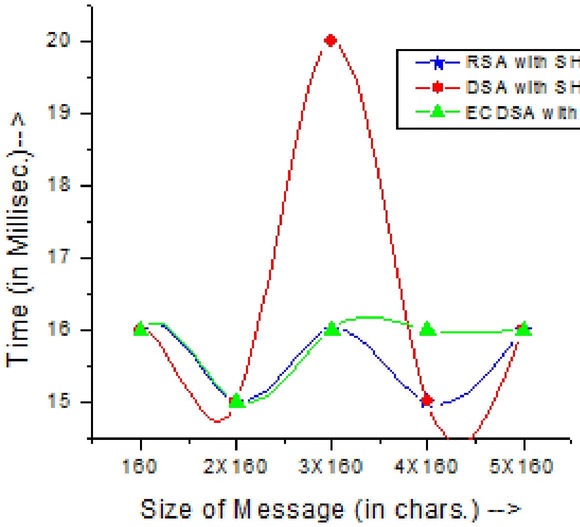


Figure 6.10: Digital signature verification: execution time vs. message size

Table 6.6: Comparison of memory utilization (encryption & decryption)

AES-Memory Usage-Encryption	AES-Memory Usage-Decryption	Blowfish-Memory Usage-Encryption	Blowfish-Memory Usage-Decryption
9329.6	4816	7816	4252.8

160 bits ECDSA key security, thus, ECDSA is preferred over RSA algorithm. Figure 6.9 and Figure 6.10 show the digital signature generation and verification time vs. message size for RSA, DSA, and ECDSA algorithms. It is clearly shown that ECDSA and DSA take less time in signature generation as compared to the RSA and are better in terms of the complexity and cryptanalysis of the algorithms. It can also be concluded that SHA1 provides better security than MD5 [115], [116]. Since, many aspects of the ECC are available as patents (also discussed in SMSSec protocol) so the DSA as well as ECDSA are choosen for the protocol. If ECDSA algorithm is available to implement, then consider it as a digital signature algorithm otherwise, use DSA algorithm.

Table 6.7: Comparison of computational efficiency (digital signature)

Algorithms	Signature Generation (Milliseconds)	Signature Verification (Milliseconds)
RSA-MD5	35.3	9.7
RSA-SHA1	34.1	10.4
RSA-OAEP-MD5	35.4	10.7
RSA-OAEP-SHA1	34.3	9.4
DSA-SHA1	13.2	9.3

Table 6.8: Comparison of memory utilization (digital signature)

Algorithms	Signature Generation (bytes)	Signature Verification (bytes)
RSA-MD5	456309.6	26678.4
RSA-SHA1	457832.8	26814.4
RSA-OAEP-MD5	460068.0	28032.0
RSA-OAEP-SHA1	459913.6	26852.8
DSA-SHA1	16840.0	16920.0

One important point is to be noted that in particular, a different per-message secret K must be generated for each different message signed; otherwise, the private key can be recovered. If a secure random or pseudorandom number generator is used, then the chance of generating a repeated K value is negligible [117]. Table 6.7 shows the results of signature generation and signature verification time (in milliseconds) for various digital signature algorithms including RSA with MD5 and SHA1, RSA padded OAEP with MD5 and SHA1, and DSA with SHA1. It concludes that DSA algorithm with the combination of SHA1 message digest (used to maintain integrity) takes minimum time for signature generation and verification of an SMS (size of 140 bytes, 160 characters). Similarly, Table 6.8 explains the memory usage (in bytes) for all the above algorithms for signature generation and verification. Here, the DSA-SHA1 uses minimum space, *i.e.*, 16840 bytes (approx. 17 KB) for signature generation and 16920 bytes (approx. 17 KB) for signature verification.

Apart from this, the DSA is more secure in comparison to RSA in terms of cryptanalysis and the impact of various attacks. All the selected algorithms and keys are proposed to store onto the SIM card or on the handset. However, from the security point of view, it is recommended to store these algorithms onto the SIM card. A noteworthy point is that the keys used in AES and ECDSA/DSA would be generated by the SK key, which is normally implemented onto the SIM card at the time of manufacturing.

6.5.3 Simulation on Real Phone

The MIDlet application was developed on J2ME platform to simulate mobile applications. The final selected algorithms, *i.e.*, AES with CTR mode algorithm and ECDSA-SHA1 algorithm have been implemented and simulated on Nokia

6300 phone by transmitting the application archive file to the mobile phone over a USB cable. The input size and key size of AES are 128 bits each while ECDSA-SHA1 key size is 160 bits. The Nokia 6300 supports GSM 900/1800/1900 or GSM 850/1800/1900 frequency band with MIDP Java 2.0 with additional Java APIs. It uses Nokia OS as its operating system. Normally, a MIDP application consists of two files, one is the application descriptor, (a text file containing application's name, API dependencies, archive size etc.) and other is a JAR file containing the classes and resources of the respective application. Applications can be deployed on mobile phones using one of the following options [118]:

1. In real deployment, the application and its descriptor file are placed on a web server, mobile phone receives the files and installs the application through OTA.

2. For testing, transmit application JAR file to mobile phone using Bluetooth.

3. For testing, transmit application JAR file to mobile phone over a USB cable.

4. For testing, transmit application JAR file to mobile phone by copying it to a memory card, then placing the card in the phone.

One can also install the MIDlet onto the mobile phone. There are two different ways of installing it: one is the download from the web onto PC, then upload onto mobile phone, and other is to install directly OTA via WAP. The application archive is available in a subdirectory of main project directory. By default, the Java ME SDK places the projects in $< home > /JavaMESDKProjects$.
$< home > /JavaMESDKProjects/SecureSMS/dist/SecureSMS.jar$

On average, the AES with CTR algorithm took 9 milliseconds for encryption and 9.2 milliseconds for decryption of an SMS, while the ECDSA algorithm took 15 milliseconds each for signature generation and signature verification on Nokia 6300 mobile phone.

6.6 Formal Proof of SecureSMS Protocol

In order to clear statement of the analysis, the BAN-Logic symbols are used in order to formally proof the authentication process of SecureSMS protocol. The notations used in BAN-Logic are described in section 3.7.4 (chapter 3).

1. *The Formal Messages in SecureSMS Protocol:*

   Phase-1: (1) MS → AS: Na, H(Na, Ta)

   (2) AS → CA/RA: $\{\text{Na, Tb}\}_{SK\_AS-CA}$, AS $\overset{SK\_AS-CA}{\leftrightarrow}$ CA/RA

   (3) CA/RA → AS: $\{\text{Tb}\}_{SK\_AS-CA}$, AS generates DK = $\text{f}_1(\text{Ta, Tc})_{SK}$, MS $\overset{SK}{\leftrightarrow}$ AS

   (4) AS → MS: H(Ta, Tc, Expirytime)

   (5) MS → AS: $\{\text{Tc}\}_{DK}$, MS $\overset{DK}{\leftrightarrow}$ AS

   Phase-2: (1) MS → AS: Na, $\{\text{Na, T}_i\}_{DK}$, MS $\overset{DK}{\leftrightarrow}$ AS

   (2) AS → MS: $\{\text{T}_i\}_{DK}$

2. *Security Assumptions:*

   a) It is assumed that SK key is shared between the MS and the AS.

   (1) MS has the secure key SK and MS $|\equiv$ MS $\overset{SK}{\leftrightarrow}$ AS

   (2) AS has the secure key SK and AS $|\equiv$ MS $\overset{SK}{\leftrightarrow}$ AS

   b) It is assumed that DK is a delegation key, which is generated with the help of SK key and is shared between the MS and the AS.

   (1) MS has the secure key SK and MS $|\equiv$ MS $\overset{DK}{\leftrightarrow}$ AS

   (2) AS has the secure key SK and AS $|\equiv$ MS $\overset{DK}{\leftrightarrow}$ AS

   c) It is assumed that the AS trusts the CA/RA.

   (1) CA/RA $|\equiv$ AS $|\Rightarrow$ MS $\overset{DK}{\leftrightarrow}$ AS, AS $|\equiv$ AS $\overset{SK\_AS-CA}{\leftrightarrow}$ CA/RA, CA/RA $|\equiv$ AS $\overset{SK\_AS-CA}{\leftrightarrow}$ CA/RA

   (2) $\frac{CA/RA \upharpoonleft P, AS \triangleleft P}{AS |\equiv CA/RA |\equiv P}$

   (3) $\frac{AS \upharpoonleft P, CA/RA \triangleleft P}{CA/RA |\equiv AS |\equiv P}$

   d) It is assumed that communication between the CA/RA and the AS is secure.

   (1) AS $|\equiv$ AS $\overset{P}{\Leftrightarrow}$ CA/RA, P is conveyance message between the AS and the CA/RA.

   (2) CA/RA $|\equiv$ AS $\overset{P}{\Leftrightarrow}$ CA/RA, P is conveyance message between the AS and the CA/RA.

3. *Security Analysis:*

   Phase-1: (1) MS $\rightarrow$ AS: MS $|\equiv \#$(Ta) $\wedge$ AS $|\equiv \#$(Ta); AS $\triangleleft$ Na, Ta, H(Na, Ta); On receiving AS checks H(Na, Ta)

   (2) AS $\rightarrow$ CA/RA: CA/RA $\triangleleft$ {Na, Tb}$_{SK\_AS-CA}$

   (3) CA/RA validates Na, and CA/RA $\rightarrow$ AS: AS $\triangleleft$ {Tb}$_{SK\_AS-CA}$, CA/RA $|\equiv \forall$ (MS $\overset{DK}{\leftrightarrow}$ AS)

   (4) AS $\rightarrow$ MS: MS $|\equiv \#$(Ta) $\wedge$ AS $|\equiv \#$(Tc), MS $\triangleleft$ Tc, Expirytime, H(Ta, Tc, Expirytime); On receiving MS checks H(Ta, Tc, Expirytime)

   (5) MS $\rightarrow$ AS: AS $\triangleleft$ {Tc}$_{DK}$, On receiving AS checks Tc

   Phase-2: (1) MS $\rightarrow$ AS: MS $|\equiv \#$(T$_i$) $\wedge$ AS $|\equiv \#$(Expirytime), AS $\triangleleft$ {Na, T$_i$}$_{DK}$

   (2) AS checks Na and T$_i$ <= Expirytime; AS $\rightarrow$ MS: MS $|\equiv \#$(T$_i$) $\wedge$ AS $|\equiv \#$(T$_i$), MS $\triangleleft$ {T$_i$}$_{DK}$

4. *Message Meaning Rule:*

   (1) $\frac{MS|\equiv(MS\overset{DK}{\leftrightarrow}AS)\wedge(AS\overset{SK\_AS-CA}{\leftrightarrow}CA/RA),MS\triangleleft H(Na,Ta)}{MS|\equiv AS\upharpoonright H(Na,Ta)}$

   (2) $\frac{AS|\equiv f_1(Ta,Tc)_{SK}\wedge(AS\overset{DK}{\leftrightarrow}MS),AS\triangleleft H(Ta,Tc,Expirytime)}{AS|\equiv MS\upharpoonright H(Ta,Tc,Expirytime)}$

5. *Nonce/Timestamp Verification Rule:*

   (1) $\frac{MS|\equiv\#(Ta)\wedge\#(Tc),MS|\equiv AS\upharpoonright H(Na,Ta)}{MS|\equiv AS|\equiv H(Na,Ta)}$

   (2) $\frac{AS|\equiv\#(Tb),AS|\equiv CA/RA\upharpoonright\{Na,Tb\}_{SK\_AS-CA}}{CA/RA|\equiv AS|\equiv\{Na,Tb\}_{SK\_AS-CA}}$

   (3) $\frac{CA/RA|\equiv\#(Tb),CA/RA|\equiv AS\upharpoonright\{Tb\}_{SK\_AS-CA}}{AS|\equiv CA/RA|\equiv\{Tb\}_{SK\_AS-CA}}$

   (4) $\frac{AS|\equiv\#(Ta)\wedge\#(Tc),AS|\equiv MS\upharpoonright H(Ta,Tc,Expirytime)}{AS|\equiv MS|\equiv H(Ta,Tc,Expirytime)}$

6. *Jurisdiction Rule:*

   (1) $\frac{MS|\equiv CA/RA\Rightarrow\#(Na),MS\triangleleft AS\upharpoonright H(Na,Ta)}{MS|\equiv CA/RA|\equiv AS}$

   (2) $\frac{AS|\equiv CA/RA\Rightarrow\#(Tb),CA/RA\triangleleft AS\upharpoonright\{Na,Tb\}_{SK\_AS-CA}}{AS|\equiv CA/RA|\equiv MS}$

7. *Protocol Goals:*

   a) Mutual Authentication between the MS and the AS

   b) Resistance SMS disclosure between the MS and the AS

   c) Resistance SMS spoofing between the MS and the AS

   d) Resistance signer and content verification attack between the MS and the AS

   e) Resistance replay attack between the MS and the AS

   f) Resistance man-in-the-middle attack between the MS and the AS

g) Resistance source substitution attack between the MS and the AS

h) Resistance time-memory tradeoff attack between the MS and the AS

i) Resistance codebook attack between the MS and the AS

j) Resistance key separation attack between the MS and the AS

k) Resistance known key attacks between the MS and the AS

a) Mutual authentication between the MS and the AS:

MS $|\equiv$ CA/RA $|\equiv$ AS $\wedge$ AS $|\equiv$ CA/RA $|\equiv$ MS $\rightarrow$ MS $|\equiv$ AS $\wedge$ AS $|\equiv$ MS. Thus, mutual authentication holds.

b) Resistance SMS disclosure between the MS and the AS:

There is a DK key used between the AS and the MS to provide the resistance to SMS disclosure attack.

MS $|\equiv$ #(Ta), MS $|\equiv$ DK $\wedge$ #(Tc), since DK = $\mathrm{f}_1(\mathrm{Ta}, \mathrm{Tc})_{SK}$ and $\mathrm{f}_1()$ = AES

AS $|\equiv$ #(Ta), AS $|\equiv$ DK $\wedge$ #(Tb), and $\{\}_{SK\_AS-CA} = \{\}_{DK}$ = AES

c) Resistance SMS spoofing between the MS and the AS:

$\frac{MS|\equiv(MS\overset{DK}{\leftrightarrow}AS), MS\triangleleft(Msg)_{DK}}{MS|\equiv AS\upharpoonleft Msg} \wedge \frac{AS|\equiv(AS\overset{DK}{\leftrightarrow}MS), AS\triangleleft(Msg)_{DK}}{AS|\equiv MS\upharpoonleft Msg}$

Since, MS $\overset{AES}{\Leftrightarrow}$ AS $\overset{AES}{\Leftrightarrow}$ CA/RA and AES is considered as a secure algorithm, thus, no malicious user or attacker can spoof the SMS and the resistance SMS spoofing holds.

d) Resistance signer and content verification attack between the MS and the AS:

MS $\overset{ECDSA}{\Leftrightarrow}$ AS, MS $\rightarrow$ AS $| \equiv$ $\mathrm{f}_{ECDSA}\{\mathrm{f}_{AES}(\mathrm{Msg})\}$ $\wedge$ AS $\rightarrow$ MS $| \equiv$ $\mathrm{f}_{ECDSA}\{\mathrm{f}_{AES}(\mathrm{Msg})\}$

Thus, it is not possible to recover the signature and attacker would not be able to get the information from the encrypted message.

e) Resistance replay attack between the MS and the AS:

If the attacker gets #(Ta) from message (1), he/she is unable to forge message (1) because he/she does not know Na and H function. If the attacker gets #(Tb) from message (2), he/she is unable to forge message (2) and (3) because he/she does not know SK_AS-CA. If the attacker gets #(Tc) from message (4), he/she is unable to forge Message (4) and (5) because he/she does not know DK key. Since Ta, Tb, and Tc changes every time for new request, hence, the goal of resistance replay attack between the MS and the AS holds.

f) Resistance man-in-the-middle attack between the MS and the AS:
MS $\overset{DK}{\leftrightarrow}$ AS, AS $\overset{SK\_AS-CA}{\leftrightarrow}$ CA/RA, and MS $\overset{AES}{\Leftrightarrow}$ AS $\overset{AES}{\Leftrightarrow}$ CA/RA; Since, the attacker knows neither the DK key nor the SK_AS-CA, thus, SecureSMS protocol prevents the communication from being eavesdropped.

g) Resistance source substitution attack between the MS and the AS:
Since AS $\overset{AES}{\Leftrightarrow}$ CA/RA and AS $\overset{SK\_AS-CA}{\leftrightarrow}$ CA/RA; Thus, there is no such possibility that the attacker could capture the public key of any user.

h) Resistance time-memory tradeoff attack between the MS and the AS:
The AS calculates H(Na, Ta) and compares it with the received H(Na, Ta) from the MS. Similarly, the MS calculates H(Ta, Tc, Expirytime) and compares it with the received H(Ta, Tc, Expirytime) from the AS. Thus, this attack does not affect the SecureSMS protocol security.

i) Resistance codebook attack between the MS and the AS:
MS → AS: $\{T_i, \text{Na}\}_{DK}$ and AS → MS: $\{T_i\}_{DK}$; Since, the value of T_i changes every time a request is sent from the MS to the AS, thus, resistance to codebook attack holds.

j) Resistance key separation attack between the MS and the AS:
DK = $f_1(\text{Ta, Tc})_{SK}$, MS $\overset{DK}{\leftrightarrow}$ AS; Thus, there would be no such possibility that any attacker could replace the message encrypted with same DK used for another connection. If an attacker tries the same then that message would also be discarded because of Na mismatch.

k) Resistance known key attacks between the MS and the AS:
There is no such possibility of attack because DK key is generated for a session $T_i \leq$ Expirytime (depends upon the session expiry time).

6.7 Chapter Summary

The chapter presents a SecureSMS protocol that provides end-to-end security to the SMS for the secure delivery of value added services to the mobile users. The protocol defeats reply attack, man-in-the-middle attack, SMS spoofing and disclosure, and solves the security issues related to OTA interface and SS7 signaling network. The SecureSMS protocol lowers the bandwidth utilized by 71% and 59% as compared to SMSSec and PK-SIM protocols with lesser computation and communication overheads, when number of authentication requests

(n) =10, 50, 100, and 200. The AES and ECDSA algorithms are tested on Nokia 6300 successfully, thus, can be used to provide confidentiality and non-repudiation. The authentication is provided by the SecureSMS protocol and integrity is maintained by SHA1.

Chapter 7

Batch Verification Based AKA Protocol for Secure Delivery of VAS using SMS

7.1 Introduction

A mobile user requests for value added services through a value added service provider (VASP). However, a mobile user is typically bound with limited computational capabilities. Hence, the required computations made by any protocol should be kept as minimum as possible. In the future, the constraints and limitations related to VASPs performance may occur, which may cause obstruction during multiple authentication requests simultaneously. To address this issue, we focus on minimizing the computational effort required by the VASP. Furthermore, the size of sent messages should be minimized in order to reduce the communication bandwidth between the user and the VASP. In various applications like railway reservation, flight tickets etc., different users send their requests simultaneously for the value added services and the authentication server has to handle multiple requests at the same time. To provide the VAS in such a scenario, there is a strong requirement of an efficient batch verification-based AKA protocol to deliver secure value added services to the mobile users simultaneously.

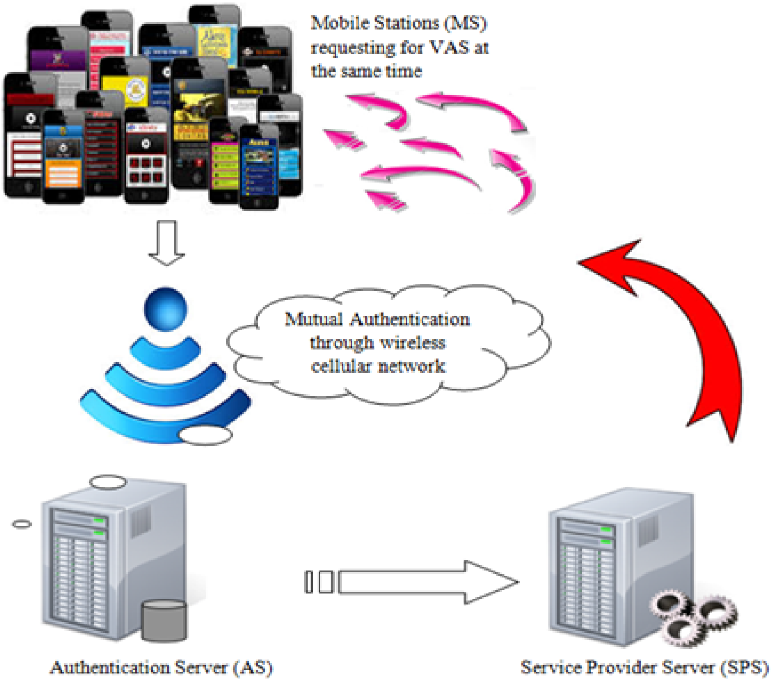

Mobile Stations (MS) requesting for VAS at the same time
Mutual Authentication through wireless cellular network
Authentication Server (AS)
Service Provider Server (SPS)

Figure 7.1: Batch verification authentication requests for VAS

7.2 System and Security Model

This section presents a system model and basic preliminaries in order to develop an efficient batch oriented protocol for providing the value added services to the mobile users.

7.2.1 System Model

A scenario for value added services is introduced where multiple MS send their authentication requests to the AS at the same time or in a fixed (very less) duration. It′s a challenge for the AS to verify and authenticate maximum number of MS based on its capacity to handle authentication requests in an efficient way. This scenario is presented in Figure 7.1. The AS handles all the authentication requests and sends the information of verified and authentic MS(s) to the server responsible for the value added service. Afterward, the service provider server (SPS) provides the service to all valid MS(s). These authentication requests may be single or multiple; however, there is very rare chance of single authentication request to the AS at one time. If server handles one request at a time, then it requires a queue to manage all the incoming requests. But then, management of a queue becomes another task that increases overhead, execution time, and

Table 7.1: Notations

Symbol	Definition	Size(Bits)
IMSI	International Mobile Subscriber Identity	128
TID	Temporary Identity	128
G	Identity of service provider	128
SK	Shared secret key between MS and AS	128
DK	Delegation key generated from SK	128
MAC	Message Authentication Code	64
T	Timestamp	64
K	Random Number	128
S	Signature generated by MS	128
Actcode	Activation Code of SK key	64
SIMcode	SIM Code of SK key	64
$f_1()$	Function used to generate DK	-
$f_2()$	Function used to generate TID	-
$f_3()$	Function used to generate MAC	-
$\oplus$	Bitwise-XOR operation	-
$\|$	Concatenation	-

cost of authentication process. In fact, the AS processing and the approach used must be very efficient to handle all the requests in very less time. One way for better handling multiple authentication requests simultaneously is to perform a batch authentication process for all incoming requests at one time or within fixed time duration. But, there may be one or more invalid requests generated by an adversary. In such case, first task is to identify the invalid request(s) and remove it/them from the batch, then perform re-batch authentication process. But, we have to pay additional cost for every re-batch authentication. Various notations used in this chapter are listed in Table 7.1 along with their sizes.

7.2.2 Security Requirements

The following are the security requirements that must be fulfilled in order to develop secure protocol for the value added services.

1. *Mutual Authentication:* The proposed protocol must provide mutual authentication, *i.e.*, MS must authenticate the valid AS to which it is requesting and the AS must verify whether the MS is a part of this system and a valid user. This mutual authentication process prevents system from eavesdropping and impersonation attacks.

2. *Session Key Establishment:* Secret session key is always a concern, when

one deals with symmetric key-based system. The protocol should be able to handle key generation, its transmission over the network, and its usage.

3. *Privacy Preservation:* The original identity of each MS must be hidden during its transmission over the network. Such privacy preservation helps to gaurd the system from man-in-the-middle attack and redirection attack.

7.3 Proposed Protocol: EXTVAS-AKA

This section proposes an efficient protocol called EXTVAS-AKA for value added services, which is shown in Figure 7.2.

7.3.1 System Assumptions

Here, some assumptions for this system are made, which are as follows: (1) It is assumed that all MS have their unique identity as IMSI, (2) We consider the AS as the AuC in the traditional cellular network, (3) The SK key is stored in AuC′s database as well as onto the SIM at the time of manufacturing as nowadays performs in the traditional cellular network, (4) It is also assumed that the AS is a trusted server that does not send the messages generated with the secret key of one mobile user to other mobile users, similar to the the traditional cellular network (This assumption supports the signature generated by each MS).

7.3.2 Batch Authentication and Key Agreement

This subsection presents the EXTVAS-AKA protocol, which provides a batch oriented mutual authentication between the AS and the MS_i. This protocol maintains the integrity between each MS_i and the AS using message authentication codes, which improves the limitation of VAS-AKA protocol [119]. Let m be the total number of authentication requests generated by various mobile users MS_i at the same time or in a fixed interval (very short) and are sent to the AS. Initially, each MS chooses a random number k_i $(1 < \mathrm{k}_i < \mathrm{m})$, generates current timestamp T_i, and DK_i, where $\mathrm{DK}_i = \mathrm{f}_1(\mathrm{T}_i)_{SK_i}$. In fact, this DK_i is generated at MS as well as at AS with a shared secret key SK_i, which is stored at AS and onto the SIM card at the time of manufacturing. The Actcode is the one time activation code, which is sent to the AS and is used to retrieve SIMcode at server. It is similar, to one that has explained in section 5.3.3 (chapter 5). It is assumed that a SIMcode is generated and stored onto the SIM card and at AS when a SIM card gets activated. This SIMcode is attached as a label to the secret key SK at AS. The generation of Actcode (at MS) and SIMcode (at AS) are not publically available and are secret in nature. This can be achieved as follows: *at MS*: Actcode = LCS$_n$(T$_1$ ⊕ LAI ⊕ SIMcode) and *at AS*: SIMcode = RCS$_n$(T$_1$ ⊕ LAI ⊕ Actcode).

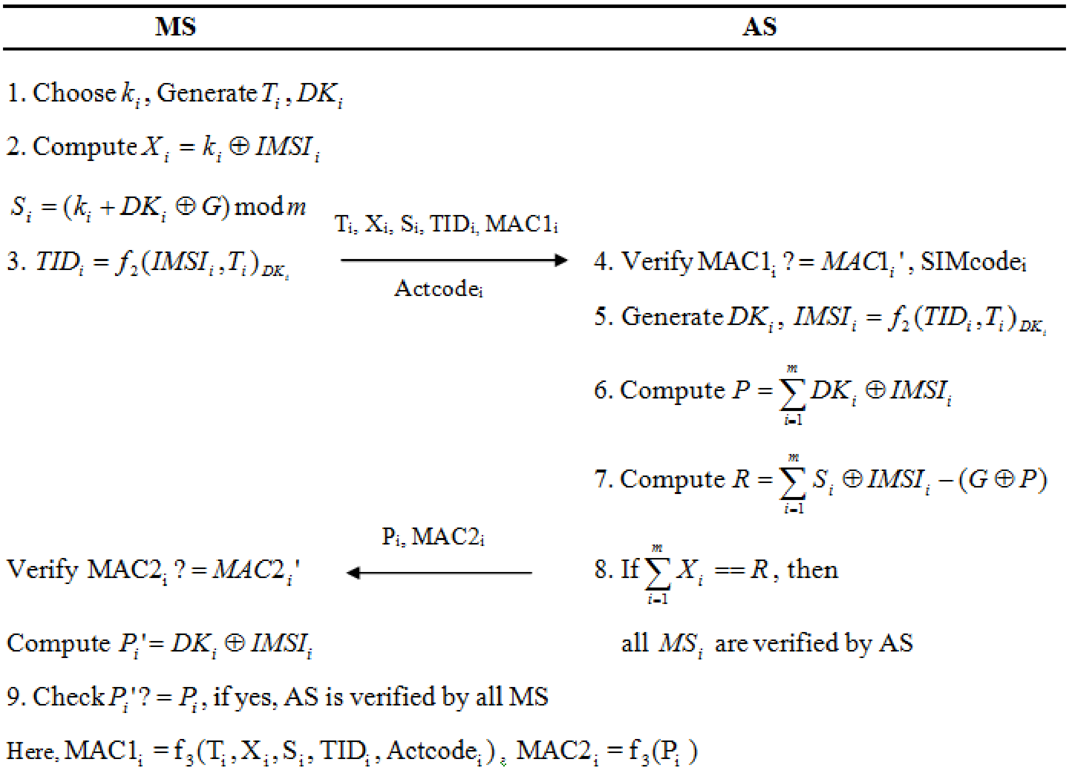

MS
AS
1. Choose k_i, Generate T_i, DK_i
2. Compute $X_i = k_i \oplus IMSI_i$
$S_i = (k_i + DK_i \oplus G) \bmod m$
3. $TID_i = f_2(IMSI_i, T_i)_{DK_i}$
T$_i$, X$_i$, S$_i$, TID$_i$, MAC1$_i$
Actcode$_i$
4. Verify MAC1$_i$?= $MAC1_i'$, SIMcode$_i$
5. Generate DK_i, $IMSI_i = f_2(TID_i, T_i)_{DK_i}$
6. Compute $P = \sum_{i=1}^{m} DK_i \oplus IMSI_i$
7. Compute $R = \sum_{i=1}^{m} S_i \oplus IMSI_i - (G \oplus P)$
P$_i$, MAC2$_i$
Verify MAC2$_i$?= $MAC2_i'$
8. If $\sum_{i=1}^{m} X_i == R$, then
all MS_i are verified by AS
Compute $P_i' = DK_i \oplus IMSI_i$
9. Check P_i' ? = P_i, if yes, AS is verified by all MS
Here, MAC1$_i$ = f$_3$(T$_i$, X$_i$, S$_i$, TID$_i$, Actcode$_i$), MAC2$_i$ = f$_3$(P$_i$)

Figure 7.2: Proposed EXTVAS-AKA protocol

Then, every MS$_i$ computes X$_i$ = k$_i$ ⊕ IMSI$_i$ and signature (symmetric signature with assumption 4) as S$_i$ = (k$_i$ + DK$_i$ ⊕ G) mod m, where ⊕ is bitwise XOR operation. Each MS$_i$ computes f$_2$(IMSI$_i$, T$_i$)$_{DK_i}$ to prevent the transmission of original IMSI$_i$ over the network, which obviates the ID theft, eavesdrop, and man-in-the-middle attacks. Now, each MS$_i$ sends authentication request as (T$_i$, X$_i$, S$_i$, TID$_i$) to the AS along with MAC1$_i$ and Actcode$_i$, where MAC1$_i$ = f$_3$(T$_i$, X$_i$, S$_i$,TID$_i$, Actcode$_i$). On receiving the authentication requests, the AS first computes MAC1$'_i$ and SIMcode$_i$, and compares MAC1$_i$?= MAC1$'_i$. If it verifies correctly, then the AS computes DK$_i$ and IMSI$_i$ = f$_2$(TID$_i$, T$_i$)$_{DK_i}$. The function f$_2$() is just like any reversible symmetric encryption function, where the plaintext and shared key generate ciphertext and then ciphertext and same key are able to produce the original plaintext. In this function, only the plain text is known however, the structure of function and DK_i are secret in nature. One can use any secure standard encryption algorithm in place of function $\mathrm{f}_2()$. Next, the AS computes P = $\sum_{i=1}^{m}$ ($\mathrm{DK}_i \oplus \mathrm{IMSI}_i$) and R = $\sum_{i=1}^{m}$ $\mathrm{S}_i \oplus \mathrm{IMSI}_i$ - (G$\oplus$P), where G is the identity of service provider. If $\sum_{i=1}^{m}\mathrm{X}_i$ == R, then all MS_i are successfully verified by the AS otherwise, one or more MS_i are malicious or invalid MS_i, which then requires re-batch authentication. In the re-batch authentication process, first, the AS finds the invalid MS_i by an algorithm namely $Invalid\_req\_algorithm(AR)$, which is discussed in the forthcoming section. The AS removes the invalid MS_i from the batch and again computes P = $\sum_{i=1}^{m-t}$ ($\mathrm{DK}_i \oplus \mathrm{IMSI}_i$) and R = $\sum_{i=1}^{m-t}$ $\mathrm{S}_i \oplus \mathrm{IMSI}_i$ - (G$\oplus$P), where t is the total number of invalid MS_i. Then, the AS compares $\sum_{i=1}^{m-t}\mathrm{X}_i$ == R. If it holds then, all MS_i are authenticated by the AS otherwise, repeat the re-batch authentication process. Finally, the AS sends all P_i to respective MS_i along with $\mathrm{MAC2}_i$, where $\mathrm{MAC2}_i$ = $\mathrm{f}_3(\mathrm{P}_i)$. All MS_i first compute $\mathrm{MAC2'}_i$ and compare it with the received $\mathrm{MAC2}_i$, *i.e.*, $\mathrm{MAC2}_i$?= $\mathrm{MAC2'}_i$. If it holds, then the MS_i compute $\mathrm{P'}_i$ and compare it with the received P_i, where $\mathrm{P'}_i = \mathrm{DK}_i \oplus \mathrm{IMSI}_i$. If both are same, then the AS is verified by all MS_i otherwise, the particular MS_i terminates the connection and resends the authentication request to the AS.

Subsequent Authentication Request: Any subsequent request by respective MS_i is treated as session authentication request within the expiry time of DK_i and is handled as follows:

All the computations are stored at MS as well as at AS until the expiry time of DK_i. Thus, if the MS_i requests for the authentication within this time, then DK_i, X_i, S_i are not changed at MS. Only a new TID_j is generated by new T_j and IMSI_i. Similarly, at AS, only IMSI_i is computed from the received TID_j and T_j. In such a scenario, DK_i remains same as previous for the respective MS_i within session time. If the AS finds the valid MS_i session active, it does not return any message to the MS_i and starts delivering the service to the corresponding MS_i.

7.4 Discussion

This subsection discusses about the reliability of the proposed protocol along with an algorithm to detect invalid MS_i requests similar to [58].

7.4.1 Reliability Analysis

In the EXTVAS-AKA protocol, if all MS$_i$ are successfully verified, then this protocol achieves its maximum reliability with respect to its performance. In such a case, this batch authentication scheme provides maximum successful authentications between the AS and the MS$_i$ at a time and generates minimum verification delay. Let N$_{MS}$ be the maximum number of authentication requests generated simultaneously. Out of these requests, some may be invalid authentication requests, assume them as N$_{IN}$. Since, N$_{MS}$ may be a very large number based on the type of value added service, the AS may not authenticate all the requests at one time due to its limited capacity. We assume that N$_{AS}$ is the maximum capacity of the AS to authenticate requests simultaneously. For the statistical analysis, it is assumed that N$_{MS}$ = 1000, N$_{AS}$ = 900, and N$_{IN}$ = 1% of N$_{MS}$, *i.e.*, 10. Let *prob*$\{t\}$ is the probability that exactly *t* invalid authentication requests are sent to the AS. Then the probability of the Hypergeometric distribution is as follows:

$$\text{prob}\{\text{t}\} = \frac{\begin{pmatrix} N_{MS} - N_{IN} \\ N_{AS} - t \end{pmatrix}\begin{pmatrix} N_{IN} \\ t \end{pmatrix}}{\begin{pmatrix} N_{MS} \\ N_{AS} \end{pmatrix}}, \text{ where t = 1, 2,...10}$$

This indicates that (N$_{AS}$ - t) valid requests are sent from the (N$_{MS}$ - N$_{IN}$). One or more invalid request(s) in a batch leads to batch verification failure and in such cases re-batch verification is required.

7.4.2 Invalid Request Detection Algorithm

This subsection presents an algorithm to detect the invalid MS$_i$ from a batch of authentication requests. The proposed algorithm is based on divide and conquer approach and is as follows:

A batch of authentication requests can be divided at most $\lceil \log_2 \text{m} \rceil$ times. At the end, this algorithm has a set of total number of invalid requests from MS$_i$ and these invalid requests must be removed from the batch for re-batch authentication. Each invalid MS$_i$ is placed in the black list of MS$_i$ and can only be removed after a predefined time. During this period the request from

Algorithm 2 Invalid_req_algorithm (AR)

```
Input: The AS receives a batch (AR) of $m$ authentication requests {R$_1$,R$_2$,R$_3$, ...,R$_m$}
Output: Returns the invalid request(s) otherwise return true

if (verify(AR)) then return True
else
    if (size(AR)==1) then return IMSI$_i$ $\epsilon$ AR as invalid request
    else
        set AR$_1$ = {R$_1$, R$_2$, R$_3$,..., R$_{\lceil m/2 \rceil}$}
        set AR$_2$ = {R$_{\lceil m/2 \rceil+1}$, R$_{\lceil m/2 \rceil+2}$, R$_{\lceil m/2 \rceil+3}$,..., R$_m$}
    end if
end if
Invalid_req_algorithm (AR$_1$)
Invalid_req_algorithm (AR$_2$)
```

particular MS$_i$ is discarded.

7.5 Security Analysis of EXTVAS-AKA Protocol

This section provides the security analysis of the EXTVAS-AKA protocol in terms of mutual authentication, session key establishment, and privacy preservation.

7.5.1 Mutual Authentication

The EXTVAS-AKA protocol provides mutual authentication between the AS and the MS$_i$. The AS authenticates the MS$_i$ by checking $\sum_{i=1}^{m}$X$_i$ == R, and each MS$_i$ authenticates the AS by comparing P$'_i$?= P$_i$.

7.5.2 Session Key Establishment

Each DK$_i$ key is used as a session key for each authentication between the AS and each MS$_i$. The DK$_i$ is generated from the SK$_i$ key as DK$_i$ = f$_i$(T$_i$)$_{SK_i}$, where T$_i$ is the current timestamp when this key is being generated. The same key is used for a session within the expiry time.

7.5.3 Privacy Preservation

The privacy of each MS_i is well protected during the authentication over the network. The TID_i is computed from the original IMSI_i as $\mathrm{TID}_i = \mathrm{f}_2(\mathrm{IMSI}_i, \mathrm{T}_i)_{DK_i}$, where function $\mathrm{f}_2()$ is reversible in nature. The function $\mathrm{f}_2()$ with DK_i prevents man-in-the-middle attack as it protects the actual identity of mobile user. Additionally, the integrity protection of transmitted messages is maintained with message authentication codes that overcome the problem of redirection attack.

7.6 Performance Evaluation of EXTVAS-AKA Protocol

This section analyzes the performance efficiency of the EXTVAS-AKA protocol in terms of transmission or communication overhead, computation overhead, protocol execution time, verification delay, and re-batch verification delay.

7.6.1 Communication Overhead

The communication overhead generated from various other protocols like IBV [59], ECDSA-AKA [60], BLA [62], and ABAKA [58] are compared along with EXTVAS-AKA. As per our knowledge, there is no protocol available in the literature that is directly related to this work. The above mentioned protocols provide authentication of value added services in vehicular ad hoc networks. However, the communication and computation overhead generated by EXTVAS-AKA protocol can be compared with the above mentioned protocols because these protocols are based on mutual authentication, and the flow of information is same in all the protocols. However, the verification delay is different for vehicular ad hoc network protocols and cellular mobile network protocol such as EXTVAS-AKA, because vehicular ad hoc network protocols have additional devices and road side equipments to communicate secure information within the network. Now, the transmission overhead during the single as well as multiple authentication requests process is discussed.

Single Authentication: When m=1 in a batch authentication process, then it is a single authentication request process, where m is the maximum number

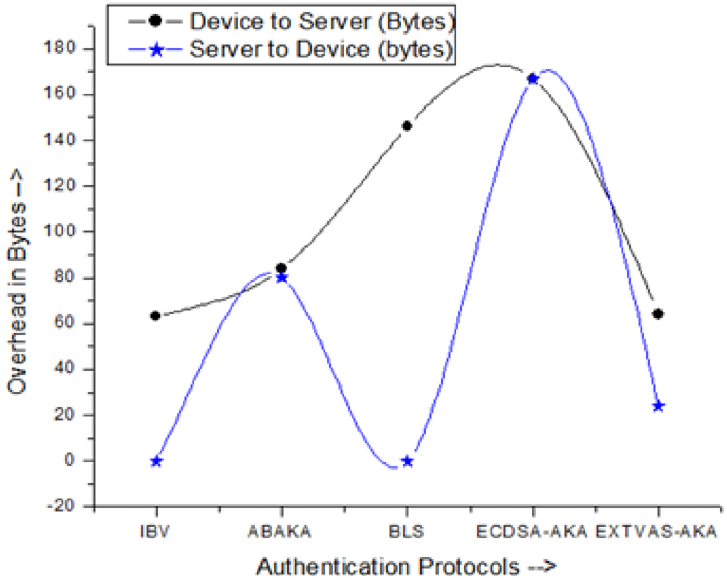


Figure 7.3: Single authentication request overhead in various protocols

Table 7.2: Communication overhead in single authentication

Protocols	Device to Server (bytes)	Server to Device (bytes)
IBV	63	N/A
ABAKA	84	80
BLS	146	N/A
ECDSA-AKA	167	167
EXTVAS-AKA	72	24

of authentication requests at one time. Table 7.2 represents the total number of bytes needs to be transmitted in different protocols from the device to the server and the server to the device. In the EXTVAS-AKA protocol, the device is a mobile station and server is an authentication server. It can be clearly observed that out of ABAKA, BLA, ECDSA-AKA,and EXTVAS-AKA, the proposed EXTVAS-AKA protocol generates lesser transmission overhead, *i.e.*, 72 and 24 bytes from the MS to the AS and from the AS to the MS respectively, however, it is little more than the IBV protocol, which produces 63 byes overhead from the device to the server. The IBV and BLS protocols do not transmit any data from the server to the device.

Multiple Authentications: Multiple authentication requests (m) can be handled by the single batch verification process simultaneously. But, it definitely increases the computations and the overheads. Table 7.3 represents the number of transmitted bytes over the network to complete the m-authentication requests

Table 7.3: Communication overhead in multiple authentication

Protocols	Device to Server (bytes)	Server to Device (bytes)
IBV	63n	N/A
ABAKA	84n	80n
BLS	146n	N/A
ECDSA-AKA	167n	167n
EXTVAS-AKA	72n	24n

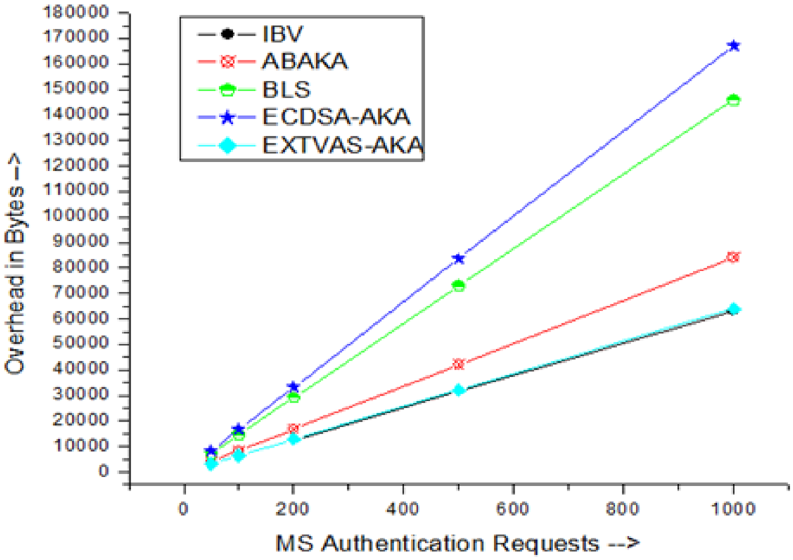


Figure 7.4: Multiple authentication requests from the device to the server

generated by different mobile users. The communication overhead generated by m-authentication requests is equal to (m*communication overhead generated by a single authentication request). Both types of authentication can be analyzed by Figure 7.3, which represents two separate graphs, one for the device to the server and other for a server to the device. One can observe that in both graphs, the EXTVAS-AKA protocol generates lesser transmission overhead. (In the server to the device graph, the IBV and the BLS protocols overheads assume to zero as they do not apply this process during the authentication).

Figure 7.4 and Figure 7.5, each shows a graph for one way multiple authentication between the MS authentication requests and the size of transmitted bytes from the device to the server and from the server to the device respectively. Both Figures clearly indicate that EXTVAS-AKA produces a lesser number of transmitted bytes (except IBV) during *m*-authentication requests, where m = 50, 100, 200, 500, and 1000. In Figure 7.5, the IBV and the BLS protocols do not send any information (byte) from the server to the device, thus have considered

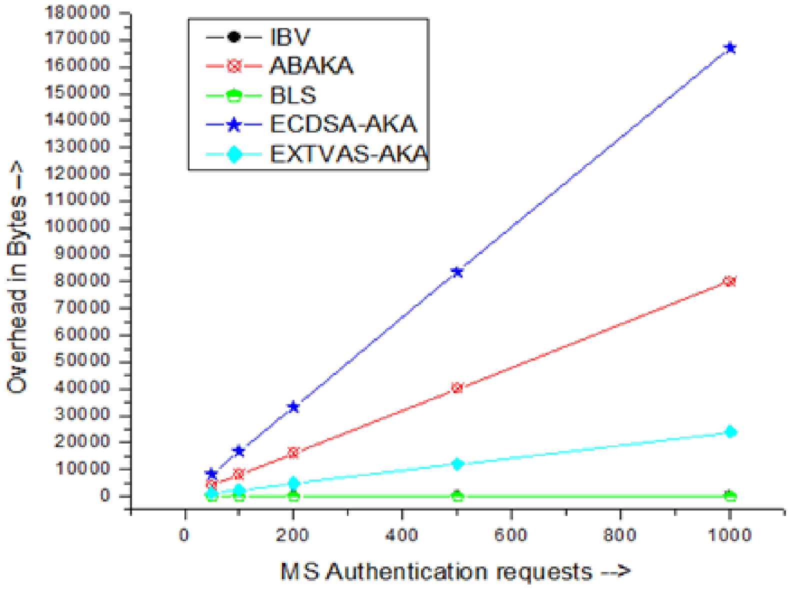


Figure 7.5: Multiple authentication requests from the server to the device

zero values. In single authentication as well as multiple authentication process, the EXTVAS-AKA protocol is compared with the other discussed protocols. In one way multiple/single authentication request(s) process from the device to the server, the EXTVAS-AKA protocol is able to reduce 14.29%, 50.69%, and 56.89% of the transmission bandwidth as compared to ABAKA, BLS, and ECDSA-AKA protocols respectively. Similarly, from the server to the device for one way multiple/single authentication request(s) process, the EXTVAS-AKA lowers the bandwidth by 70% and 85.63% in comparison to ABAKA and ECDSA-AKA protocols respectively.

7.6.2 Computation Overhead

This subsection analyzes the computation overhead generated for both type of authentication in the EXTVAS-AKA protocol. *Single Authentication (i = 1):* The following computations are performed at MS and at AS during the single authentication:

Computation at MS: (1) $f_2(IMSI_i, T_i)_{DK_i}$, (2) $f_1(T_i)_{SK_i}$, (3) $k_i \oplus IMSI_i$, (4) $Y=G \oplus DK_i$, (5) k_i+Y, (6) $DK_i \oplus IMSI_i$, (7) $MAC1_i = f_3()$, (8) $MAC2'_i = f_3()$, (9) $Actcode_i$

Computation at AS: (1) $f_1(T_i)_{SK_i}$, (2) $f_2(TID_i, T_i)_{DK_i}$, (3) $DK_i \oplus IMSI_i$, (4) $S'_i = S_i \oplus IMSI_i$, (5) $G'=G \oplus P$, (6) $S' - G'$, (7) $MAC1'_i=f_3()$, (8) $MAC2'_i=f_3()$, (9) $SIMcode_i$

Multiple/Batch Authentication (i = 2, 3,...,m): The computation performed during the multiple authentication is below as:

Computation at MS: (1) m[f_2(IMSI$_i$, T$_i$)$_{DK_i}$], (2) m[f_1(T$_i$)$_{SK_i}$], (3) [k$_i$ $\oplus$ IMSI$_i$], (4) [Y=G $\oplus$ DK$_i$], (5) m[k$_i$+Y], (6) [DK$_i$ $\oplus$ IMSI$_i$], (7) m[MAC1$_i$ = f_3()], (8) m[MAC2$'_i$ = f_3()], (9) m[Actcode$_i$]

Computation at AS: (1) m[f_1(T$_i$)$_{SK_i}$], (2) m[f_2(TID$_i$, T$_i$)$_{DK_i}$], (3) m[DK$_i$ $\oplus$ IMSI$_i$], (4) (m-1)[P=$\sum_{i=1}^{m}$ P$_i$], (5) m[S$'_i$ = S$_i$ $\oplus$ IMSI$_i$], (6) G$'$=G $\oplus$ P, (7) (m-1)[S$'$=$\sum_{i=1}^{m}$ S$_i$], (8) S$'$ - G$'$, (9) (m-1)[$\sum_{i=1}^{m}$ X$_i$], (10) m[MAC1$_i$=f_3()], (11) m[MAC2$'_i$=f_3()], (12) m[SIMcode$_i$]

7.7 Simulation of EXTVAS-AKA Protocol

The simulation of EXTVAS-AKA protocol is performed in Java environment. This section calculates total execution time, batch verification time, and re-batch verification time of the proposed protocol.

7.7.1 Protocol Execution Time

It is assumed that the time taken by the MS or the AS to perform an addition of two bitwise numbers is T$_{add}$, the time to perform a multiplication, *i.e.*, bitwise-XOR is T$_{mul}$, and the time to execute a subtraction is T$_{sub}$. The message authentication codes (MAC) are generated using f_3(). The total number of operations (addition, multiplication, and subtraction) required in a batch authentication process is as: m* f_2(IMSI$_i$, T$_i$)$_{DK_i}$, m* f_2(TID$_i$, T$_i$)$_{DK_i}$, m* f_1(T$_i$)$_{SK_i}$, T$_{sub}$, (5m+1)* T$_{mul}$, (4m-3)* T$_{add}$, 2* f_3(), 2* f_3(), m* Actcode$_i$, and m* SIMcode$_i$

This implementation considers a client-server paradigm where MS is a client and AS is a server. On an average, the execution time to perform each operation, *i.e.*, addition, multiplication (bitwise-XOR), and subtraction, is T$_{add}$ = .000933 milliseconds, T$_{mul}$ = .030322 milliseconds, and T$_{sub}$ = .000933 milliseconds respectively. On an average, the server connection establishment time is 3383 milliseconds, transmission time for message (T$_i$, X$_i$, S$_i$, TID$_i$, MAC1$_i$, Actcode$_i$) is 8.35 milliseconds, and transmission time for message (P$_i$, MAC2$_i$) is 9.39 milliseconds. Since, in results, different time estimation values are received from different MS and the AS for sending the messages, thus, the average value of 30 iterations are considered for each result. This work has been simulated with 50 MS clients sending their authentication requests to a single server AS. Here, Enc = Encryption, Dec = Decryption

Table 7.4: Computation for function $f_2()$ and Actcode/SIMcode

$f_2()$				Actcode/SIMcode function		
Enc ExT	TUM	Dec ExT	TUM	ExT	PCPUT	TUM
42	9139681	15	9124165	0.89	93.60	12968290

Table 7.5: Computation for $f_3()$ and $f_1()$ functions

$f_3()$			$f_1()$		
ExT	PCPUT	TUM	ExT	PCPUT	TUM
221.60	296.40	15211840	273.41	296.40	15204024

The $f_2()$ is implemented as AES-128 algorithm with 128 bits key. It takes $IMSI_i/TID_i$ and T_i as input, similar to AES-CTR. Table 7.4 represents the results obtained for Actcode/SIMcode and AES algorithm the execution time of Actcode/SIMcode took 0.89 milliseconds, while the encryption (generation of TID_i) and decryption (generation of $IMSI_i$) of AES took 42 and15 milliseconds respectively. The $f_1()$ and $f_3()$ are implemented as HMACSHA256 and HMACSHA1 respectively. The observed results for the same is shown in Table 7.5, where the output of HMACSHA1 and HMACSHA256 are truncated to 64 and 128 bits because the output of $f_3()$ is 64 bits that is a MAC, while $f_1()$ generates the output of 128 bits that is DK_i key. The input to HMACSHA1 and HMACSHA256 are 512 bits (actual input size plus trailing zeros to make it multiple of 512).

Total Operations Required by EXTVAS-AKA in Single Authentication:

At MS: $f_1(T_i)_{SK_i}$, $f_2(IMSI_i, T_i)_{DK_i}$, 3* T_{mul}, T_{add}, $MAC1_i = f_3()$, $MAC2'_i = f_3()$, $Actcode_i$

At AS: $f_1(T_i)_{SK_i}$, $f_2(IMSI_i, T_i)_{DK_i}$, 3* T_{mul}, T_{sub}, $MAC1_i = f_3()$, $MAC2'_i = f_3()$, $SIMcode_i$

Total Execution Time for EXTVAS-AKA in Single Authentication:

Total time = server connection establishment time + transmission time for all messages + time at MS + time at AS = 3383 + (8.35 + 9.39) + (221.60 + 42 + 3*0.030322 + 0.000933 + 273.41 + 273.41 + 0.89) + (221.60 + 15 + 3*0.030322 + 0.000933 + 273.41 + 273.41 + 0.89) = 4996.54 milliseconds = 4.99 seconds

(= 5 seconds)

Total Operations Required by EXTVAS-AKA in Multiple/Batch Authentication:

At MS: m*$f_1(T_i)_{SK_i}$, m*$f_2(IMSI_i, T_i)_{DK_i}$, 3m*T_{mul}, m*T_{add}, m*$MAC1_i = f_3()$, m*$MAC2'_i = f_3()$, m*$Actcode_i$

At AS: m*$f_1(T_i)_{SK_i}$, m*$f_2(IMSI_i, T_i)_{DK_i}$, (2m+1)*T_{mul}, T_{sub}, (3m-3)*T_{add}, m*$MAC1_i = f_3()$, m*$MAC2'_i = f_3()$, m*$SIMcode_i$

Total Execution Time for EXTVAS-AKA in Multiple/Batch Authentication:

Total time = server connection establishment time + transmission time for all messages + time at MS + time at AS = 3383*m + (8.35 + 9.39)*m + (m*221.60 + m*42 + 3*m*0.030322 + m*0.000933 + m*273.41 + m*273.41 + m*0.89) + (m*221.60 + m*15 + (2m+1)*0.030322 + 0.000933 + (3m-3)*0.000933 + m*273.41 + m*273.41 + m*0.89) = 0.028456 + m*4996.51 milliseconds = 0.000028 + m*4.99 seconds (= 0.000028 + m*5 seconds)

7.7.2 Verification Delay

This subsection discusses the verification delay during the single as well as multiple authentication. The verification delay is the time estimation between the sent message and received response/completion of protocol. Here, verification delay time for the MS is the time between the sent message (T_i, X_i, S_i, TID_i, $MAC1_i$, $Actcode_i$) and received response message (P_i, $MAC2_i$) while the same time can be counted for the AS between the sent message (P_i, $MAC2_i$) and protocol completion.

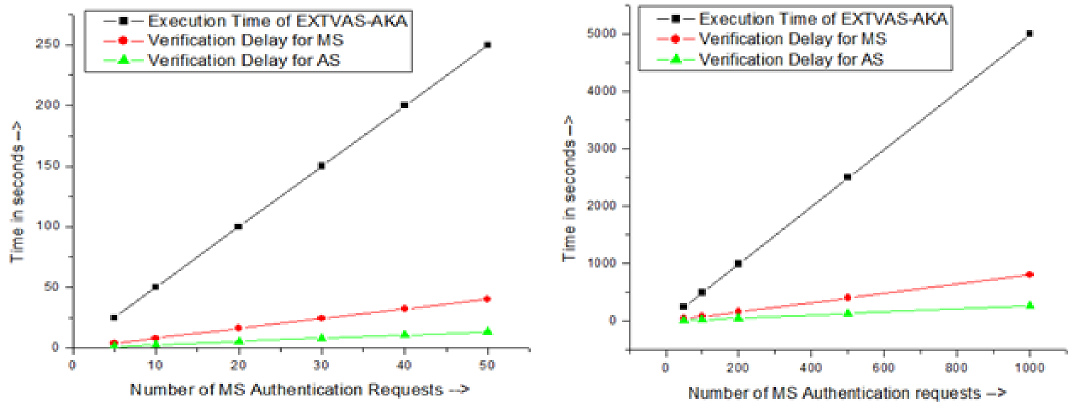


Figure 7.6: Execution time and verification delay for (a) m = 5, 10, 20, 30, 40, 50 (b) m = 50, 100, 200, 500, 1000

Operations Performed in Single Authentication:

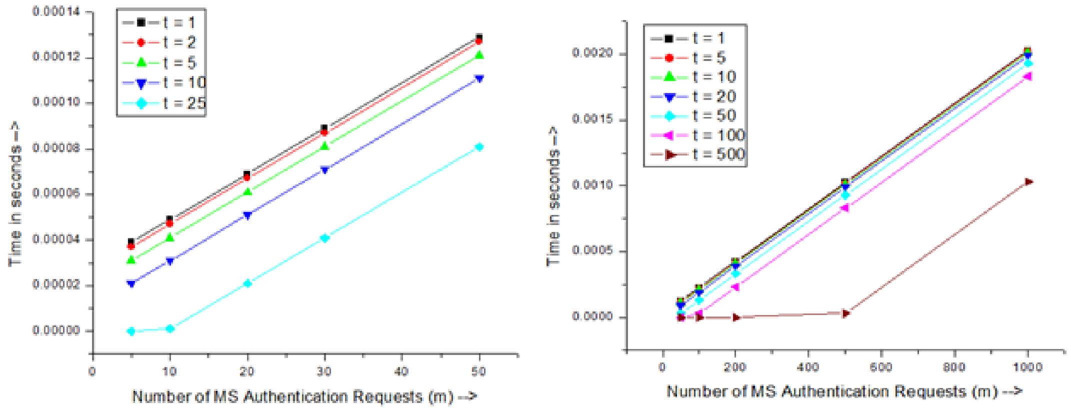


Figure 7.7: Re-batch verification delay for (a) m = 5, 10, 20, 30, 50; t =1, 2, 5, 10, 25 (b) m = 50, 100, 200, 500, 1000; t = 1, 5, 10, 20, 50, 100, 500

For MS: $f_1(T_i)_{SK_i}$, $f_2(IMSI_i, T_i)_{DK_i}$, 3* T_{mul}, T_{sub}, $MAC1_i = f_3()$, $MAC2'_i = f_3()$, $SIMcode_i$

Verification delay time = 221.60 + 42 + 273.41 + 273.41 + 3*0.030322 + 0.000933 + 0.89 = 811.40 milliseconds = 0.81 seconds

For AS: $MAC2'_i = f_3()$, T_{mul}

Verification delay time = 273.41 + 0.030322 = 273.44 milliseconds = 0.27 seconds

Operations Performed in Multiple/Batch Authentication:

For MS: m*$f_1(T_i)_{SK_i}$, m*$f_2(IMSI_i, T_i)_{DK_i}$, (2m+1)* T_{mul}, T_{sub}, (3m-3)*T_{add}, m*$MAC1_i = f_3()$, m*$MAC2'_i = f_3()$, m*$SIMcode_i$

Verification delay time = m*221.60 + m*42 + m*273.41 + m*273.41 + (2m+1)*0.030322 + 0.000933 + (3m-3)*0.000933 + 0.89*m = 0.28456 + m*811.37 milliseconds = 0.000284 + m*0.81 seconds

For AS: m*$MAC2'_i = f_3()$, m*T_{mul}

Verification delay time = m*273.41 + m*0.030322 = 273.44*m milliseconds = 0.27*m seconds

7.7.3 Re-batch Verification Delay

If the batch authentication process is not successful then it requires re-batch authentication without including the detected invalid MS_i. After detected the invalid MS_i, remove them from the batch and execute re-batch authentication. Total operations required in re-batch authentication = (m-t)[$P=\sum_{i=1}^{m-t} P_i$], (m-t)[$S'=\sum_{i=1}^{m-t} S_i$], $G'=G \oplus P$, (m-t)[$\sum_{i=1}^{m-t} X_i$], and $S' - G' = 3(m-t)*T_{add}+T_{mul}+$

T_{sub}, where t is the number of invalid MS_i that have been removed in re-batch authentication.
Total verification delay time = 0.031255 + 0.002799*(m-t) milliseconds = 0.000031 + 0.000002*(m-t) seconds

Figure 7.6 shows the graphs for EXTVAS-AKA's execution time, verification delay time forthe MS, and verification delay time for the AS, when (a) MS authentication requests m = 5, 10, 20, 30, 40, 50 (b) m = 50, 100, 200, 500, 1000. Similarly, Figure 7.7 represents re-batch verification delay time, when (a) m = 5, 10, 20, 30, 50; t =1, 2, 5, 10, 25 and (b) m = 50, 100, 200, 500, 1000; t = 1, 5, 10, 20, 50, 100, 500. Please note that the protocol execution time is the complete time for the mutual authentication between all MS_i and the AS, depends upon the number of authentication requests.

7.8 Formal Proof of EXTVAS-AKA Protocol

Security Proof using BAN-Logic:

We use the BAN-Logic symbols in order to formally proof the authentication process of EXTVAS-AKA protocol. Various notations used in BAN-Logic are described in section 3.7.4.

1. *The Formal Messages in EXTVAS-AKA Protocol:*
   (1) $MS_i \rightarrow AS$: T_i, TID_i, S_i, X_i, $MAC1_i$, $Actcode_i$; $DK_i = f_1(Ta)_{SK_i}$, $TID_i = f_2(IMSI_i, Ta)_{DK_i}$, $MS_i \overset{SK_i}{\leftrightarrow} AS$
   (2) $AS \rightarrow MS_i$: P_i, $MAC2_i$

2. *Security Assumptions:*
   It is assumed that SK_i is shared between the MS_i and the AS.
   (1) MS_i has the secure key SK_i and $MS_i \mid\equiv MS_i \overset{SK_i}{\leftrightarrow} AS$
   (2) AS has the secure key SK_i and $AS \mid\equiv MS_i \overset{SK_i}{\leftrightarrow} AS$

3. *Security Analysis:*
   (1) $MS_i \rightarrow AS$: T_i, $f_2(IMSI_i, T_i)_{DK_i}$, $(k_i \oplus IMSI_i)$, $((k_i + DK_i \oplus G) \bmod m)$, $MAC1_i$, $Actcode_i$;
   (2) $AS \rightarrow MS_i$: On receiving message (1), the AS first computes $MAC1'_i$ and $SIMcode_i$, and compares $MAC1_i$ with $MAC1'_i$. If it holds, then the AS computes $DK_i = f_1(T_i)_{SK_i}$ and $IMSI_i = f_2(TID_i, T_i)_{DK_i}$. Next, it

calculates P and R, and then compares $\sum_{i=1}^{m}(\mathrm{X}_i)$==R. If it is true, then all MS$_i$ are verified by the AS.

Next, the AS transmits message (2) to the MS$_i$; AS → MS$_i$: (DK$_i\oplus$ IMSI$_i$), MAC2$_i$. When receiving, each MS$_i$ computes MAC2$'_i$ and compares it with the received MAC2$_i$. If both are equal, then each MS$_i$ computes DK$_i\oplus$ IMSI$_i$ and compares it with received values. If it is true, then the AS is verified by all MS$_i$.

4. *Message Meaning Rule:*

(1) $\frac{MS_i|\equiv(MS_i\overset{SK_i,DK_i}{\leftrightarrow}AS),MS_i\triangleleft f_2(IMSI_i,T_i)_{DK_i}}{MS_i|\equiv AS\upharpoonleft f_2(IMSI_i,T_i)_{DK_i}}$

(2) $\frac{AS|\equiv(AS\overset{SK_i,DK_i}{\leftrightarrow}MS_i),AS\triangleleft f_2(TID_i,T_i)_{DK_i}}{AS|\equiv MS_i\upharpoonleft f_2(TID_i,T_i)_{DK_i}}$

5. *Nonce/Timestamp Verification Rule:*

(1) $\frac{MS_i|\equiv\#(T_i),MS_i|\equiv AS\upharpoonleft f_2(IMSI_i,T_i)_{DK_i}}{MS_i|\equiv AS|\equiv f_2(IMSI_i,T_i)_{DK_i}}$

(2) $\frac{AS|\equiv\#(T_j),AS|\equiv MS_i\upharpoonleft f_2(TID_i,T_i)_{DK_i}}{AS|\equiv MS_i|\equiv f_2(TID_i,T_i)_{DK_i}}$

6. *Jurisdiction Rule:*

(1) $\frac{MS_i|\equiv AS\Rightarrow f_2(IMSI_i,T_i)_{DK_i},MS_i\triangleleft MS_i\upharpoonleft f_2(IMSI_i,T_i)_{DK_i}}{MS_i|\equiv AS|\equiv MS_i}$

(2) $\frac{MS_i|\equiv AS\Rightarrow f_2(TID_i,T_i)_{DK_i},AS\triangleleft AS\upharpoonleft f_2(TID_i,T_i)_{DK_i}}{MS_i|\equiv AS|\equiv MS_i}$

7. *Protocol Goals:*

a) Mutual authentication between the MS$_i$ and the AS:

MS$_i$ $|\equiv$ AS $\wedge$ AS $|\equiv$ MS$_i$. Thus, mutual authentication holds.

b) Session key agreement:

The DK$_i$ keys between each MS$_i$ and the AS provide session key agreement. MS$_i$ $|\equiv$ DK$_i$ $\wedge$ #(T$_i$), since DK$_i$ = f$_1$(T$_i$)$_{SK_i}$

c) Freshness of messages:

AS $|\equiv$ #(T$_j$) $\wedge$ MS$_i$ #(T$_i$); Thus, key freshness between each MS$_i$ and the AS holds.

d) Privacy between each MS$_i$ and the AS:

$$\frac{MS_i|\equiv(MS_i\overset{DK_i}{\leftrightarrow}AS),MS_i\triangleleft(Msg)MAC1_i}{MS_i|\equiv AS\upharpoonleft Msg}\wedge\frac{MS_i|\equiv(MS_i\overset{DK_i}{\leftrightarrow}AS),MS_i\triangleleft f_2(IMSI_i,T_i)_{DK_i}}{MS_i|\equiv AS\upharpoonleft IMSI_i}$$

$$\frac{AS|\equiv(AS\overset{DK}{\leftrightarrow}MS_i),AS\triangleleft(Msg)MAC2_i}{AS|\equiv MS_i\upharpoonleft Msg}\wedge\frac{AS|\equiv(AS\overset{DK}{\leftrightarrow}MS_i),AS\triangleleft f_2(TID_i,T_i)_{DK_i}}{AS|\equiv MS_i\upharpoonleft TID_i}$$

Security Proof using Proverif:

(* Reachability Property *)

```
(* Public channel between the MS and the AS *)
free pubChannel : channel .
(* types *)
type key .
type ident .
type nonce .
type msgHdr .
type resp .
type sessKey .
type mac .
(* constant message headers *)
const MSG1 : msgHdr .
const MSG2: msgHdr .
const MSG: msgHdr .
const CMC: msgHdr .
(* Functions *)
fun f1 ( nonce , key ) : sessKey .
fun f2 ( ident , nonce, sessKey ) : ident .
fun f3 (nonce , bitstring, bitstring, ident, bitstring ) : mac .
fun f4 (bitstring) : mac .
fun f5 (nonce , bitstring) : bitstring .
fun f6 (nonce , bitstring) : bitstring .
fun f7 (bitstring , ident): bitstring .
fun f8 (bitstring, sessKey, bitstring): bitstring .
fun f9 (sessKey , ident): bitstring .
fun sencrypt ( bitstring , sessKey ) : bitstring .
reduc forall m: bitstring , k : sessKey ;
sdecrypt ( sencrypt (m, k ) , k ) = m.
(* The key table consists of pairs ( ident , key ) shared between the MS and the
HN. Table is not accessible by the attacker *)
table keys ( ident , key ) .
table keys2 ( ident , sessKey ) .
free s : bitstring [ private ] .
(* Secrecy Property *)
```

```
query attacker ( s ) .
(* The standard secrecy queries of ProVerif only *)
(* deal with the secrecy of private free names *)
(* DK is secret if and only if all DK are secret *)
free DK : sessKey [ private ] .
query attacker ( DK ) .
not attacker (new kims ) .
(* Authentication queries *)
event begAS( ident , sessKey ) .
event endAS( ident , sessKey ) .
event begMS( ident , sessKey ) .
event endMS( ident , sessKey ) .
query x1 : ident , x2 : sessKey ;
event (endAS( x1 , x2 ) ) ==> event (begAS( x1 , x2 ) ) .
query x1 : ident , x2 : sessKey ;
event (endMS( x1 , x2 ) ) ==> event (begMS( x1 , x2 ) ) .
event enableEnc .
(* When the attacker knows s, the event enableEnc has been executed. *)
query attacker ( s ) ==> event ( enableEnc ) .
let processMS =
(* The ident and pre-shared key of the MS *)
new imsims : ident ;
new kims : key ;
new tims : nonce ;
new SIMcodei : bitstring ;
new ki : bitstring ;
new G : bitstring;
new m : bitstring ;
insert keys ( imsims , kims ) ;
let DKms : sessKey = f1 (tims , kims) in
insert keys2 ( imsims , DKms ) ;
let actcodei : bitstring = f5 (tims , SIMcodei) in
let Xi : bitstring = f7 (ki , imsims) in (* ki XOR imsims ; *)
let Si : bitstring = f8 (ki, DKms, G) in (* (ki + DKms XOR G) mod m ; *)
let Pi : bitstring = f9 (DKms, imsims) in (* (DKms XOR imsims) ; *)
let tidms :ident = f2 (imsims , tims, DKms) in
let mac1ims : mac = f3 (tims, Xi , Si , tidms , actcodei) in
let mac2ims : mac = f4 (Pi) in
(*MS is authenticating itself to AS*)
out ( pubChannel , ( MSG1 , tims , Xi, Si, tidms , mac1ims , actcodei)) ;
(*Compute response and encryption key *)
event begAS( imsims , DKms ) ;
(* Input challenge message from AS *)
in ( pubChannel , (=MSG2, Pias : bitstring , mac2ias : mac ) ) ;
if mac2ims = mac2ias then
event endAS (imsims , DKms );
(*Receive GSM cipher mode command *)
in ( pubChannel , (=CMC, enableEncms : bool ) ) ;
(* event endMS( imsims , DKms ) ; *)
event endMS( imsims , DKms ) ;
(*Receive message from AS *)
in ( pubChannel , (=MSG, msg : bitstring ) ) ;
(* out ( pubChannel , sencrypt ( DK , DKms ) ) ; *)
out ( pubChannel , sencrypt ( msg , DKms ) ) ;
if enableEncms = true then
let msgcontent : bitstring = sdecrypt (msg, DKms ) in 0.
let processAS =
new kias : key ;
new DKas : sessKey ;
new ki2 : bitstring ;
new Gas : bitstring;
new mas : bitstring ;
in ( pubChannel,(=MSG1, tims: nonce, Xi: bitstring, Si: bitstring, tidas: ident,
mac1ims: mac, actcodei: bitstring) );
let imsias: ident = f2 (tidas , tims, DKas) in
insert keys ( imsias , kias ) ;
let DKas = f1 (tims , kias) in
insert keys2 ( imsias , DKas ) ;
```

```
let SIMcodei : bitstring = f6 (tims , actcodei) in
let Xias : bitstring = f7 (ki2 , imsias) in
let Sias : bitstring = f8 (ki2, DKas, Gas) in
let Pias : bitstring = f9 (DKas, imsias) in
let mac1ias : mac = f3 (tims, Xias , Sias , tidas , actcodei) in
let mac2ias : mac = f4 (Pias) in
new msg : bitstring ;
if mac1ims = mac1ias then
(* At this point, AS authenticated MS*)
event endAS( imsias , DKas ) ;
(*Send authentication challenge to MS *)
out ( pubChannel , (MSG2, Pias, mac2ias ) ) ;
(*AS decide whether to encrypt messages; based on rec. capa.of MS*)
new enableEncas : bool ;
event begMS( imsias , DKas ) ;
(*Send out cipher mode command *)
out ( pubChannel , (CMC, enableEncas ) ) ;
out ( pubChannel , sencrypt ( msg , DKas ) ) ;
if enableEncas = false then
event enableEnc ;
out ( pubChannel , (MSG, s ) )
else
out ( pubChannel , (MSG, sencrypt ( s , DKas ) ) ) .
process
( ( ! processMS ) | processAS )
```

Output of Proverif:

```
Nitesh@Nitesh-PC  /proverif1.88 $ ./proverif examples/gsm/batchVAS2.pv
Process:
– Query attacker(s[]) ==> event(enableEnc)
Completing...
ok, secrecy assumption verified: fact unreachable attacker(kims[!1 = v_945])
Starting query attacker(s[]) ==> event(enableEnc)
RESULT attacker(s[]) ==> event(enableEnc) is true.
– Query event(endMS(x1,x2)) ==> event(begMS(x1,x2))
```

```
Completing...
ok, secrecy assumption verified: fact unreachable attacker(kims[!1 = v_2079])
Starting query event(endMS(x1,x2)) ==> event(begMS(x1,x2))
RESULT event(endMS(x1,x2)) ==> event(begMS(x1,x2)) is true.
– Query event(endAS(x1_2400,x2_2401)) ==> event(begAS(x1_2400,x2_2401))
Completing...
ok, secrecy assumption verified: fact unreachable attacker(kims[!1 = v_3256])
Starting query event(endAS(x1_2400,x2_2401)) ==> event(begAS(x1_2400,x2_2401))
RESULT event(endAS(x1_2400,x2_2401)) ==> event(begAS(x1_2400,x2_2401)) is true.
– Query not attacker(DK[])
Completing...
ok, secrecy assumption verified: fact unreachable attacker(kims[!1 = v_4331])
Starting query not attacker(DK[])
RESULT not attacker(DK[]) is true.
– Query not attacker(s[])
Completing...
ok, secrecy assumption verified: fact unreachable attacker(kims[!1 = v_5377])
Starting query not attacker(s[])
RESULT not attacker(s[]) is true.
```

7.9 Chapter Summary

This chapter presents an efficient EXTVAS-AKA protocol that provides mutual authentication between each MS and the AS. The protocol verifies multiple authentication requests at one time or in a fixed time duration (very less) and also hides the original IMSI during the authentication process. The probability model to predict the number of invalid users is presented. A discussion of invalid request detection algorithm is carried out to find malicious authentication requests in a batch. The work reports that in the authentication process from the device to the server, the EXTVAS-AKA protocol saves 14.29%, 50.69%, and 56.89% of the total transmission bandwidth in comparison to ABAKA, BLS, and ECDSA-AKA protocols respectively, when number of authentication requests (n) =50, 100, 200, 500, and 1000. Similarly, from the server to the device authentication process, the EXTVAS-AKA reduces the communication bandwidth by 70% and 85.63% as compared to ABAKA and ECDSA-AKA protocols respectively.

Chapter 8

AKA Protocol for End-to-End SMS Security for Mobile Users

8.1 Introduction

The traditional SMS service offered by various mobile operators does not provide information security to the message being sent over the network. In order to protect confidential information such as password, account number etc., it should provide end-to-end secure communication between end users. SMS usage is threatened with security concerns such as SMS disclosure, man-in-the-middle attack, replay attack, and impersonation attack. SMS are transmitted as plaintext between the MS and the SMSC in the cellular network where network operators can also read the SMS content stored at SMSC.

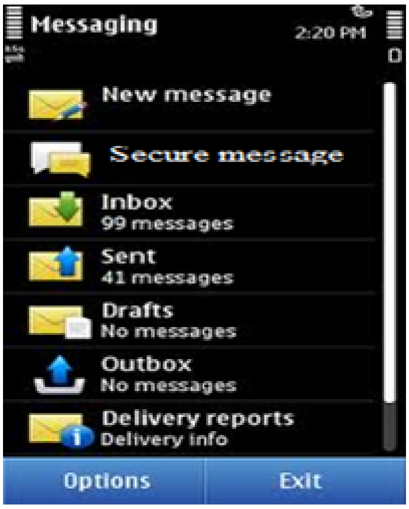


Figure 8.1: Secure message: option added in the mobile software

Table 8.1: Symbols and abbreviations

Symbol	Definitions	Bits
MS	Mobile Station referring Mobile User	—
AS	Authentication Server	—
CA/RA	Certification Authority/Registration Authority	—
IDMS	International Mobile Subscriber Identity of MS	128
ReqNo	Request Number	8
Q/Q$_n$	New Session Identifier	28
Rc/Nc/Ns/Na	Random Number	128
P$_f$	Private Port Number	16
M	Message	Variable
SQ/Seq	Sequence Number	28
PK/PK_PK-SIM	Public key of server	128
SK/SK_MS	Symmetric key b/w the MS and the AS	128
SK_AS-CA	Symmetric key b/w the AS and the CA/RA	128
SK_AS$_1$-AS$_2$	Symmetric key b/w the AS$_1$ and the AS$_2$	128
UAKey	Primary key	128
ExpT	Expiry time of DK$_1$ key	64
C_ME	Certificate of ME	40
CertSAG	Certificate of Security Access Gateway (SAG)	40
MAC/H	Message Authentication Code/Hash function	64
T$_i$	Timestamp	64
DK$_1$	Delegation key	256

8.2 System, Communication and Attack Models

This section, we presents the system and communication model along with an attack model.

8.2.1 System and Communication Model

In order to provide encryption to the data, various cipher algorithms are proposed. Here, it is recommend that the cipher algorithms should be stored onto the SIM (part of MS) as well as at AS. Since, providing security needs to do some extra effort, which is measured in terms of cost, thus, adding extra security means increasing more cost. We propose to include one more service as “Secure Message” in the menu of mobile software developed by various mobile companies as shown in Figure 8.1. Mobile operators can add some extra charges to send secure message by their customers over the network. Whenever a user wishes to send a secure message to other user, the proposed protocol named EasySMS

Table 8.2: Cryptographic functions definition

Function	Definition
f_1	Function to generate MAC
f_2	Key generation function for DK_1
$\{\}_{SK\_AS-CA}$	Encryption with Symmetric key b/w AS and CA
$\{\}_{SK\_AS_1-AS_2}$	Encryption with Symmetric key b/w AS_1 and AS_2
$\{\}_{SK\_MS_2}$	Encryption with Symmetric key MS_2 shared b/w MS_2-CA
$\{\}_{DK_1}$	Encryption with Delegation key DK_1
$\|\|$	Concatenation

is executed, which makes available the symmetric shared key at both MS, and then ciphering of message takes place using a symmetric key algorithm.

8.2.2 Attack Model

An attack model describes different scenarios for the possibilities of various attacks, where a malicious MS can access the authentic information, or misguide the legitimate MS. In this attack model, different threats and attacks like access the content of SMS during its transmission at SMSC, insecure OTA interface, cryptanalysis attack, man-in-the-middle attack, replay attack, and impersonation attack are considered. Table 8.1 represents the list of various symbols used with their size, while Table 8.2 defines various functions used in this chapter.

8.3 Proposed Protocol: EasySMS

This section proposes a new EasySMS protocol with two different scenarios, which provides end-to-end secure transmission of information in the cellular network. First scenario is illustrated in Figure 8.2 where both MS belong to same AS, in other words share the same HLR. The second scenario is presented in Figure 8.3, where both MS belong to different AS, *i.e.*, both are in different HLR. There are two main entities in the EasySMS protocol. First is the AS, which works as authentication center AuC and stores all the symmetric keys shared between the AS and the respective MS. Second entity is the certified authority/registration authority CA/RA, which stores all the information related to the mobile subscribers. It is proposed that every subscriber has to register his/her mobile number with CA/RA entity and only after the verification of identity, the SIM card gets activated by this entity. Thus, this entity is responsible to validate the identity of the subscribers. It is also assumed that a symmetric key is shared between the AS and the CA/RA, which provides the proper security to all the transmitted information between the AS and the CA/RA. It is considered that various authentication servers are connected with each other through a secure channel, since one centralized server is not efficient to handle data all around. All the transmission among various AS take place by encrypting the message with a symmetric key shared between each pair of AS. Both scenarios of this protocol are as follows:

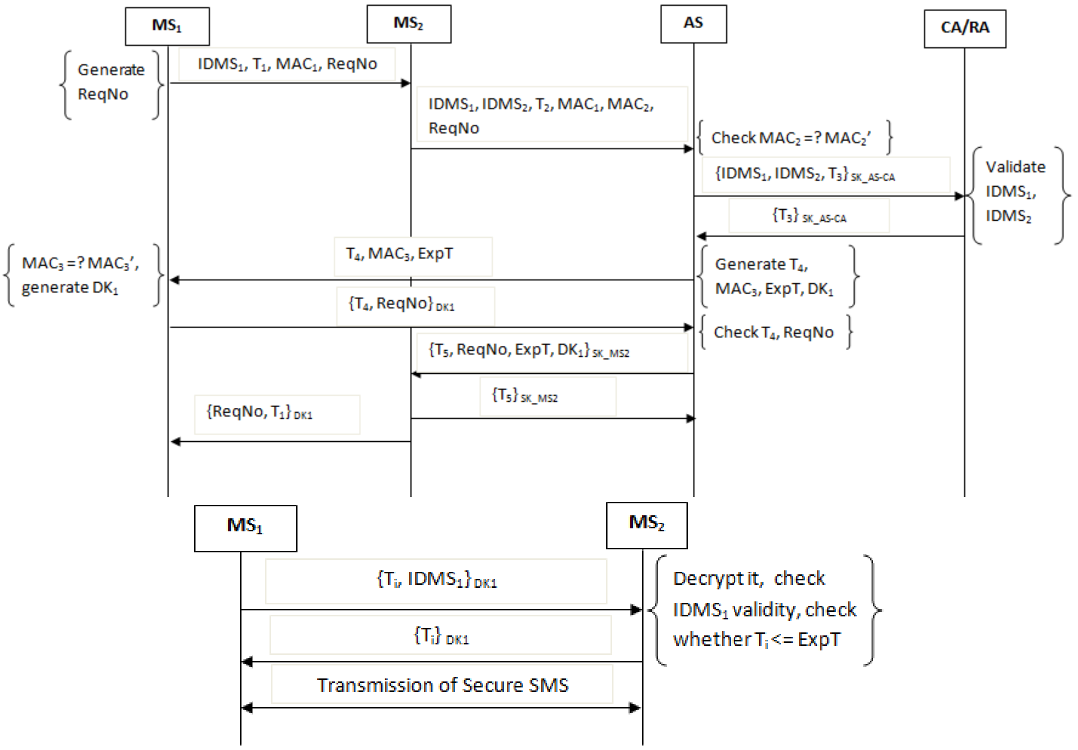

MS1
MS2
AS
CA/RA
Generate ReqNo
IDMS1, T1, MAC1, ReqNo
IDMS1, IDMS2, T2, MAC1, MAC2, ReqNo
Check MAC2 =? MAC2'
{IDMS1, IDMS2, T3}SK_AS-CA
Validate IDMS1, IDMS2
{T3} SK_AS-CA
MAC3 =? MAC3', generate DK1
T4, MAC3, ExpT
Generate T4, MAC3, ExpT, DK1
{T4, ReqNo} DK1
Check T4, ReqNo
{T5, ReqNo, ExpT, DK1} SK_MS2
{T5} SK_MS2
{ReqNo, T1} DK1
MS1
MS2
{Ti, IDMS1} DK1
Decrypt it, check IDMS1 validity, check whether Ti <= ExpT
{Ti} DK1
Transmission of Secure SMS

Figure 8.2: EasySMS protocol scenario-1: (a) phase-1 (b) phase-2

Scenario-1: When both the MS belong to same AS: This scenario is presented in Figure 8.2, where MS$_1$ sends a message to MS$_2$ and both MS belong to same AS. This scenario is subdivided into two phases.

Phase-1: First, the mobile user who wishes to send the SMS (say MS$_1$) transmits an initial request to other mobile user (say MS$_2$) for the connection. This initial request consists of IMSI of the MS$_1$ (say IDMS$_1$), a timestamp T$_1$, a request number ReqNo, and message authentication code MAC$_1$ = f$_1$(IDMS$_1$, ReqNo)$_{SK_1}$ (message (1)). The SK$_1$ is a symmetric key shared between the MS$_1$ and the AS.

MS_1 *to* MS_2*:* $IDMS_1,\ T_1,\ MAC_1,\ ReqNo$

On receiving message from the MS_1, the mobile user who receives this request (say MS_2) computes $MAC_2 = f_1(IDMS_2, T_2, MAC_1)_{SK_2}$. Then MS_2 sends a message to the AS containing the $IDMS_1$, $IDMS_2$, T_2, MAC_1, ReqNo, and MAC_2, where $IDMS_2$ is IMSI of the MS_2 (message (2)). The SK_2 is a symmetric key shared between the MS_2 and the AS. With this message, the MS_2 requests to the AS to check the validity of $IDMS_1$.

MS_2 to AS: $IDMS_1$, $IDMS_2$, T_2, MAC_1, MAC_2, ReqNo

When the AS receives a message from the MS_2, it computes MAC'_2 and compares it with the received MAC_2, where $MAC'_2 = f_1(IDMS_2, T_2, MAC_1)_{SK_2}$. If it holds then the AS sends not only $IDMS_1$ but also $IDMS_2$ to the CA/RA along with a timestamp T_3 using a symmetric shared key between the AS and the CA/RA (say SK_AS-CA) to validate the identity of both the MS (message (3)). If MAC_2 and MAC'_2 are not equal, then the connection is terminated.

AS to CA/RA: $\{IDMS_1, IDMS_2, T_3\}_{SK\_AS-CA}$

Next, the CA/RA checks the validity of both the entities and sends the reply back to the AS with the received timestamp T_3 (message (4)).

CA/RA to AS: $\{T_3\}_{SK\_AS-CA}$

On receiving the message from the CA/RA, if the AS finds any of the entities is invalid, then the connection is simply terminated and the MS_1 needs to send a fresh connection request. If both the entities are valid, then the AS generates a new timestamp T_4, an expiry time to authenticate the MS_1 (say ExpT), a delegate key DK_1 ($DK_1 = f_2(T_4, ReqNo)_{SK_1}$) generated from SK_1 using a function f_2, and a new message authentication code $MAC_3 = f_1(T_4, ExpT, ReqNo)_{SK_1}$. Then, the AS sends ($T_4$, MAC_3, ExpT) to the MS_1 (message (5)).

AS to MS_1: T_4, MAC_3, ExpT

After receiving the message from the AS, the MS_1 first computes MAC'_3 and compares it with the received MAC_3, where $MAC'_3=f_1(T_4, ExpT, ReqNo)_{SK_1}$. If both are same, then the MS_1 computes DK_1 key, where $DK_1 = f_2(T_4, ReqNo)_{SK_1}$. Next, the MS_1 sends T_4 and corresponding ReqNo to the AS encrypted with DK_1 key (message (6)).

MS_1 to AS: $\{T_4, ReqNo\}_{DK_1}$

The AS checks the received T_4 with its stored value and confirms the request number ReqNo. If both are correct, then the authentication of the MS_1 is completed. Thereafter, the AS sends DK_1 key to the MS_2 along with T_5, ExpT, and ReqNo, all encrypting using the SK of MS_2 (SK_MS_2), which is a shared key between the AS and the MS_2 (message (7)).

AS to MS_2: $\{T_5, ReqNo, ExpT, DK_1\}_{SK\_MS_2}$

The MS_2 simply confirms the reception of DK_1 key by replying to the AS, the T_5 encrypted with the SK of MS_2 (message (8)).

MS_2 to AS: $\{T_5\}_{SK\_MS_2}$

The MS_2 also sends ReqNo and T_1 to the MS_1 encrypted with DK_1 key so that the MS_1 can verify the correctness of T_1 and ReqNo (message (9)). This message also verifies the successful reception of DK_1 key by the MS_2.

MS_2 to MS_1: $\{ReqNo, T_1\}_{DK_1}$

Phase-2: Once both MS have a shared secret symmetric key, they can exchange the message information in a secure manner using a suitable and strong cryptographic algorithm like AES/ MAES (explained later). After the phase-1, a session is generated, which provides the secure communication between both the MS for a specified time period ExpT. In this time period, the same DK_1 key is used to provide ciphering between the MS_1 and the MS_2, but, after the ExpT time, the session gets expired and the MS_1 needs to send a fresh request to the MS_2 including a new request number ReqNo with the same procedure of phase-1. Within the ExpT, the following steps are used for secure communication between both the MS:

The MS_1 sends $IDMS_1$ and a timestamp (say T_i) to the MS_2 encrypted with symmetric key of the MS_1, *i.e.*, DK_1 (message (1)).

MS_1 to MS_2: $\{T_i, IDMS_1\}_{DK_1}$

The MS_2 decrypts message using the same DK_1 key, checks the validity of $IDMS_1$ and verifies whether $T_i <= ExpT$. If both are correct, then the MS_1 is successfully authenticated and proved as a valid user for the connection. Then MS_2 replies the received T_i encrypted with DK_1 key as an acknowledgement to the MS_1 (message (2)).

MS_2 to MS_1: $\{T_i\}_{DK_1}$

Now, a secure SMS communication between both the MS takes place (message (3)).

Scenario-2: When both the MS belong to different AS: This scenario is presented in Figure 8.3, where the MS_1 sends a message to the MS_2, while both MS belong to different AS. This case is one where both the mobile users are located

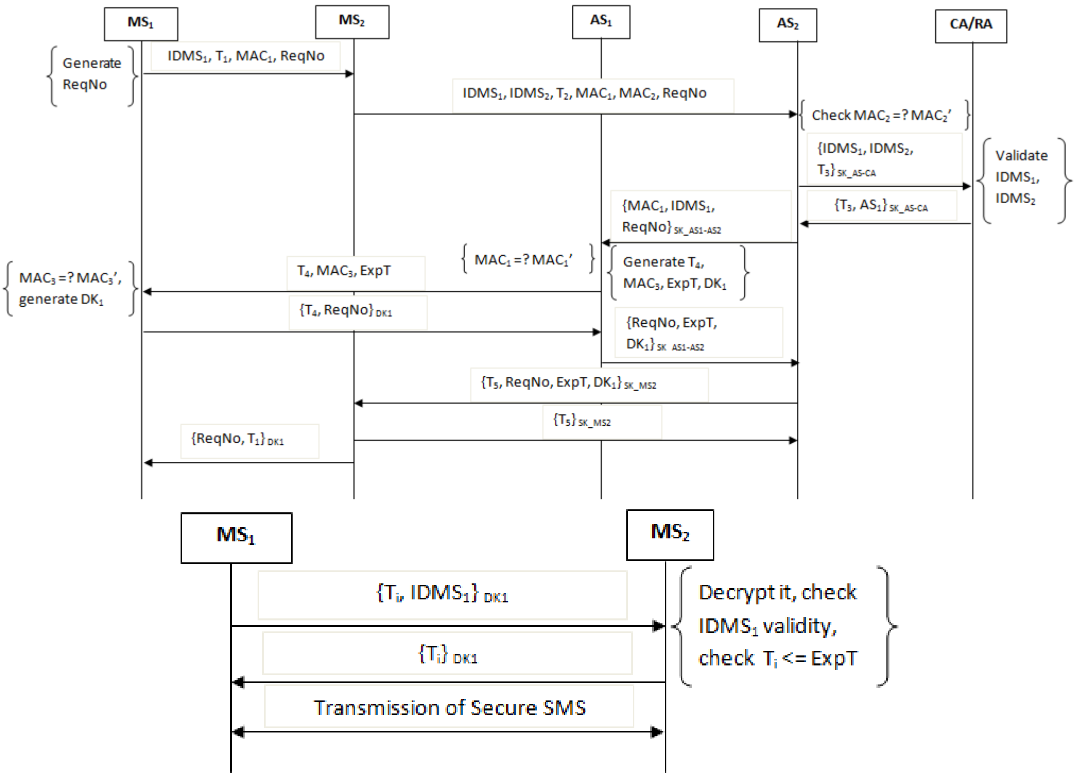

MS₁
MS₂
AS₁
AS₂
CA/RA
Generate ReqNo
IDMS₁, T₁, MAC₁, ReqNo
IDMS₁, IDMS₂, T₂, MAC₁, MAC₂, ReqNo
Check MAC₂ =? MAC₂′
{IDMS₁, IDMS₂, T₃} SK_AS-CA
Validate IDMS₁, IDMS₂
{T₃, AS₁} SK_AS-CA
{MAC₁, IDMS₁, ReqNo} SK_AS1-AS2
MAC₁ =? MAC₁′
Generate T₄, MAC₃, ExpT, DK₁
T₄, MAC₃, ExpT
MAC₃ =? MAC₃′, generate DK₁
{T₄, ReqNo} DK1
{ReqNo, ExpT, DK₁} SK AS1-AS2
{T₅, ReqNo, ExpT, DK₁} SK_MS2
{T₅} SK_MS2
{ReqNo, T₁} DK1
MS₁
MS₂
{Tᵢ, IDMS₁} DK1
Decrypt it, check IDMS₁ validity, check Tᵢ <= ExpT
{Tᵢ} DK1
Transmission of Secure SMS

Figure 8.3: EasySMS protocol scenario-2: (a) phase-1 (b) phase-2

in the geographically far areas and have different authentication centers. It may be the case, where both MS are of different service providers (however, each service provider has to trust other providers), thus, they genuinely have different authentication centers. This scenario is also subdivided into two phases and is as follows:

Phase-1: First step (message (1)) is same as presented in step-1 of scenario-1. Here, SK_1 is a symmetric key shared between the MS_1 and the AS_1.

MS_1 to MS_2: $IDMS_1$, T_1, MAC_1, ReqNo

The MS_2 passes ($IDMS_1$, $IDMS_2$, ReqNo, T_2, MAC_1, MAC_2) to the AS, through which it is connected (say AS_2) (message (2)). The SK_2 is a symmetric key shared between the MS_2 and the AS_2. With this message, the MS_2 requests to the AS_2 to check the validity of $IDMS_1$. The MS_2 stores the timestamp T_1 in the memory that was received from the MS_1.

MS_2 to AS_2: $IDMS_1$, $IDMS_2$, T_2, MAC_1, MAC_2, ReqNo

The AS_2 computes the same as presented in step-3 of scenario-1 (message (3)) and checks whether MAC_2 ?= MAC'_2.

AS_2 to CA/RA: $\{IDMS_1, IDMS_2, T_3\}_{SK\_AS-CA}$

The CA/RA checks the validity of both the entities and sends the reply back to the AS$_2$ with the received timestamp T$_3$ and the identity of AS to which the MS$_1$ belongs (say AS$_1$) (message (4)).

CA/RA to AS$_2$: $\{T_3, AS_1\}_{SK\_AS-CA}$

The AS$_2$ checks the same as in scenario-1 step-5, if both entities are valid, then the AS$_2$ sends (IDMS$_1$, ReqNo, MAC$_1$) to the AS$_1$ through a secure channel or using a symmetric key shared between the AS$_1$ and the AS$_2$ (say SK_AS$_1$-AS$_2$) (message (5)). It is assumed that all the AS communicate with each other using the pre-computed symmetric shard keys.

AS$_2$ to AS$_1$: $\{MAC_1, IDMS_1, ReqNo\}_{SK\_AS_1-AS_2}$

When the AS$_1$ receives the message from the AS$_2$, it computes MAC$'_1$= f$_1$(IDMS$_1$, ReqNo)$_{SK_1}$ and compares MAC$'_1$ with the received MAC$_1$. If both are different, then the connection is terminated. If both are same, then the AS$_1$ generates a new timestamp T$_4$, an expiry time to authenticate the MS$_1$ (say ExpT), DK$_1$ key generated from SK$_1$ of the MS$_1$ using a function f$_2$, and MAC$_3$, where MAC$_3$ = f$_1$(T$_4$, ExpT, ReqNo)$_{SK_1}$ and DK$_1$ = f$_2$(T$_4$, ReqNo)$_{SK_1}$. Then the AS$_1$ sends (T$_4$, MAC$_3$, ExpT) to the MS$_1$ (message (6)).

AS$_1$ to MS$_1$: $T_3, MAC_3, ExpT$

After receiving the message from the AS$_1$, the MS$_1$ repeats the same as in scenario-1 step-6 and sends (T$_4$, ReqNo) to the AS$_1$ encrypted with DK$_1$ key (message (7)).

MS$_1$ to AS$_1$: $\{T_4, ReqNo\}_{DK_1}$

The AS$_1$ checks T$_4$ and ReqNo as in scenario-1 step-7. Then, AS$_1$ conveys confirmation of authentication of MS$_1$ by sending a message (ReqNo, ExpT, DK$_1$) to the AS$_2$ using SK_AS$_1$-AS$_2$ key (message (8)).

AS$_1$ to AS$_2$: $\{ReqNo, ExpT, DK_1\}_{SK\_AS_1-AS_2}$

The AS$_2$ sends DK$_1$ key to the MS$_2$ along with T$_5$, ExpT, and ReqNo, all encrypting using the SK of the MS$_2$ (say SK_MS$_2$), which is a shared key between the AS$_2$ and the MS$_2$ (message (9)).

AS$_2$ to MS$_2$: $\{T_5, ReqNo, ExpT, DK_1\}_{SK\_MS_2}$

The MS$_2$ repeats the same as in scenario-1 step-8, and sends encrypted reply of T$_5$ to the AS$_2$ (message (10)).

MS$_2$ to AS$_2$: $\{T_5\}_{SK\_MS_2}$

This step is same as in scenario-1 step-9 (message (11)).

MS$_2$ to MS$_1$: $\{ReqNo, T_1\}_{DK_1}$

Phase-2: It is same as discussed in the previous scenario of phase-2.

MS_1 to MS_2: $\{T_i, IDMS_1\}_{DK_1}$ (message (1))

MS_2 to MS_1: $\{T_i\}_{DK_1}$ (message (2))

8.4 Security Analysis of EasySMS Protocol

This section analyzes proposed protocol in various aspects such as mutual authentication, prevention from various threats and attacks, key management, and computation and communication overheads.

Is the Secret Key SK Safely Stored?: Since the malicious user does not know the structure of cryptographic functions like $\mathrm{f}_1()$ and $\mathrm{f}_2()$, thus, he/she can neither generate the correct MAC_1 nor correct DK_1 key. Further, the SK key is stored on the AS as well as embedded onto the SIM at the time of manufacturing, thus, it is almost impossible to extract SK key. The storage scenario of SK key, we presented is same as nowadays used for the voice communication in traditional cellular network. If any or some service providers do not wish to use actual SK key in the protocol execution, then they can simply compute alternate secret keys as: $\mathrm{SK'}_1 = \mathrm{f''}(\mathrm{IDMS}_1)_{SK_1}$ and $\mathrm{SK'}_2 = \mathrm{f''}(\mathrm{IDMS}_2)_{SK_2}$. We do not prefer it because it increases overall overhead of the protocol.

Is there any Alternative for IMSI?: Since a malicious user with only known IMSI (by some IMSI catcher but, the functions and secret keys are still unknown) cannot break the security of proposed protocol. Thus, the proposed protocol is secure, but, there is one alternate for it. In this respect, a new function $\mathrm{f'}()$ can be proposed, which computes a temporary identity TMSI for each MS whenever it wishes to communicate, a *MS*: computes IDMS_1=$\mathrm{f'}(\mathrm{IMSI}_1, \mathrm{MAC}_1)$; a *AS*: computes IMSI_1= $\mathrm{f'}(\mathrm{IDMS}_1, \mathrm{MAC}_1)$. This is simply possible by XOR the IMSI_1 (or IDMS_1) and MAC_1 (twice), because the size of MAC_1 is 64 bits while $\mathrm{IMSI}_1/\mathrm{IDMS}_1$ is of 128 bits. The function $\mathrm{f'}()$ should be known to the MS as well as the AS but, publically unknown. It is recommended to use a complex function to compute the same. However, it is not preferred because it increases the overhead at MS as well as at AS.

8.4.1 Mutual Authentication between the MS and the AS

In scenario-1 of EasySMS protocol, the AS authenticates the MS_1 by verifying MAC_2 and checks the identity of MS_1 by the CA/RA. When the AS receives MAC_2, it simply calculates MAC'_2 and compares it with the received MAC_2. If it matches, then authentication of MS_1 is done by the AS. Similarly, on receiving MAC_3, the MS_1 computes MAC'_3 to authenticate the AS. If MAC_3 is equal to the MAC'_3, then the authentication of AS is successful. All this ensures the mutual authentication between the MS_1 and the AS through MS_2. Similarly, in scenario-2, the AS_1 authenticates the MS_1 through AS_2 and MS_2. The integrity is maintained between MS_1-AS_1 and MS_1-AS_1 by comparing MAC_1-MAC'_1 and MAC_2-MAC'_2 respectively. The MS_1 authenticates the AS_1 by comparing MAC_3 with MAC'_3.

8.4.2 Efficient Key Management

The EasySMS protocol is able to efficiently handle the key management in both the scenarios, where the generated DK_1 key (from the symmetric key of MS_1) is securely transmitted by the AS to the MS_2 (in scenario-1) or by the AS_2 to the MS_2 through the AS_1 (in scenario-2). Thus, this scheme ciphers the message before its transmission over the network. Further, it is preferred to have a symmetric key algorithm for the encryption because these algorithms are 1000 times faster than the asymmetric algorithms [69] and improve the efficiency of the system.

8.4.3 Resistance to Attacks

This subsection justifies that the EasySMS protocol is able to prevent the transmitted SMS from various attacks over the network. It is assumed that the cryptographic functions used in this chapter are not publically available and are secret. The capturing of any SK key is not possible because no secret key has been transmitted in any phase of the proposed protocol and always a DK_1 key is being transferred in the cipher mode, whenever is required. Secret keys are also not publically available and are secret.

1. *SMS Disclosure:* In the EasySMS protocol, a cryptographic encryption algorithm AES/MAES is maintained to provide end-to-end confidentiality to the transmitted SMS over the network. Thus, encryption approach prevents the transmitted SMS from SMS disclosure.

2. *Replay Attack:* The proposed protocol is free from this attack because it sends one timestamp (T_1, T_2, T_3, T_4, and T_5) with each message during the communication over the network.

3. *Man-in-the-middle Attack:* In the EasySMS protocol, AES/MAES is used to encrypt end-to-end communication between the MS and the AS in both the scenarios. The message is end-to-end securely encrypted with DK_1 key for every subsequent authentication. Since, the attacker does not have sufficient information to generate DK_1, hence, it prevents the communication from MITM attack over the network.

4. *Over the Air Modification in SMS Transmission:* The EasySMS protocol provides end-to-end security to the SMS from sender to the receiver including OTA interface with an additional strong encryption algorithm AES/MAES. The protocol does not depend upon the cryptographic security of encryption algorithm (such as A5/1, A5/2) exists between the MS and the BTS in traditional cellular network.

5. *Impersonation Attack:* There are two cases to evaluate this attack with the EasySMS protocol. Both the cases are as follows:
   a) When an attacker impersonates the MS: In the EasySMS protocol, if an attacker tries to impersonate the MS, he/she will not get success because in scenario-1, the AS calculates the MAC'_2 and compares it with the received MAC_2, while in scenario-2, the AS_2 computes MAC'_2 and compares with MAC_2. If it holds, then the AS_1 computes MAC'_1 and checks whether MAC'_1 is equal to MAC_1. Thus, at any stage, if the AS finds the above comparison false, then the connection is simply terminated.
   b) When an attacker impersonates the AS: If an attacker tries to impersonate the AS (or AS_1/AS_2), an attempt to impersonate the AS will be failed as the MS_1 computes MAC'_3 and compares it with the received MAC_3. Thus, an attempt to impersonate the AS terminates the connection.

8.5 Performance Analysis of EasySMS Protocol

This section discusses performance evaluation of the EasySMS protocol in terms of communication overhead, computation overhead, and bandwidth utilization.

8.5.1 Computation Overhead

To compute the overhead, all security functions used in EasySMS, SMSSec and PK-SIM protocols are considered of unit value. On the basis of authentication requests n and number of functions used in three protocols, the calculation for computation overhead is as follows:

1. *SMSSec Protocol: Phase-1:* [H, $\{\}_{PK}$, $\{\}_{SK}$, $\{\}_{SK}$, $\{\}_{SK}$] = 5
   Phase-2: [H, HU, $\{\}_{SK}$, $\{\}_{SK\_n}$, $\{\}_{SK\_n}$, $\{\}_{SK\_n}$]*n = 6*n
   Total Computation Overhead = 5+6*n

2. *PK-SIM Protocol: Phase-1:* [H(CertSAG), $\{\}_{SK\_SAG}$, H(C_ME), $\{\}_{SK\_SAG}$, H(Ns, Nc, UAKey, Expiry), $\{\}_{SK\_SAG}$, $\{\}_{PK\_PK-SIM}$, $\{\}_{E\_UAKey}$]=8
   Phase-2: [MAC, $\{\}_{E\_SK}$, MAC$'$, $\{\}_{E\_SK}$]*n = 4*n
   Total Computation Overhead = 8+4*n

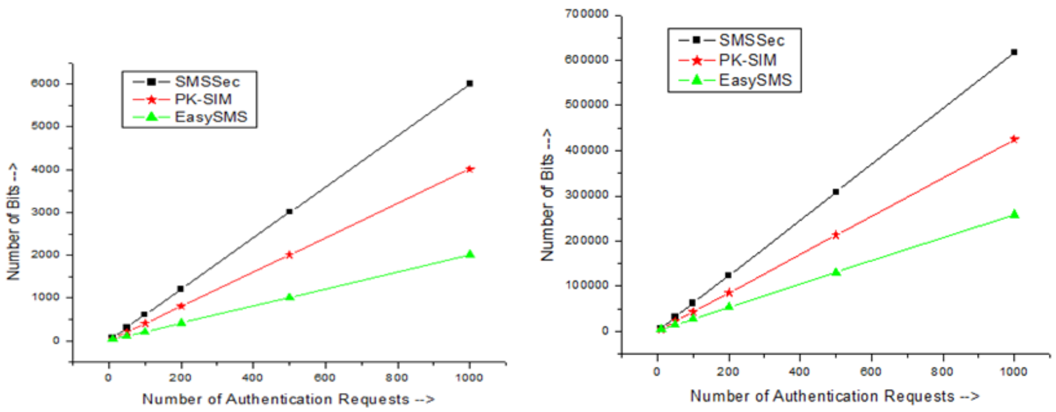


Figure 8.4: SMSSec, PK-SIM, and EasySMS protocol (a) computation overhead (b) communication overhead

3. *EasySMS Protocol: Scenario-1: Phase-1:* f_1, f_1, f_1, f_1, f_1, f_2, f_2, $\{\}_{SK\_AS-CA}$, $\{\}_{SK\_AS-CA}$, $\{\}_{SK\_MS_2}$, $\{\}_{SK\_MS_2}$, $\{\}_{DK_1}$, $\{\}_{DK_1}$ = 13

Phase-2: $[\{\}_{DK_1}, \{\}_{DK_1}]$*n = 2*n

Total Computation Overhead = 13+2*n

Scenario-2: *Phase-1:* f_1, f_1, f_1, f_1, f_1, f_1, f_2, f_2, $\{\}_{SK\_AS-CA}$, $\{\}_{SK\_AS-CA}$, $\{\}_{SK\_AS_1-AS_2}$, $\{\}_{SK\_AS_1-AS_2}$, $\{\}_{SK\_MS_2}$, $\{\}_{SK\_MS_2}$, $\{\}_{DK_1}$, $\{\}_{DK_1}$ = 16

Phase-2: $[\{\}_{DK_1}, \{\}_{DK_1}]$*n = 2*n

Total Computation Overhead = 16+2*n

8.5.2 Communication Overhead

The communication overhead of EasySMS, SMSSec, and PK-SIM protocols are evaluated. The total number of transmitted bits in each protocol (with the help of size specified in Table 8.1) are as follows:

1. *SMSSec Protocol:* Here, random number Rc is of 128 bits.

   Phase-1: (1)+(2)+(3)+(4) = (40+64+64+ 28+128)+(128+16 +28) +(28)+ (28) = 552 bits

   Phase-2: (for n values) = ((1)+(2)+(3)+ (4))*n = ((64+40 +64+64+28+ 128)+ (128+ 16+28)+ (28)+(28))*n = 616*n

   Total transmitted bits = 552+616*n

2. *PK-SIM Protocol: Phase-1:* (1)+(2)+(3)+(4)+(5) =(40+128+64+28)+(40) + (40+64)+ (128+128+64+ 64)+ (128) = 916 bits

   Phase-2: (for n values) = ((1)+(2))*n =((40+128+64)+(128+64))*n = 424*n

   Total transmitted bits = 916+424*n

3. *EasySMS Protocol: Scenario-1: Phase-1:* (1)+(2)+(3)+(4)+(5)+(6)+(7)+(8) +(9) = (128+64+64+8) + (128+128+64+64+64 +8) + (128+128+64)+(64) + (64+64+64) + (64+8) + (64+8+64+ 256) + (64) + (64+8) = 1896 bits

   Phase-2: ((1)+(2))*n = ((64+ 128)+(64))*n = 256*n bits

   Total transmitted bits = 1896+256*n

   Scenario-2: Consider the identity of AS1 is 128 bits. *Phase-1:* (1)+(2)+ (3)+(4) +(5)+(6) +(7)+ (8)+(9)+(10)+(11) = (128+64+64+8) + (128+128 +64+64+64 +8) + (128+128+ 64) +(64+128)+(64+128+8)+(64+64+64) + (64+8) + (8+64 +256) + (64+8+64+256) + (64) + (8+64) = 2552 bits

   Phase-2: ((1))+(2))*n = ((64+128) + (64)*n = 256*n bits

Total transmited bits = 2552+256*n

Figure 8.4 shows graphs between the number of bits for overhead and number of authentication requests generated. It can be clearly observed that EasySMS generates lesser computation overhead (Figure 8.4(a)) and communication overhead (Figure 8.4(b)) as compared to SMSSec and PK-SIM protocols.

Table 8.3: Bandwidth utilization of various protocols

No. of Authentication Requests	EasySMS/SMSSec	EasySMS/PK-SIM
10	0.76	0.99
50	0.48	0.69
100	0.45	0.64
200	0.43	0.62
500	0.42	0.61
1000	0.41	0.60
Average	0.49	0.69

Table 8.4: Message exchange ratio of various protocols

No. of Authentication Requests	EasySMS/SMSSec	EasySMS/PK-SIM
10	0.55	0.75
50	0.38	0.55
100	0.35	0.52
200	0.34	0.51
500	0.33	0.50
1000	0.33	0.50
Average	0.38	0.55

8.5.3 Bandwidth Utilization

This subsection evaluates the bandwidth utilized by all three protocols and verifies them with respect to each other. Table 8.3 presents the bandwidth utilization of EasySMS with respect to SMSSec and PK-SIM protocols, where on average, the EasySMS protocol lowers 51% and 31% of the bandwidth consumption as compared to both protocols, when n = 10, 50, 100, 200, 500, 1000. Similarly, Table 8.4 shows that the proposed protocol diminishes 62% and 45% of the message exchanged ratio during the authentication in comparison to SMSSec and PK-SIM protocols respectively.

8.6 Simulation and Evaluation of Encryption Algorithms

This section focuses on the selection criteria to choose a block cipher-based symmetric key algorithm. The efficiency of a block cipher algorithm depends upon the block size and key size. Since, with a larger block size we can encrypt large chunk of data in one cycle of the algorithm, thus, it speeds up the execution of algorithm. However, a larger key results in a slower algorithm, because in general, all bits of key are involved in an execution cycle of the algorithm. A large number of rounds make the algorithm slower but, are supposed to provide greater security [120]. Thus, there is always a trade off between security and performance in block cipher algorithms [121]. Biham E. [122] has suggested that performance of algorithm should be measured by timing for the minimum number of secure rounds for each algorithm, *i.e.*, the estimated number of rounds to make a brute force key search, which is the most efficient form of attack. However, there is no easy way of obtaining impartial and widely accepted values for the minimum number of secure rounds for each algorithm.

8.6.1 Simulation

Some existing symmetric key algorithms like DES, Triple-DES with 2-keys, Triple-DES with 3-keys, and AES have been implemented and verified. J2ME with WMA implementation of these algorithms is limited with 160 characters only, *i.e.*, single SMS, while JDK 1.6 is used for these algorithms with more than 160 characters. The application can send and receive SMS messages in binary format using the WMA [123]. Since the J2ME environment does not contain cryptographic algorithms, it is recommended to use the Lightweight API from the Legion of the Bouncy Castle. The standard key size used in DES, Triple DES with 2-keys, Triple-DES with 3-keys, and AES, are 64 (out of which 56 bits are used), 112, 168, and 128 bits respectively.

Figure 8.5(a) and Figure 8.5(b) show the results for the encryption and decryption with DES, Triple-DES with 2-keys, Triple-DES with 3-keys, and AES. The results conclude that out of these algorithms, AES takes minimum time to encrypt and decrypt the SMS with various sizes, where one SMS size is

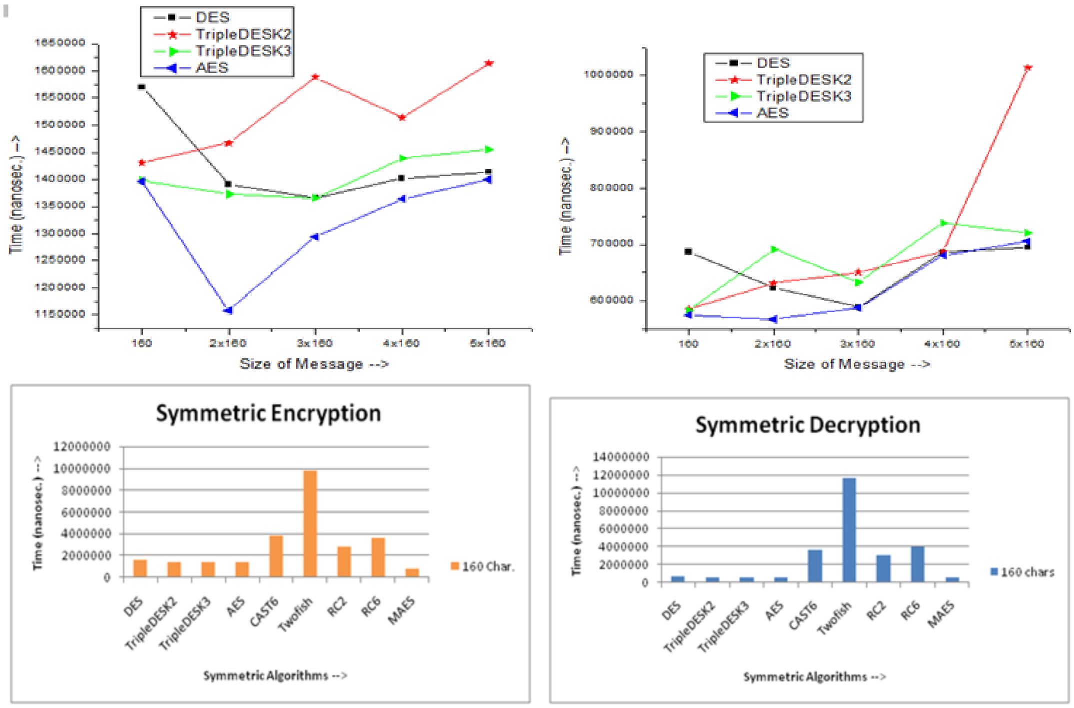


Figure 8.5: Performance of various symmetric algorithms (left to right, top to bottom) (a) encryption of 160, 160x2, 160x3, 160x4, 160x5 char. message size, (b) decryption of 160, 160x2, 160x3, 160x4, 160x5 char. message size, (c) encryption of 160 char. message size (d) decryption of 160 char. message size

Table 8.5: Message size (plaintext, ciphertext)

Mode/ Algorithm	DES	TripleDES2K	TripleDES3K	AES
PCBC	160, 80	160, 155	160, 155	160, 80
ECB	160, 143	160, 160	160, 168	160, 80
CBC	160, 80	160, 156	160, 156	160, 155
CTR	160, 82	160, 82	160, 82	160, 160
CFB	160, 82	160, 82	160, 159	160, 161
OFB	160, 161	160, 82	160, 82	160, 82

160 characters. Table 8.5 represents the pairs of plain text and cipher text with respect to various algorithms DES, AES, Triple-DES with 2-keys, and Triple-DES with 3-keys implemented in various modes of operations like PCBC, ECB, CBC, CTR, OFB, and CFB. Out of all these modes, CTR mode is most popular and usable because it provides the parallelism to encrypt and decrypt all blocks of data simultaneously. Nowadays, DES and Triple-DES algorithms are not considered as very secure algorithms [124], [125], since, previously some attacks have been found on both algorithms. Thus, AES is the best option for this purpose. With the input of 160 characters, the DES, AES, Triple-DES with 2-keys, and Triple-DES with 3-keys algorithms with CTR mode generate 82,

82, 82, and 160 characters cipher respectively, which mean one can still send 160 characters after encrypting the SMS by AES. Each algorithm results are calculated 30 times by repeating execution and the average value is considered.

8.6.2 Reliability Analysis with Confidence Interval

The range of confidence interval for various encryption algorithms are calculated by considering the interval 95% for each algorithm with 160 characters as input because the margin of error is about twice the standard deviation [126]. Confidence interval is an interval estimate of a population parameter and is used to indicate the reliability of an estimate. Figure 8.6(a), 8.6(b), 8.6(c), 8.6(d), and 8.6(e) represent the range of confidence interval (high & low range values) for the encryption (E_low_interval, E_high_interval) and decryption (D_low_interval, D_high_interval) of the message (SMS) with 160, 320, 480, 640, and 800 characters in length for DES, Triple-DES with 2-keys, Triple-DES with 3-keys, and AES algorithms. Here, the unit of time is nanoseconds. The t-distribution is used to calculate the confidence interval because it computes confidence intervals for small n (30 samples in our analysis), if the data is not normally distributed [127]. In this process, the SMS size from 160 to 800 characters is evaluated, where more than 160 characters in an SMS is split and concatenated with another SMS. In other words, the transmitted message can contain a range of 1120 bits to 56000 bits, where each character is mapped with 7-bits ASCII value. A low standard deviation indicates that data points tend to be very close to the mean, whereas high standard deviation indicates that data points are spread out over a large range of values. Since the AES is strict to its output range as compared to DES, TripleDESK2, and TripleDESK3. Hence, it is considered the best among them.

8.6.3 A Variant of AES: MAES Algorithm

The AES with 128-bit key has been proved to be an efficient algorithm to encrypt the SMS but, its security cannot be remain maintained in the subsequent years. Various researchers have found attacks (partial) on AES 128-bit key [89], [128] with some assumptions. Thus, a variant of AES namely MAES (modified AES) is proposed, which is more secure with 256-bit key (as original AES) and 256-bit

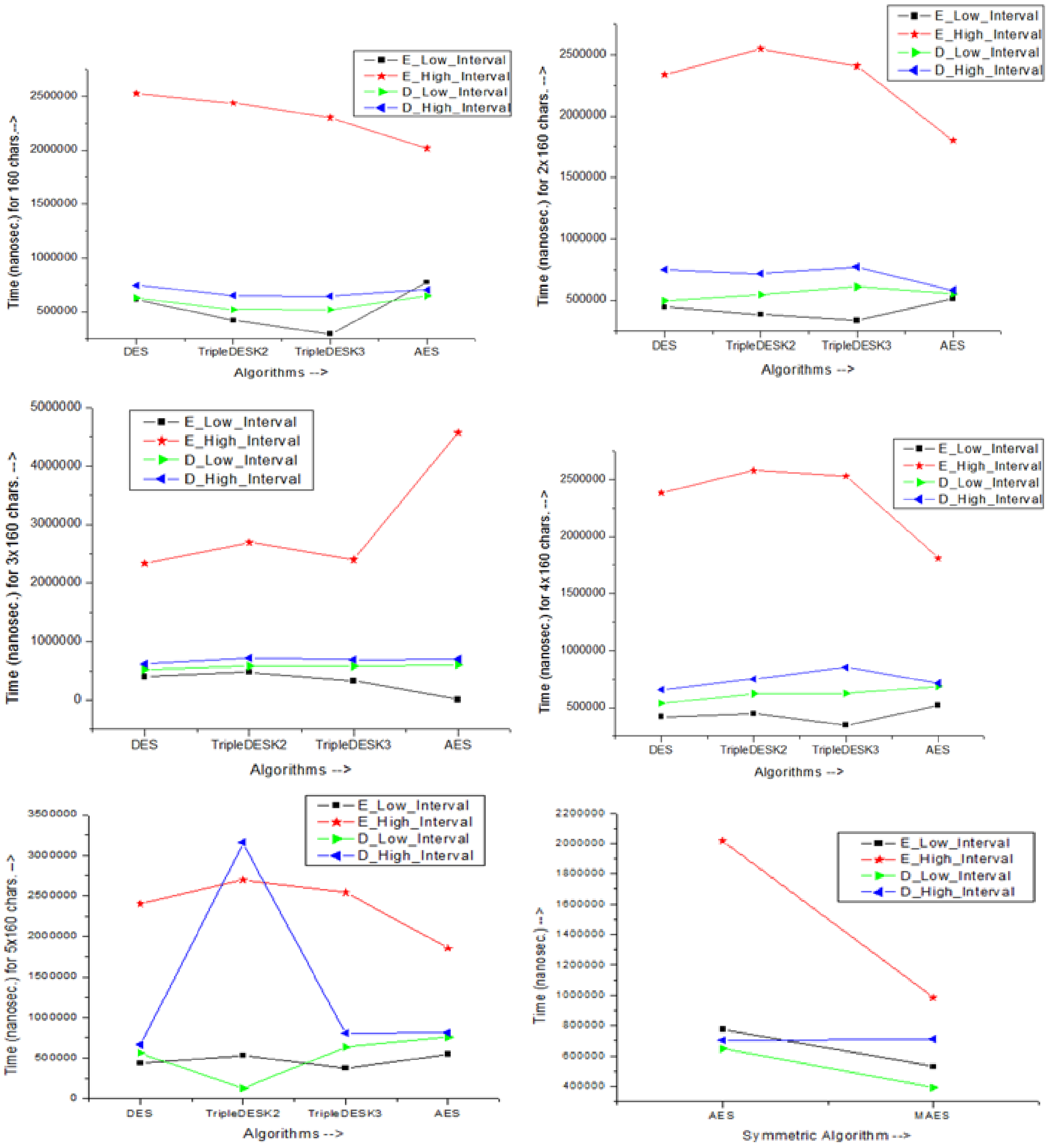


Figure 8.6: Confidence interval of different algorithms for SMS size (left to right, top to bottom) (a) 160 (b) 2x160 (c) 3x160 (d) 4x160 (e) 5x160 (f) 160 chars. with AES and MAES

each block of data. The increase in length of each block improves the performance of MAES than the original AES. Various steps of MAES algorithm are as follows:

1. *Key Generation:* In the EasySMS protocol, 256-bit of DK_1 key is generated at MS_1 as well as at AS, which is used as cipher key for the MAES algorithm and round keys are derived from this 256 bits cipher key using AES key schedule.

2. *Initial Round:* AddRoundKey - each byte of the state is combined with the round key using bitwise-XOR.

3. *Rounds:* (i) SubBytes - a non-linear substitution step where each byte is replaced with another according to a lookup table, (ii) ShiftRows - a transposition step where each row of the state is shifted cyclically a certain number of steps, (iii) MixColumns - a mixing operation, which operates on the columns of the state, (iv) AddRoundKey.

4. *Final Round (no MixColumns):* (i) SubBytes (ii) ShiftRows (iii) AddRoundKey

For MAES, we have considered that SubByte and ShiftRow are swapped and an alternative matrix is used (in MixColumns), which is same as its inverse, as explained in section 4.6.2. Table 8.6 shows the performance of AES and MAES algorithms with one SMS size of plaintext and ciphertext pairs in bits and characters, where MAES generates 158 characters after ciphering the SMS of 160 characters. Various algorithms DES, Triple-DES, AES, CAST6, Twofish, RC2, RC6, MAES have been implemented and the encryption/decryption time of SMS with 160 characters are evaluated, which is clearly shown in Figure 8.5(c) and Figure 8.5(d). Finally, it is concluded that out of these algorithms (DES, TripleDESK2, TripleDESK3, AES, MAES), the MAES algorithm is more efficient to encrypt the SMS. The confidence interval for AES and MAES can be observed from Figure 8.6(f), where confidence interval (high & low range values) of MAES is strictly close to the encryption process.

Table 8.6: SMS size (input, output)

SMS Size	AES	MAES
1 SMS (in bits)	1120, 1540	1120, 1111
1 SMS (in char.)	160, 220	160, 158

8.7 Formal Proof of EasySMS Protocol

BAN-Logic symbols are used in order to formally proof the authentication process of EasySMS protocol. Various notations used in BAN-Logic are described in section 3.7.4.

1. *The Formal Messages in EasySMS Protocol:*

   Phase-1: (1) $\mathrm{MS}_1 \rightarrow \mathrm{MS}_2$: ID_1, Ta, ReqNo, $\mathrm{f}_1(\mathrm{ID}_1, \mathrm{ReqNo})_{SK_1}$; $\mathrm{MS}_1 \overset{SK_1}{\leftrightarrow} \mathrm{AS}_1$

   (2) $\mathrm{MS}_2 \rightarrow \mathrm{AS}_2$: ID_1, ID_2, Tb, ReqNo, $\mathrm{f}_1(\mathrm{ID}_1, \mathrm{ReqNo})_{SK_1}$, $\mathrm{f}_1(\mathrm{ID}_2, \mathrm{Tb}, \mathrm{f}_1(\mathrm{ID}_1, \mathrm{ReqNo})_{SK_1})_{SK_2}$; $\mathrm{MS}_2 \overset{SK_2}{\leftrightarrow} \mathrm{AS}_2$

   (3) $\mathrm{AS}_2 \rightarrow$ CA/RA: $\{\mathrm{ID}_1, \mathrm{ID}_2, \mathrm{Tc}\}_{SK\_AS-CA}$; $\forall\ \mathrm{AS}_i \overset{SK\_AS_i-CA}{\leftrightarrow}$ CA

   (4) CA/RA $\rightarrow \mathrm{AS}_2$: $\{\mathrm{AS}_1, \mathrm{Tc}\}_{SK\_AS-CA}$

   (5) $\mathrm{AS}_2 \rightarrow \mathrm{AS}_1$: $\{\mathrm{ID}_1, \mathrm{ReqNo}, \mathrm{f}_1(\mathrm{ID}_1, \mathrm{ReqNo})_{SK_1}\ \}_{SK\_AS_1-AS_2}$; $\forall\ \mathrm{AS}_i \overset{SK\_AS_i-AS_j}{\leftrightarrow} \mathrm{AS}_j$; where $i \neq j$

   (6) $\mathrm{AS}_1 \rightarrow \mathrm{MS}_1$: Td, Expirytime, $\mathrm{f}_1(\mathrm{Td}, \mathrm{Expirytime}, \mathrm{ReqNo})_{SK_1}$

   (7) $\mathrm{MS}_1 \rightarrow \mathrm{AS}_1$: $\{\mathrm{Td}, \mathrm{ReqNo}\}_{DK_1}$; $\mathrm{MS}_1 \overset{DK_1}{\leftrightarrow} \mathrm{AS}_1$

   (8) $\mathrm{AS}_1 \rightarrow \mathrm{AS}_2$: $\{\mathrm{ReqNo}, \mathrm{Expirytime}, \mathrm{f}_2(\mathrm{Td}, \mathrm{ReqNo})_{SK_1}\}_{SK\_AS_1-AS_2}$

   (9) $\mathrm{AS}_2 \rightarrow \mathrm{MS}_2$: $\{\mathrm{Te}, \mathrm{ReqNo}, \mathrm{Expirytime}, \{\mathrm{f}_2(\mathrm{Td}, \mathrm{ReqNo})_{SK_1}\ \}_{SK\_AS_1-AS_2}\ \}_{SK_2}$

   (10) $\mathrm{MS}_2 \rightarrow \mathrm{AS}_2$: $\{\mathrm{Te}\}_{SK_2}$

   (11) $\mathrm{MS}_2 \rightarrow \mathrm{MS}_1$: $\{\mathrm{Ta}, \mathrm{ReqNo}\}_{DK_1}$

   Phase-2: (1) $\mathrm{MS}_1 \rightarrow \mathrm{MS}_2$: $\{\mathrm{T}_i, \mathrm{ID}_1\}_{DK_1}$

   (2) $\mathrm{MS}_2 \rightarrow \mathrm{MS}_1$: $\{\mathrm{T}_i\}_{DK_1}$

2. *Security Assumptions:*

   a) It is assumed that SK key is shared between each MS and the AS.

   (1) Each MS has a secure key SK and MS $|\equiv$ MS $\overset{SK}{\leftrightarrow}$ AS

   (2) AS has a secure key SK and AS $|\equiv$ MS $\overset{SK}{\leftrightarrow}$ AS

   b) It is assumed that the AS trusts the CA/RA using a secret key.

   (1) CA/RA $|\equiv$ CA/RA $\overset{SK\_AS-CA}{\leftrightarrow}$ AS

(2) AS $|\equiv$ CA/RA $\overset{SK\_AS-CA}{\leftrightarrow}$ AS

c) It is assumed that communication between all the AS are done with a secret key shared between each pair of AS, *i.e.*, AS$_i$ $|\equiv$ AS$_i$ $\overset{SK\_AS_1-AS_2}{\leftrightarrow}$ AS$_j$ and AS$_j$ $|\equiv$ AS$_j$ $\overset{SK\_AS_1-AS_2}{\leftrightarrow}$ AS$_i$, where i $\neq$ j.

3. *Security Analysis:*

Phase-1: (1) MS$_1$ $\rightarrow$ MS$_2$: MS$_1$ $|\equiv$ #(Ta) $\wedge$ AS$_1$ $|\equiv$ #(Ta); MS$_2$ $\triangleleft$ ID$_1$, Ta, ReqNo, f$_1$(ID$_1$, ReqNo)$_{SK_1}$; MS$_1$ $\overset{SK_1}{\leftrightarrow}$ AS$_1$

(2) MS$_2$ $\rightarrow$ AS$_2$: MS$_2$ $|\equiv$ #(Tb) $\wedge$ AS$_2$ $|\equiv$ #(Tb); AS$_2$ $\triangleleft$ ID$_1$, ID$_2$, Tb, ReqNo, f$_1$(ID$_1$, ReqNo)$_{SK_1}$, f$_1$(ID$_2$, Tb, f$_1$(ID$_1$, ReqNo)$_{SK_1}$)$_{SK_2}$; MS$_2$ $\overset{SK_2}{\leftrightarrow}$ AS$_2$

(3) On receiving, the AS$_2$ calculates f$_1$(ID$_2$, Tb, f$_1$(ID$_1$, ReqNo)$_{SK_1}$)$_{SK_2}$, and if it matches then AS$_2$ $\rightarrow$ CA/RA: $\{$ID$_1$, ID$_2$, Tc$\}_{SK_AS-CA}$; $\forall$ AS$_i$ $\overset{SK\_AS_i-CA}{\leftrightarrow}$ CA

(4) After receiving the message from the AS$_2$, the CA/RA validate ID$_1$ and ID$_2$, and then CA/RA $\rightarrow$ AS$_2$: $\{$AS$_1$, Tc$\}_{SK_AS-CA}$

(5) AS$_2$ $\rightarrow$ AS$_1$: $\{$ID$_1$, ReqNo, f$_1$(ID$_1$, ReqNo)$_{SK_1}\}_{SK\_AS_1-AS_2}$; $\forall$ AS$_i$ $\overset{SK\_AS_i-AS_j}{\leftrightarrow}$ AS$_j$; where i $\neq$ j

(6) First, the AS$_1$ computes f$_1$(ID$_1$, ReqNo)$_{SK_1}$, then AS$_1$ $\rightarrow$ MS$_1$: Td, Expirytime, f$_1$(Td, Expirytime, ReqNo)$_{SK_1}$

(7) The MS$_1$ computes f$_1$(Td, Expirytime, ReqNo)$_{SK_1}$ and compares it with the received one, then MS$_1$ $\rightarrow$ AS$_1$: $\{$Td, ReqNo$\}_{DK_1}$; MS$_1$ $\overset{DK_1}{\leftrightarrow}$ AS$_1$

(8) The AS$_1$ checks ReqNo and #Td then AS$_1$ $\rightarrow$ AS$_2$: $\{$ReqNo, Expirytime, f$_2$(Td, ReqNo)$_{SK_1}\}_{SK\_AS_1-AS_2}$

(9) AS$_2$ $\rightarrow$ MS$_2$: $\{$Te, ReqNo, Expirytime, $\{$f$_2$(Td, ReqNo)$_{SK_1}$ $\}_{SK\_AS_1-AS_2}$ $\}_{SK_2}$

(10) MS$_2$ $\rightarrow$ AS$_2$: $\{$Te$\}_{SK_2}$, and checks #Te with the received #Te

(11) MS$_2$ $\rightarrow$ MS$_1$: $\{$Ta, ReqNo$\}_{DK_1}$, if MS$_1$ finds correct #Ta and ReqNo, then the authentication is successful.

Phase-2: (1) MS$_1$ $\rightarrow$ MS$_2$: $\{$T$_i$, ID$_1\}_{DK_1}$; On receiving the message the MS$_2$ checks validity of ID$_1$ and T$_i$ $<=$ ExpTime.

(2) MS$_2$ $\rightarrow$ MS$_1$: $\{$T$_i\}_{DK_1}$; If received T$_i$ is same as was sent, then authentication is completed.

4. *Message Meaning Rule:*

(1) $\dfrac{MS_1|\equiv(MS_1\overset{DK_1}{\leftrightarrow}MS_2)\wedge(MS_1\overset{SK_1}{\leftrightarrow}AS_1)\wedge(MS_2\overset{SK_2}{\leftrightarrow}AS_2),AS_2\triangleleft f_1(ID_2,Tb,f_1(ID_1,ReqNo)_{SK_1})_{SK_2}}{MS_2|\equiv AS_2\ulcorner f_1(ID_2,Tb,f_1(ID_1,ReqNo)_{SK_1})_{SK_2}}$

(2) $\dfrac{AS_1|\equiv f_2(Td,ReqNo)_{SK_1}\wedge(AS_1\overset{SK_1}{\leftrightarrow}MS_1),MS_1\triangleleft f_1(Td,ReqNo,Expirytime)_{SK_1}}{AS_1|\equiv MS_1\ulcorner f_1(Td,ReqNo,Expirytime)_{SK_1}}$

5. *Nonce/Timestamp Verification Rule:*

(1) $\dfrac{MS_1|\equiv\#(Ta)\wedge MS_2\#(Tb),MS_2|\equiv AS_2\ulcorner f_1(ID_2,Tb,f_1(ID_1,ReqNo)_{SK_1})_{SK_2}}{MS_2|\equiv AS_2|\equiv f_1(ID_2,Tb,f_1(ID_1,ReqNo)_{SK_1})_{SK_2}}$

(2) $\dfrac{AS_2|\equiv(\#(Tc)\wedge\#(Te))\wedge AS_1|\equiv\#(Td),AS_1|\equiv MS_1\ulcorner f_1(Td,ReqNo,Expirytime)_{SK_1}}{AS_!|\equiv MS_1|\equiv f_1(Td,ReqNo,Expirytime)_{SK_1}}$

6. *Jurisdiction Rule:*

(1) $\dfrac{MS_2|\equiv AS_2\Rightarrow f_1(ID_2,Tb,f_1(ID_1,ReqNo)_{SK_1})_{SK_2},MS_2\triangleleft AS_2\ulcorner f_1(ID_2,Tb,f_1(ID_1,ReqNo)_{SK_1})_{SK_2}}{MS_1|\equiv MS_2|\equiv AS_2|\equiv AS_1}$

(2) $\dfrac{AS_1|\equiv MS_1\Rightarrow f_1(Td,ReqNo,Expirytime)_{SK_1},AS_1\triangleleft MS_1\ulcorner f_1(Td,ReqNo,Expirytime)_{SK_1}}{(AS_1|\equiv MS_1)\wedge(AS_2|\equiv MS_2)|\equiv AS_2|\equiv MS_1}$

7. *Protocol Goals:*

a) Mutual authentication between the MS and the AS:

$\mathrm{MS}_2 \mid\equiv \mathrm{AS}_2 \wedge \mathrm{AS}_1 \mid\equiv \mathrm{MS}_1 \to \mathrm{MS}_1 \mid\equiv \mathrm{MS}_2 \equiv \mathrm{AS}_2 \mid\equiv \mathrm{AS}_1$; Thus, mutual authentication is hold.

b) Efficient key management between the sender and receiver MS:

A DK_1 key is used between the MS_1 and the MS_2 to provide agreement. $\mathrm{AS}_1 \mid\equiv \#(\mathrm{Td})$, $\mathrm{MS}_1 \mid\equiv \mathrm{DK}_1 \wedge \#(\mathrm{Td})$, since $\mathrm{DK}_1=\mathrm{f}_2(\mathrm{Td},\ \mathrm{ReqNo})_{SK_1}$; $\mathrm{AS}_2 \mid\equiv \#(\mathrm{Te})$, $\mathrm{MS}_2 \mid\equiv \mathrm{SK}_2 \wedge \#(\mathrm{Te})$, and $(\mathrm{AS}_1 \to \mathrm{AS}_2) \wedge \mathrm{AS}_2 \to \mathrm{MS}_2) \mid\tilde{}\,\mathrm{DK}_1$

c) Key freshness between the MS and the AS:

$\mathrm{AS}_1 \mid\equiv \#(\mathrm{Td}) \wedge \mathrm{MS}_1 \mid\equiv \mathrm{D}\#(\mathrm{Td})$, $\mathrm{AS}_2 \mid\equiv \#(\mathrm{Te}) \wedge \mathrm{MS}_2 \mid\equiv \mathrm{D}\#(\mathrm{Te})$, since $\mathrm{DK}_1=\mathrm{f}_2(\mathrm{Td},\ \mathrm{ReqNo})_{SK_1}$

d) Confidentiality between the end-to-end MS via AS:

$$\frac{MS_1|\equiv(MS_1\overset{DK_1}{\leftrightarrow}MS_2),MS_2\triangleleft(Msg)_{DK_1}}{MS_1|\equiv MS_2\ulcorner Msg}\wedge\frac{MS_2|\equiv(MS_2\overset{DK_1}{\leftrightarrow}MS_1),MS_1\triangleleft(Msg)_{DK_1}}{MS_2|\equiv MS_1\ulcorner Msg}$$

e) Resistance replay attack:

If the attacker gets #(Ta) from message (1) and #(Tb) from message (2), he/she is unable to forge the message because he/she does not know SK_1 and SK_2. If the attacker gets #(Td) from message (6) and #(Te) from message (9), he/she is unable to forge the message because he/she does not know DK_1 and SK_2. Since #Ta, #Tb, #Td and #Te will be changed next time, thus, it defeats the attack.

f) Resistance man-in-the-middle attack:

Since, the attacker knows neither DK_1 nor $\{\}_{DK_1}$ encryption algorithm, hence, it prevents the communication from being eavesdropped.

g) Resistance SMS disclosure and OTA attack:

The MAES algorithm is proposed to use as $\{\}_{DK_1}$, which prevents SMS disclosure attack. End-to-end security of message between both the MS is provided by MAES with DK$_1$ key.

h) Resistance impersonation attack:

1) Adversary tries to impersonate the MS: Since, f$_1$(ID$_2$, Tb, f$_1$(ID$_1$, ReqNo)$_{SK_1}$)$_{SK_2}$ and f$_1$(ID$_1$, ReqNo)$_{SK_1}$ are computed at MS$_2$ and MS$_1$, and are compared at AS$_2$ and AS$_1$ respectively. This prevents the MS from the impersonation attack.

2) Adversary tries to impersonate the AS: The integrity value of f$_1$(ID$_2$, Tb, f$_1$(ID$_1$, ReqNo)$_{SK_1}$)$_{SK_2}$ at MS$_2$ and at AS$_2$ will be violated. Additionally, if the MS$_1$ receives f$_1$(Td, ReqNo, Expirytime)$_{SK_1}$ at any time, then the connection will be terminated because the MS$_1$ had not sent any request.

8.8 Chapter Summary

We have designed, the EasySMS protocol in order to provide end-to-end secure SMS communication between mobile users. This protocol provides mutual authentication between the user and the network. The analysis of proposed protocol shows that the protocol is able to prevent replay attack, SMS disclosure, over the air modification, man-in-the-middle attack and impersonation attack. The transmission of symmetric key to the mobile user is efficiently managed by the proposed protocol. The protocol produces lesser communication and computation overheads than the SMSSec and PK-SIM protocols. On an average, EasySMS protocol reduces bandwidth consumption by 51% and 31% and is able to lower the message exchanged ratio by 62% and 45% during the authentication in comparison to SMSSec and PK-SIM protocols respectively, when number of authentication requests (n) = 10, 50, 100, 200, 500, and 1000. Further, an improved MAES algorithm has been proposed, which provides faster encryption and decryption execution time in comparison to the other existing algorithms like DES, TripleDES with 2 keys, TripleDES with 3 keys, AES, Twofish, RC2, RC6, and CAST6.

Chapter 9

Batch Verification Based AKA Protocol for End-to-End SMS Security for Mobile Users

9.1 Introduction

The existing SMS facility does not have a mechanism to transmit confidential information from one mobile user to another user or to a group of users. It is a great challenge to provide end-to-end SMS security for multiple recipients at the same time especially when the system deals with symmetric key cryptography. Developed such protocol can be used in various applications like urgent business meeting information, military services like transmission of secure information in the battlefield, current location to our friends or family members, share market information etc.

9.2 System, Security and Attack Models

This section presents a system model, communication security requirements, and attack model in order to design a batch oriented end-to-end secure protocol for SMS transmission.

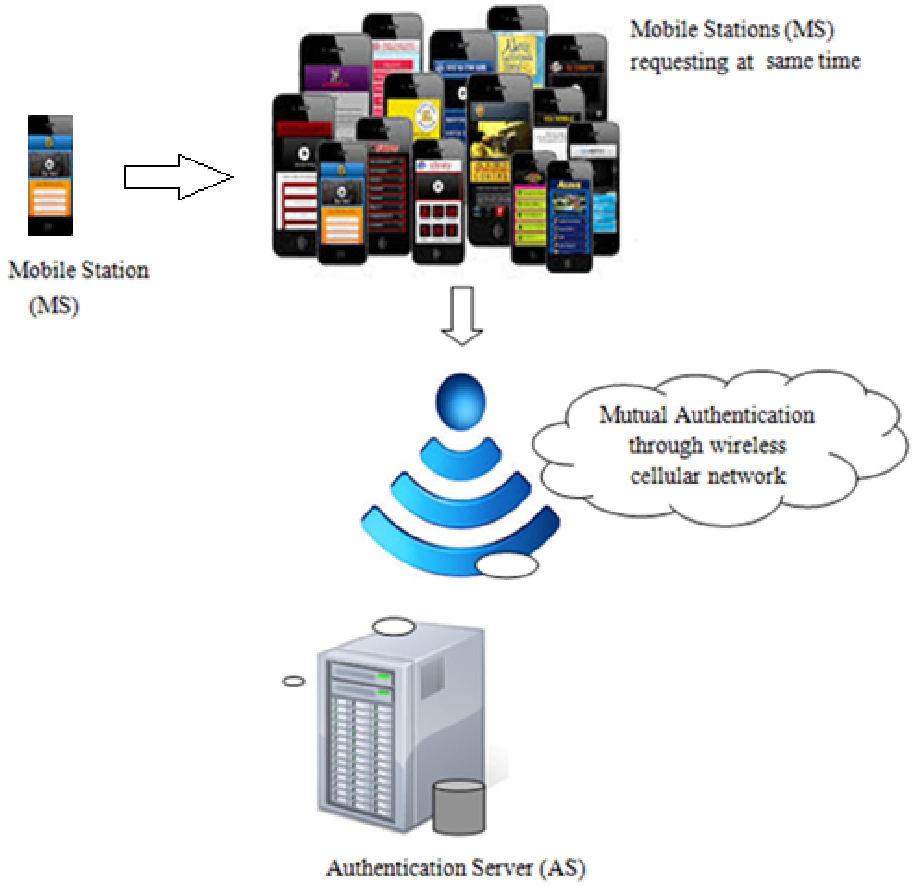


Figure 9.1: Batch verification-based end-to-end secure transmission of SMS

9.2.1 System Model

Consider a scenario where an MS sends the SMS to multiple MS simultaneously. Each MS, who receives the SMS, sends authentication request to the AS to verify the identity of the sender MS. The same scenario may execute for many other MS who wish to send SMS to the multiple recipients MS at the same time. The AS must be able to verify the maximum number of MS at one time based on its capacity. A scenario is demonstrated in Figure 9.1 where multiple MS send their authentication requests to the AS for the verification of sending MS′s identity at the same time or in a fixed (very less) duration time. The AS handles all the authentication requests and authenticates all the MS requested for the same. An efficient way is to perform a batch oriented authentication for multiple requests. But, it is required to verify whether there is any malicious request in the batch. If yes, then we need to identify it, remove it from the batch, and then again re-batch authentication is performed. One has to pay additional cost for every re-batch authentication. Various notations used in this chapter are presented

Table 9.1: Notations

Symbol	Definition	Size(Bits)
IMSI	International Mobile Subscriber Identity	128
TID	Temporary Identity	128
ReqNo	Request Number	128
SK	Shared secret key between MS and AS	128
DK	Delegation key generated from SK	128
MAC	Message Authentication Code	64
T	Timestamp	64
K	Random Number	128
Z	Signature generated by MS	128
Actcode	Activation Code of SK key	64
SIMcode	SIM Code of SK key	64
$f_1()$	Function used to generate DK	-
$f_2()$	Function used to generate TID	-
$f_3()$	Function used to generate MAC	-
$E\{\}_{DK}$	Encryption function with DK key	-
$D\{\}_{DK}$	Decryption function with DK key	-
$\oplus$	Bitwise-XOR operation	-
$\|\|$	Concatenation	-

in Table 9.1 along with their definitions and size. Here, the mobile operator′s server is an authentication server AS, which is basically an authentication center AuC in the traditional cellular network.

9.2.2 Security Requirements

A strong protocol must include basic security requirements for end-to-end SMS security. These requirements are mutual authentication, session key establishment, and privacy preservation, which have explained in section 7.2.2.

9.2.3 Attack Model

When an SMS is sent from one MS to another MS, it follows the path as MS→ BTS→ BSC→ MSC→ SMSC→ SMS-Gateway→ MSC→ BSC→ BTS→ MS. It is required that the proposed protocol must be able to provide prevention from SMS disclosure attack, SMS spoofing attack, OTA interface attack, cryptanalysis attack of generated cryptographic keys, replay attack, man-in-the-middle attack, and unencrypted SS7 network.

9.3 Proposed Protocol: BOPSMS

This section proposes an efficient protocol for SMS security over the network. Basically, the development of this protocol is divided into two parts. The first part focuses on the performance analysis of the authentication server to efficiently handle multiple authentication requests received at one time from each MS, and second part provides the illustration of complete end-to-end protocol for secure SMS along with first part. The first part is shown in Figure 9.2, while the complete BOPSMS protocol can be understood by Figure 9.3. The AS is always secure from any personal access by end user/mobile operator/intruder and is considered same as presently exists in the traditional cellular networks. Thus, it is almost impossible to extract the secret key SK. The storage scenario of SK key presented is same as nowadays used for voice communication in traditional cellular network. One should not confuse the AuC with the SMSC where the mobile operator can easily access the content of message. The AuC is secure from any personal access, and the keys stored at AuC can be accessed only by the execution of authentication protocol.

9.3.1 Focus on Efficient AS

In this subsection, an efficient approach for the AS is presented, which provides mutual authentication between the AS and each MS_i, similar to presented for value added services in [119], however, the generation of temporary identity (TID)/IMSI is different. Let m be the total number of authentication requests generated by various MS_i who receives the SMS sent by sender MS at any point of time. The Actcode is the one time activation code that is sent to the AS, and has explained in section 7.3.2 (*at MS:* $\mathrm{Actcode} = \mathrm{LCS}_n(\mathrm{T}_1 \oplus \mathrm{LAI} \oplus \mathrm{SIMcode})$ and *at AS:* $\mathrm{SIMcode} = \mathrm{RCS}_n(\mathrm{T}_1 \oplus \mathrm{LAI} \oplus \mathrm{Actcode})$). Similar to EXTVAS-AKA explained in section 7.3.2, each MS chooses a random number k_i ($1 < \mathrm{k}_i < \mathrm{m}$), generates current timestamp T_i and DK_i, where $\mathrm{DK}_i = \mathrm{f}_1(\mathrm{T}_i)_{SK_i}$. Then, every MS_i generates $\mathrm{Y}_i = \mathrm{k}_i \oplus \mathrm{IMSI}_i$ and signature as $\mathrm{Z}_i = (\mathrm{k}_i + \mathrm{DK}_i \oplus \mathrm{ReqNo}) \bmod \mathrm{m}$. Each MS_i also computes $\mathrm{Actcode}_i$ and $\mathrm{TID}_i = \mathrm{f}_2(\mathrm{IMSI}_i, \mathrm{T}_i)_{DK_i}$ to protect the original IMSI_i over the network. This defeats user-id theft and man-in-the-middle attacks.

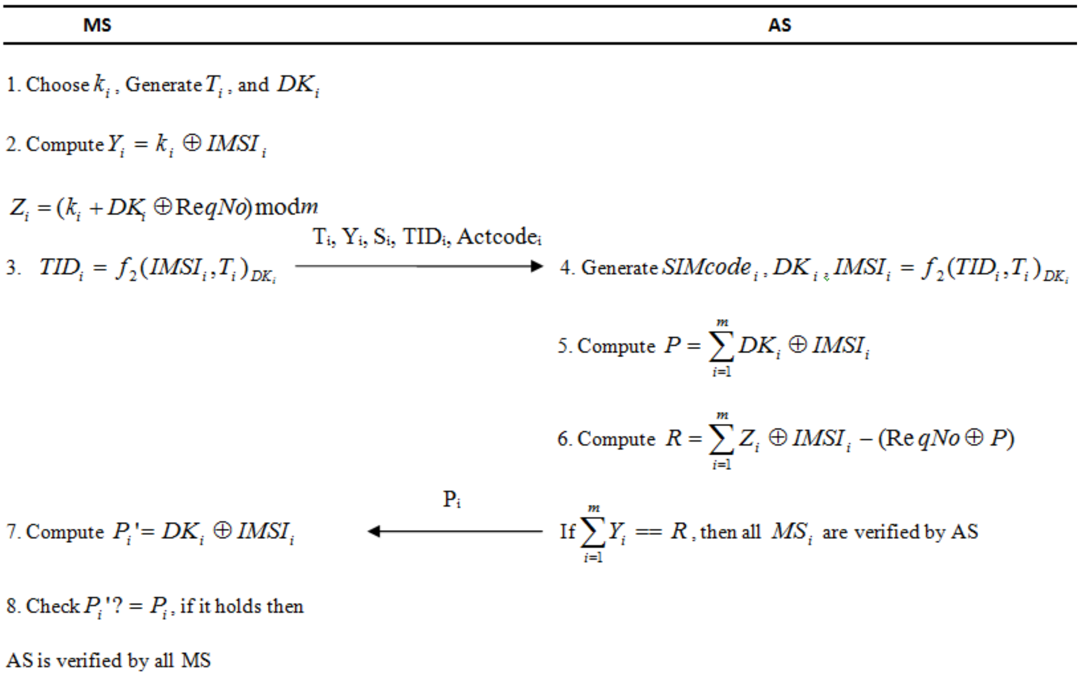

MS
AS
1. Choose k_i, Generate T_i, and DK_i
2. Compute $Y_i = k_i \oplus IMSI_i$
$Z_i = (k_i + DK_i \oplus ReqNo) \bmod m$
3. $TID_i = f_2(IMSI_i, T_i)_{DK_i}$
Tᵢ, Yᵢ, Sᵢ, TIDᵢ, Actcodeᵢ
4. Generate $SIMcode_i$, DK_i, $IMSI_i = f_2(TID_i, T_i)_{DK_i}$
5. Compute $P = \sum_{i=1}^{m} DK_i \oplus IMSI_i$
6. Compute $R = \sum_{i=1}^{m} Z_i \oplus IMSI_i - (ReqNo \oplus P)$
Pᵢ
7. Compute $P_i' = DK_i \oplus IMSI_i$
If $\sum_{i=1}^{m} Y_i == R$, then all MS_i are verified by AS
8. Check $P_i' ?= P_i$, if it holds then
AS is verified by all MS

Figure 9.2: Proposed approach for the AS

Now, each MS$_i$ sends an authentication request as (T$_i$, Y$_i$, Z$_i$, TID$_i$, Actcode$_i$) to the AS. On receiving the authentication requests, the AS computes SIMcode$_i$, DK$_i$, and IMSI$_i$ = f$_2$(TID$_i$, T$_i$)$_{SK_i}$ with the help of SK$_i$. Next, the AS computes P = $\sum_{i=1}^{m}$ (DK$_i$⊕IMSI$_i$) and R = $\sum_{i=1}^{m}$ Z$_i$⊕IMSI$_i$ - (ReqNo⊕P). If $\sum_{i=1}^{m}$Y$_i$ == R, then all the MS$_i$ are successfully verified by the AS otherwise, one or more MS$_i$ are malicious or invalid, which requires re-batch authentication. Here, ReqNo is the request number for authentication. In the re-batch authentication process, first the AS finds malicious MS$_i$ by $Malicious\_Req\_Algorithm(AR)$ algorithm (explained in section 7.4.2). The AS removes the malicious MS$_i$ from the batch and again computes P = $\sum_{i=1}^{m-t}$ (DK$_i$⊕IMSI$_i$) and R = $\sum_{i=1}^{m-t}$ Z$_i$⊕IMSI$_i$ - (ReqNo⊕P), where t is the total number of invalid MS$_i$. Then the AS compares $\sum_{i=1}^{m-t}$Y$_i$ == R. If it holds, then all the MS$_i$ are authenticated by the AS otherwise, repeat the re-batch authentication. Finally, the AS sends all P$_i$ to the respective MS$_i$. All the MS$_i$ compute P$'_i$ as P$'_i$ = DK$_i$ ⊕ IMSI$_i$ and compare it with the received P$_i$. If both are same, then the AS is verified by all the MS$_i$ otherwise, the particular MS$_i$ terminates the connection and resends the authentication request to the AS.

9.3.2 BOPSMS Protocol for End-to-End SMS Security

This subsection proposes a complete batch verification-based protocol to provide secure SMS facility, which is shown in Figure 9.3 and is divided into two phases as follows:

Phase-1: Initially, the sender MS generates T_1, ReqNo, $Actcode_1$, $TID_1 = f_2(IMSI_1, T_1)_{DK_1}$ and $MAC1_1 = f_3(T_1, ReqNo, TID_1, Actcode_1)$, and sends to all targeted MS_i (message (1)). Each MS_i chooses a random number k_i ($1< k_i <m$), generates timestamp T_i and $DK_i = f_1(T_i)_{SK_i}$. Every MS_i computes $Y_i = k_i \oplus IMSI_i$ and signature as $Z_i = (k_i + DK_i \oplus ReqNo) mod\ m$. Each MS_i also computes $Actcode_i$ and $TID_i = f_2(IMSI_i, T_i)_{DK_i}$ and $MAC2_i = f_3(T_1, ReqNo, TID_1, Actcode_1, MAC1_1 , T_i, Y_i, Z_i, TID_i, Actcode_i)$ to prevent the transmission of original $IMSI_i$ over the network, and sends (T_1, ReqNo, TID_1, $Actcode_1$, $MAC1_1$, T_i, Y_i, Z_i, TID_i, $Actcode_i$, $MAC2_i$) to the AS (message (2)). Here, TID, IMSI, ReqNo, Y, Z, SK, DK, P, R, and Q are of 128 bits, whereas timestamp T and expiry time ExpT are of 64 bits in size. On receiving the authentication requests, the AS first computes $MAC1'_1$ and $MAC2'_i$, and compares them with the received $MAC1_1$ and $MAC2_i$. If both are same, then only the AS proceeds for next step otherwise, the connection is terminated for each invalid result. The, the AS computes all $SIMcode_i$, DK_i, and $IMSI_i$ for valid results. Thereafter, the AS computes $P = \sum_{i=1}^{m} (DK_i \oplus IMSI_i)$ and $R = \sum_{i=1}^{m} Z_i \oplus IMSI_i$ - $(ReqNo \oplus P)$. If $Q = (\sum_{i=1}^{m} Y_i) == R$, then all the MS_i are successfully verified by the AS otherwise, one or more MS_i are malicious or invalid, which then requires re-batch authentication, where the AS finds the malicious MS_i with a detection algorithm and removes the malicious MS_i from the batch.

Afterwards, the AS sends $\{T_{m+1}, ReqNo, ExpT\}_{DK_1}$ to the MS (message (3)). The MS replies back to the AS with $\{T_{m+1}\}_{DK_1}$ (message (4)). Thereafter, the AS sends all P_i to the respective MS_i along with $MAC3_i = f_3(P_i)$ and $\{T_{m+2}, ReqNo, ExpT, DK_1\}_{DK_i}$ (message (5)). All the MS_i first compute and compare $MAC3_i$?= $MAC3'_i$. If it holds for all MS_i, then compute $P'_i = (DK_i \oplus IMSI_i)$ and compare it with the received P_i. If both are equal, then the AS is verified by all the MS_i otherwise, the particular MS_i terminates the connection. Finally, the AS sends $\{T_1, ReqNo\}_{DK_1}$ to the MS (message (6)), where T_1 (first timestamp) shows the completion of authentication process.

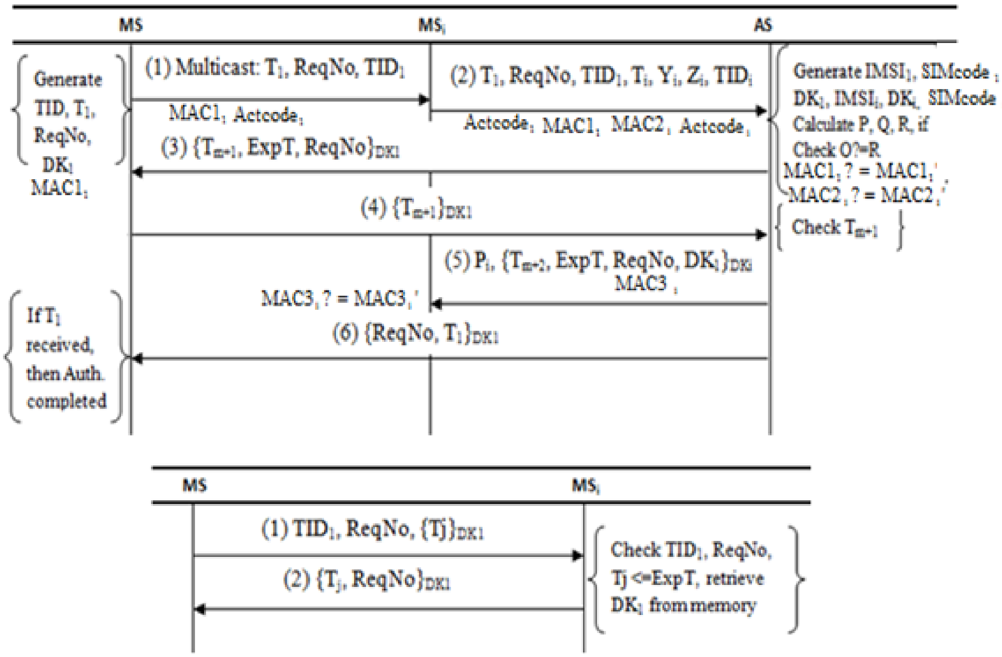

MS
MSi
AS
Generate TID, T1, ReqNo, DK1 MAC1i
(1) Multicast: T1, ReqNo, TID1
MAC1i Actcodei
(2) T1, ReqNo, TID1, Ti, Yi, Zi, TIDi
Actcodei MAC1i MAC2i Actcodei
Generate IMSI1, SIMcode1 DK1, IMSIi, DKi, SIMcodei Calculate P, Q, R, if Check O?=R
MAC1i ? = MAC1i' MAC2i ? = MAC2i'
(3) {Tm+1, ExpT, ReqNo}DK1
(4) {Tm+1}DK1
Check Tm+1
(5) Pi, {Tm+2, ExpT, ReqNo, DKi}DKi MAC3i
MAC3i ? = MAC3i'
If T1 received, then Auth. completed
(6) {ReqNo, T1}DK1
MS
MSi
(1) TID1, ReqNo, {Tj}DK1
Check TID1, ReqNo, Tj <=ExpT, retrieve DK1 from memory
(2) {Tj, ReqNo}DK1

Figure 9.3: BOPSMS protocol (a) phase-1 (b) phase-2

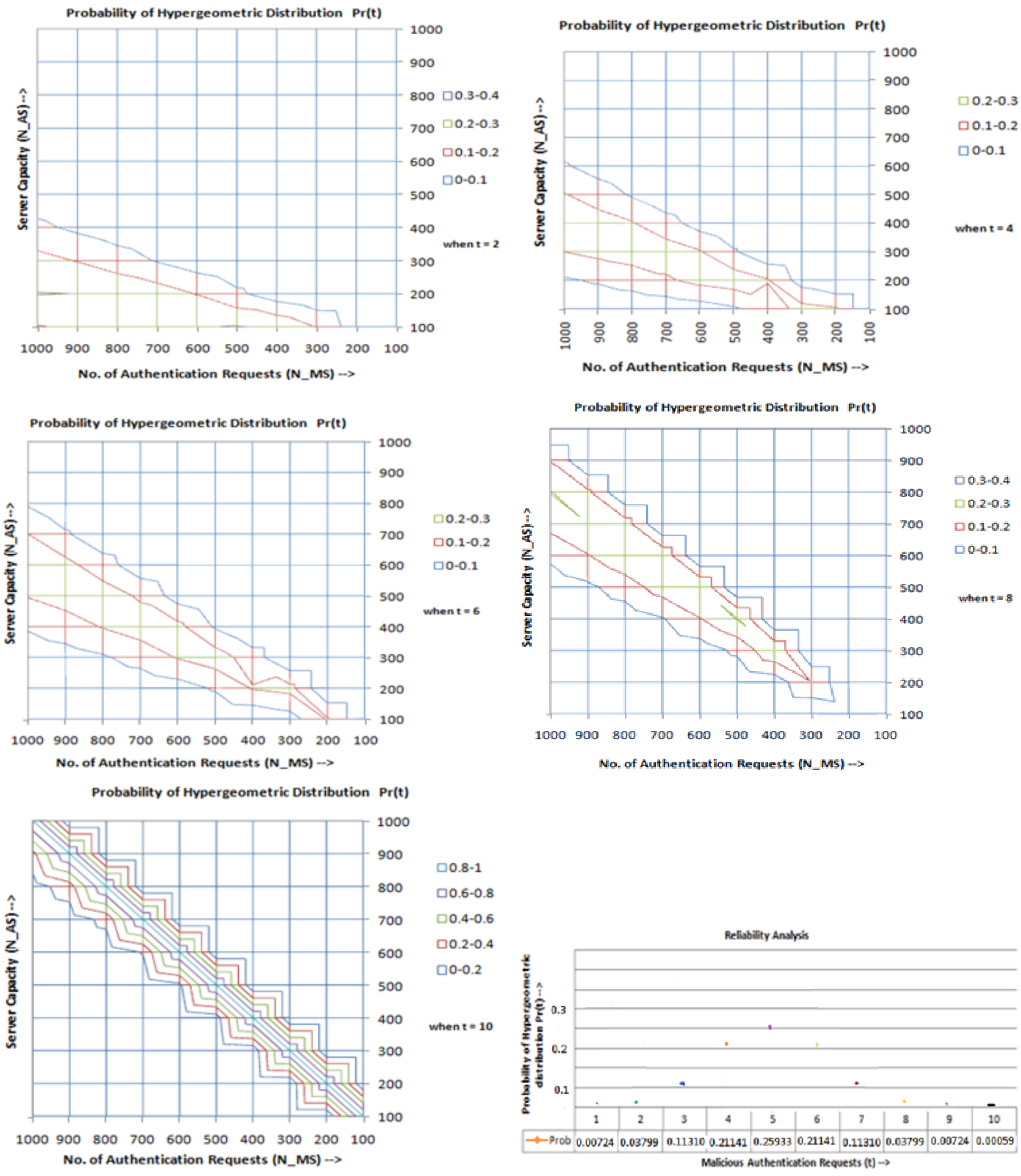
Phase-2: For each subsequent authentication within expiry time of DK$_1$ key, the MS sends TID$_1$, ReqNo, $\{T_j\}_{DK_1}$ to respective MS$_i$ (message (1)). On receiving the message, all the MS$_i$ check (ReqNo, ExpT) and respond back to the MS with {ReqNo, $T_j\}_{DK_1}$ (message (2)).

9.4 Discussion

The reliability analysis of BOPSMS protocol and the algorithm to detect malicious requests are explained in section 7.4 (chapter 7). Figure 9.4 shows the reliability analysis of proposed protocol in terms of Hypergeometric distribution probability, when 100 $\leq$ N$_{MS}$ 1000, 100 $\leq$ N$_{AS}$, N$_{IN}$ = 10, and (a) t=2 (b) t=4 (c) t=6 (d) t=8 (e) t=10. The Figure 9.4(f) represents the probability of Hypergeometric distribution, when N$_{MS}$ = 1000, N$_{AS}$ = 900, N$_{IN}$ = 10, and malicious authentication requests t = 1, 2, 3,...,10. This probability is maximum (0.25) for t = 5, and minimum (0.00059) for t = 10.

9.5 Security Analysis of BOPSMS Protocol

This section provides the security analysis of the BOPSMS protocol in terms of mutual authentication, session key establishment, and privacy preservation.



	1	2	3	4	5	6	7	8	9	10
Prob	0.00724	0.03799	0.11310	0.21141	0.25933	0.21141	0.11310	0.03799	0.00724	0.00059

Figure 9.4: Reliability analysis (left to right, top to bottom) (a) t=2 (b) t=4 (c) t=6 (d) t=8 (e) t=10 (f) t=1, 2,...,10

9.5.1 Mutual Authentication

The BOPSMS protocol provides mutual authentication between the AS & the MS, and the AS & the MS$_i$. The AS authenticates all the MS$_i$ by checking $\sum_{i=1}^{m}$ Y$_i$ == R, and each MS$_i$ authenticates the AS by comparing P$'_i$?= P$_i$. The MS (sender) authenticates the AS with DK$_1$, and the MS is authenticated by the AS with the received message including T$_{m+1}$.

9.5.2 Session Key Establishment

The DK$_i$ key is used as a session key for each authentication between the AS and the MS$_i$. This key is used for a session within the expiry time.

9.5.3 Privacy Preservation

The privacy of each MS$_i$ is well protected during the authentication process. The TID$_i$ is computed from the original IMSI$_i$ as TID$_i$ = f$_2$(IMSI$_i$, T$_i$)$_{DK_i}$.

9.5.4 Resistance from Various Attacks

1. *SMS Disclosure:* The encryption approach ($\{\}_{DK_1}$ = AES) used in BOPSMS protocol helps in transmitting authentication information securely and prevents the system from SMS disclosure.

2. *SMS Spoofing:* The proposed protocol provides mutual authentication between all the MS$_i$ and the AS. When all the MS$_i$ and the AS authenticate each other then only the authentication is successfully completed. Hence, no malicious user or attacker can spoof the SMS.

3. *Replay Attack:* The BOPSMS protocol is free from this attack because it sends timestamp and request number along with the information over the network.

4. *Man-in-the-middle Attack:* In the BOPSMS protocol, a symmetric cryptographic algorithm ($\{\}_{DK_1}$ = AES) is used between the sender MS and all the recipients MS through the AS. Further, DK$_1$ and DK$_i$ keys are secret in nature. Thus, it prevents the communication from eavesdropping and MITM attack.

5. *OTA Protection:* The BOPSMS protocol provides end-to-end security from the sender MS to all the receivers including OTA interface with a strong encryption $\{\}_{DK_1}$ with DK$_1$ key, which is generated from SK key shared between the MS (sender) and the AS.

6. *Redirection Attack:* The integrity protection with message authentication codes MAC1$_1$, MAC2$_i$, and MAC3$_i$ protects the messages from redirection attack.

7. *Security Protection over the SS7 Channel:* The proposed protocol provides end-to-end security so that no attacker could gain the secret information from the message over the SS7 channel.

9.6 Performance Evaluation of BOPSMS Protocol

This section provides the performance evaluation of our proposed approach for the AS along with the proposed BOPSMS protocol.

9.6.1 Analysis of Proposed Approach for AS

We analyze the performance of the proposed approach for the AS (discussed in section 9.3.1) in terms of communication overhead and compare it with various other protocols of vehicular ad hoc networks like IBV [59], ECDSA-AKA [60], BLA [62], and ABAKA [58].

Table 9.2: Communication overhead in single authentication

Protocols	Device to Server (bytes)	Server to Device (bytes)
IBV	63	N/A
ABAKA	84	80
BLS	146	N/A
ECDSA-AKA	167	167
Our Approach for AS	56	16

Single Authentication: It can be clearly observed from Table 9.2 that the proposed approach for the AS generates lesser transmission overhead, *i.e.*, 64

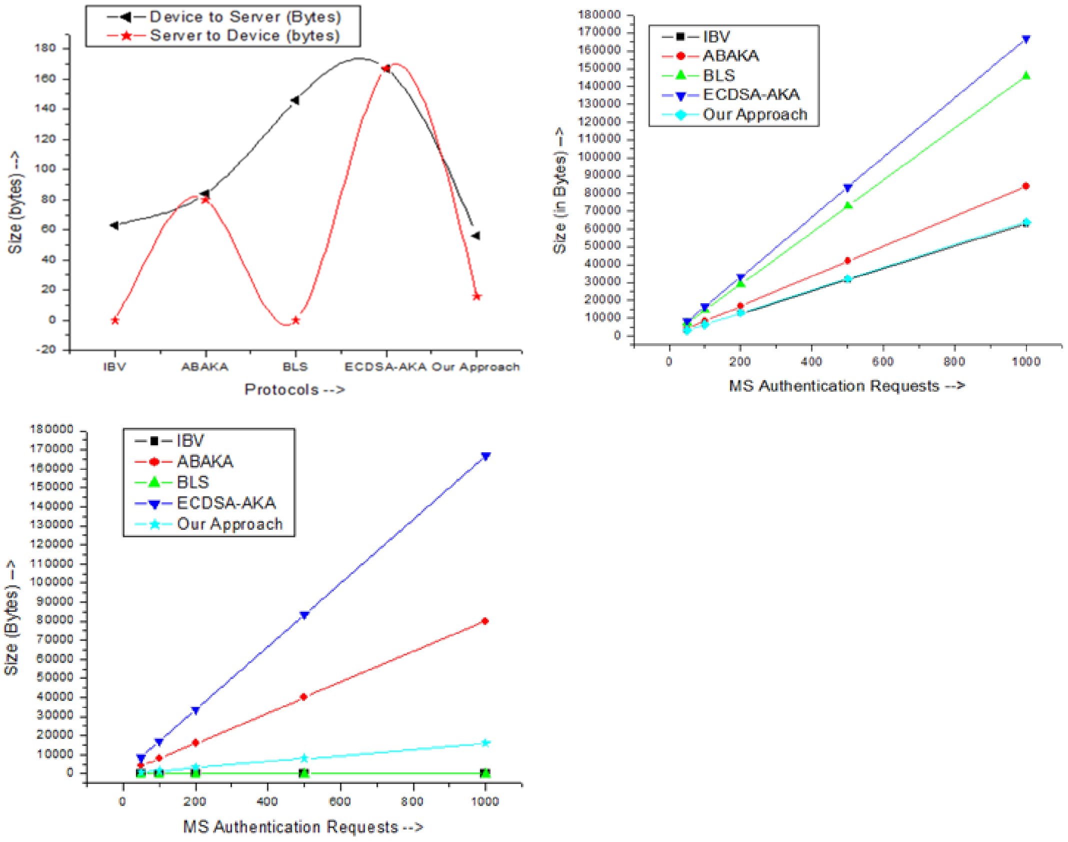


Figure 9.5: Total transmitted bytes in various protocols (left to right, top to bottom) (a) single authentication, (b) batch authentication: from the device to the server, (c) batch authentication: from the server to the device

Table 9.3: Communication overhead in batch authentication

Protocols	Device to Server (bytes)	Server to Device (bytes)
IBV	63m	N/A
ABAKA	84m	80m
BLS	146m	N/A
ECDSA-AKA	167m	167m
Our Approach for AS	56m	16m

and 16 bytes, from the device to the server and the server to the device respectively as compared to ABAKA, BLA, and ECDSA-AKA protocol. However, it is slightly large than the IBV protocol that produces 63 byes overhead from the device to the server.

Multiple Authentications: Table 9.3 represents the communication overhead generated by m-authentication requests, which is similar to m-times single authentication. Figure 9.5(a) represents two separate graphs for transmitting the information from the MS to the AS and from the AS to the MS. Figure 9.5(b) and Figure 9.5(c), each shows a graph for device to server and server to device multiple authentication and conclude that proposed approach for the AS generates lesser (except IBV) number of transmitted bytes, when m = 50, 100, 200, 500, and 1000. In Figure 9.5(c), the IBV and BLS protocols are considered with zero value from the server to the device authentication. From the device to the server, the batch oriented proposed approach is able to lower 23.81%, 56.17%, and 61.68% of the transmission bandwidth as compared to ABAKA, BLS, and ECDSA-AKA protocols respectively. Similarly, for the transmission of bytes from the server to the device, the proposed approach reduces the bandwidth by 80% and 90.5% in comparison to ABAKA and ECDSA-AKA protocols respectively.

9.6.2 Analysis of BOPSMS Protocol

This subsection analyzes the performance of BOPSMS protocol for end-to-end SMS security. Let m be the number of targeted/recipient MS and r be the number of subsequent multiple authentication requests within the expiry time, *i.e.*, ExpT.

1. *Communication Overhead:* The transmission overhead for the BOPSMS

protocol during the m-authentication requests can be calculated as:

Phase-1: Total number of transmitted bits = (1)+(2)+(3)+(4)+(5)+(6) = (128+ 64+128 +64+64)*m + (128+64+128+64+64+128+64+128+128+ 64+64)*m+(64+64+128)+(64)+((128+64)*m+(64+128+64+128)*m) + (128+64) = 512+2048*m

Phase-2: Total number of transmitted bits = ((7)+(8))*r = (128+128+ 64)*r+ (64+128)*r = 512*r

Total overhead = 512+(2048*m)+(512*r) bits

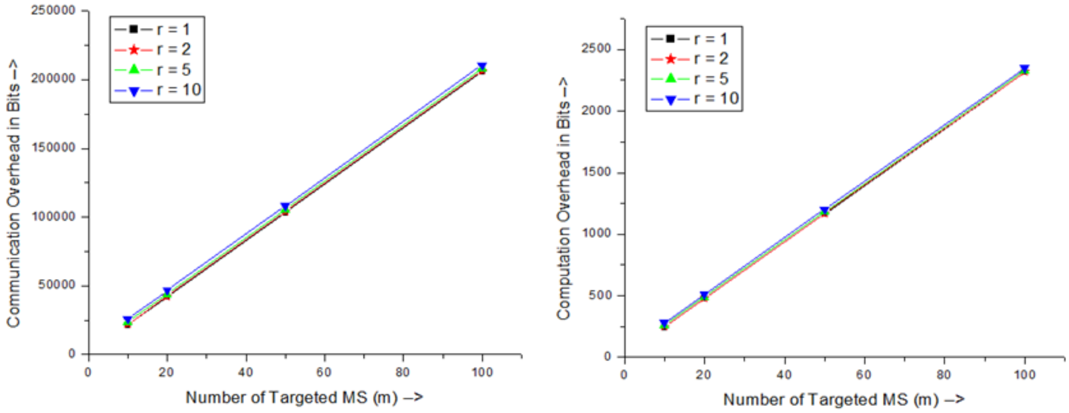


Figure 9.6: BOPSMS protocol (a) communication overhead (b) computation overhead

2. *Computation Overhead:* The computation overhead generated by BOPSMS during m-authentication requests is computed as below:

   Phase-1: Computation at MS: $f_1()$, $f_2()$, 2*D$\{\}_{DK_1}$, E$\{\}_{DK_1}$, $f_3()$, Actcode$_1$

   Computation at MS$_i$: m*$f_1()$, m*$f_2()$, m*(DK$_i$ $\oplus$ IMSI$_i$), m*(k$_i$ $\oplus$ IMSI$_i$), W$_i$ = m*(DK$_i$ $\oplus$ ReqNo), Z$_i$ = m*(k$_i$ $\oplus$ W$_i$), m*D$\{\}_{DK_1}$, m*Actcode$_1$, 2m*$f_3()$

   Computation at AS: (m+1)*$f_1()$, (m+1)*$f_2()$, P$_i$ = m*(DK$_i$ $\oplus$ IMSI$_i$), P = (m-1)*$\sum_{i=1}^{m}$ P$_i$, Z$'_i$ = m*(Z$_i$ $\oplus$ IMSI$_i$), R$'$ = (ReqNo $\oplus$ P), Z$'$ = (m-1)*$\sum_{i=1}^{m}$ Z$'_i$, (Z$'$ - R$'$), Q = (m-1)*$\sum_{i=1}^{m}$ Y$_i$, D$\{\}_{DK_1}$, (m+2)*E$\{\}_{DK_1}$, 3m*$f_3()$, 2m*SIMcode$_1$

   Phase-2: Computation at MS: r*E$\{\}_{DK_1}$, r*D$\{\}_{DK_1}$; *at MS$_i$:* r*D$\{\}_{DK_1}$, r*E$\{\}_{DK_1}$

   Now, all functions are considered with a single unit cost. The computations at MS, MS$_i$, and AS are as follows:

   Phase-1: at MS: 7; *at MS$_i$:* 10*m; *at AS:* 3*(m-1)+(8*m+5)+2+(2*m)

= 13*m+4

Phase-2: at MS: 2*r; *at* MS_i*:* 2*r

Total overhead = 7+(10*m)+(13*m+4)+(2*r)+(2*r) = 11+(23*m)+(4*r) bits

Figure 9.6 shows the communication and computation overheads (in bits) when m = 10, 20, 50, 100; r = 1, 2, 5, 10.

9.7 Simulation of BOPSMS Protocol

This section presents the simulated results of BOPSMS protocol in Java environment and evaluates total execution time, batch verification delay, and re-batch verification delay of BOPSMS protocol.

9.7.1 Protocol Execution Time

This implementation involves a client-server paradigm, where MS is a client and AS is a server. On an average, the execution time to perform each operation, *i.e.*, addition, multiplication (bitwise XOR), and subtraction, is T_{add} = .000933 milliseconds, T_{mul} = .030322 milliseconds, and T_{sub} = .000933 milliseconds respectively. Total number of operations required in a batch authentication are: (2m+2)*$f_1()$, (2m+2)*$f_2()$, (5m+1)*$f_3()$, T_{sub}, (5m+1)*T_{mul}, (4m-3)*T_{add}, (m+2r+3)*$E\{\}_{DK_1}$, (m+2r+3)*$D\{\}_{DK_1}$, (m+1)*$Actcode_i$, 2m*$SIMcode_i$.

Table 9.4: Computation for $f_2()$ and $f_3()$ used in BOPSMS protocol

$f_2()$				$f_3()$		
Enc ExT	TUM	Dec ExT	TUM	ExT	PCPUT	TUM
42	9139681	15	9124165	221.60	296.40	15211840

Table 9.5: $f_1()$ and E{}/D{} used in BOPSMS protocol

$E\{\}_{DK}/D\{\}_{DK}$				$f_1()$		
$E\{\}_{DK}$ ExT	TUM	$D\{\}_{DK}$ ExT	TUM	ExT	PCPUT	TUM
8.8	9329.6	9.1	4816	273.41	296.40	15204024

In phase-1 of BOPSMS protocol, on an average, the connection establishment time between the MS/MS_i and the MS_i/AS is 3659 milliseconds, and the transmission time: for message (1) is 3.1 milliseconds, for (2) is 3.8 milliseconds,

Table 9.6: Actcode/SIMcode in BOPSMS protocol

Computation of Actcode/SIMcode function		
ExT	PCPUT	TUM
0.89	93.60	12968290

Table 9.7: Execution time for BOPSMS protocol (in seconds)

Authentication Requests (m)	r = 1	r = 2	r = 5	r = 10
10	57	59.2	65.8	76.8
20	110.9	113.1	119.7	130.7
50	272.6	274.8	281.4	292.4
100	542.1	544.3	550.9	561.9

for (3) is 4.6 milliseconds, for (4) is 3.0 milliseconds, for (5) is 4.8 milliseconds, and for (6) is 4.4 milliseconds. Similarly, for phase-2, on an average, the connection establishment time between the MS and the MS_i is 2161 milliseconds, and transmission time for message (1) is 4.6 milliseconds, and for message (2) is 6.1 milliseconds. This work has been simulated with 50 MS clients sending their authentication request messages to single AS and the average value of 30 iterations are considered for each result. Table 9.4 represents the results obtained for $\text{f}_2()$ with AES-CTR algorithm, where encryption (generation of TID_i) took 42 milliseconds and decryption (generation of IMSI_i) executed 15 milliseconds. In total operations of $\text{f}_2()$, half operations are used as encryption (to generate TID_i) and other half as decryption (to generate IMSI_i). As shown in Table 9.5, it has been found that AES (without mode) takes 8.8 milliseconds for $\text{E}\{\}_{DK_1}$ and 9.1 milliseconds for $\text{D}\{\}_{DK_1}$ on J2ME WTK platform (version 2.5.1). The $\text{f}_3()$ and $\text{f}_1()$ are implemented as HMACSHA1 and HMACSHA256 respectively. The results obtained for the same are shown in Table 9.4 and Table 9.5. The output of HMACSHA1 and HMACSHA256 are truncated to 64 and 128 bits respectively because the output of $\text{f}_3()$ is 64 bits MAC while the output of $\text{f}_1()$ is 128 bits, which is DK_i key. The input to HAMCSHA1 and HMACSHA256 are 512 bits (actual input size plus trailing zeros to make it multiple of 512). Table 9.6 shows the results obtained for Actcode/SIMcode, which is a function of XOR and took 0.89 milliseconds.

Total Operations Required by BOPSMS in Single Authentication: Please note that phase-2 will not execute for single authentication.

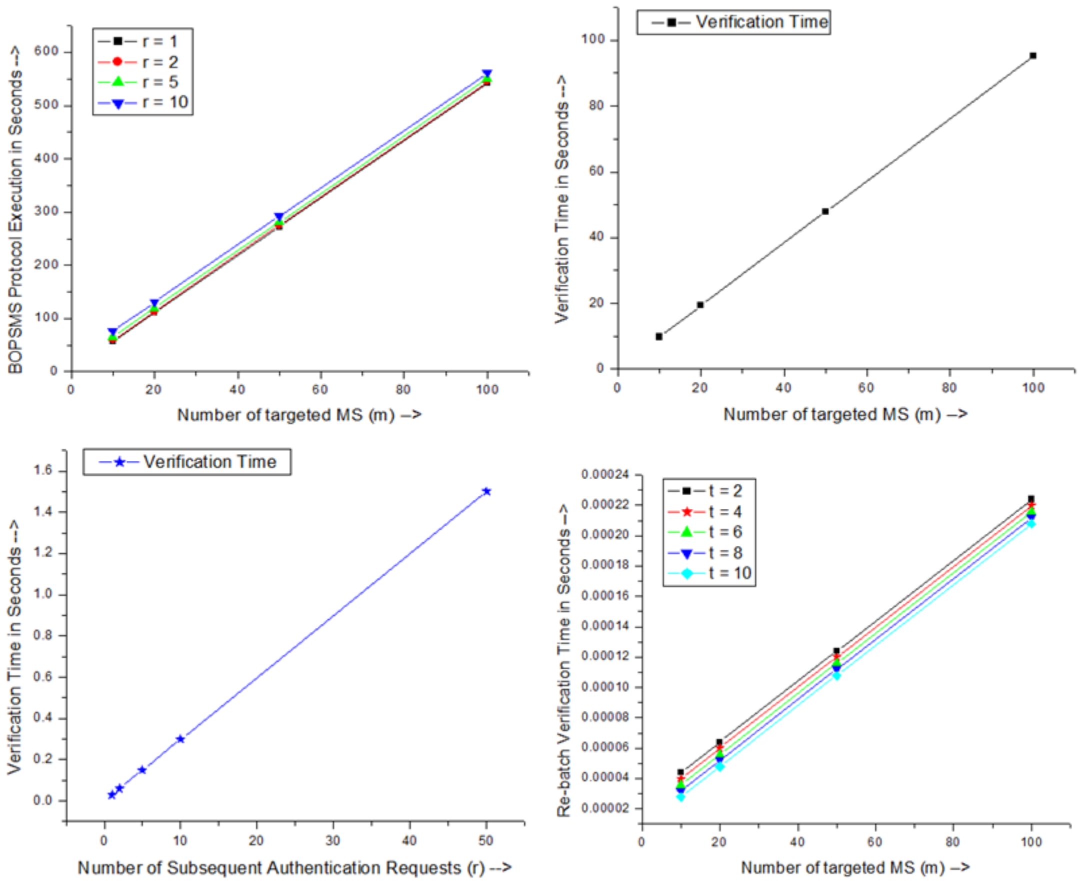


Figure 9.7: BOPSMS protocol (left to right, top to bottom) (a) execution time (b) verification time for phase-1 (c) verification time for phase-2 (d) re-batch verification time

at MS: $f_1()$, $f_2()$, $E\{\}_{DK_1}$, $2*D\{\}_{DK_1}$, $f_3()$, $Actcode_1$

at MS_i: $f_1()$, $f_2()$, $3*T_{mul}$, T_{add}, $D\{\}_{DK_1}$, $2*f_3()$, $Actcode_i$

at AS: $2*f_1()$, $2*f_2()$, $3*T_{mul}$, T_{sub}, $3*E\{\}_{DK_1}$, $D\{\}_{DK_1}$, $3*f_3()$, $2*SIMcode_i$

Total execution time for BOPSMS in single authentication = server connection establishment time phase-1 + transmission time for all messages phase-1 + time at MS + time at MS_i + time at AS = 3659 + (3.1 + 3.8 + 4.6 + 3.0 + 4.8 + 4.4) + (273.41 + 42 + 8.8 + 2*9.1 + 221.60 +0.89) + (273.41 + 42 + 3*0.030322 + 0.000933 + 9.1 + 2*221.60 +0.89) + (2*273.41 + 2*15 + 3*0.030322 + 0.000933 + 3*8.8 + 9.1 + 3*221.60 + 2*0.89) = 6295.28 milliseconds = 6.30 seconds

Total Operations Required by BOPSMS in Multiple/Batch Authentication:

at MS: $f_1()$, $f_2()$, $(1+r)*E\{\}_{DK_1}$, $(2+r)*D\{\}_{DK_1}$, $f_3()$, $Actcode_1$

at MS_i: $m*f_1()$, $m*f_2()$, $3*T_{mul}$, T_{add}, $r*E\{\}_{DK_1}$, $(m+r)*D\{\}_{DK_1}$, $2m*f_3()$, $m*Actcode_i$

at AS: $(1+m)*f_1()$, $(1+m)*f_2()$, $(2m+1)*T_{mul}$, $(3m-3)*T_{add}$, T_{sub}, $3*E\{\}_{DK_1}$, $D\{\}_{DK_1}$, $3m*f_3()$, $2m*Actcode_i$

In phase-1, the frequency of messages (1), (2), and (5) are based on the number of authentication requests, *i.e.*, m, while messages (3), (4), and (6) are single messages.

Total execution time for BOPSMS in batch authentication = server connection establishment time phase-1 + transmission time for all messages phase-1 + server connection establishment time phase-2 + transmission time for all messages phase-2 + time at MS + time at MS_i + time at AS = 3659*m + ((3.1 + 3.8 + 4.8)*m + 4.6 + 3.0 + 4.4) + 2161*r + (4.6 + 6.1)*r + (273.41 + 42 + (1+r)*8.8 + (2+r)*9.1 + 221.60+ 0.89) + (m*273.41 + m*42 + 3*0.030322 + 0.000933 + r*8.8 + (m+r)*9.1 + 2m*221.60 + 0.89*m) + ((1+m)*273.41 + (1+m)*15 + (2m+1)*0.030322 + (3m-3)*0.000933 + 0.000933 + 3*8.8 + 9.1 + 3m*221.60 + 2m*0.89) = 900.04+ 2207.5*r+5394.35*m milliseconds = 0.9+2.20*r+5.39*m seconds

Table 9.7 shows execution time (in seconds) for BOPSMS protocol, when m = 10, 20, 50, 100, and subsequent authentication requests r = 1, 2, 5, 10. It is clear that execution time reduces when m is increased.

9.7.2 Verification Delay

The verification delay for BOPSMS protocol is observed in this subsection.

BOPSMS Phase-1: Operations to verify the MS$_i$ by the AS = m*f$_1$(), m*f$_2$(), m*(DK$_i$ $\oplus$ IMSI$_i$), P = (m-1)*$\sum_{i=1}^{m}$ P$_i$, Z$'_i$ = m*(Z$_i$ $\oplus$ IMSI$_i$), Z$'$ = (m-1)*$\sum_{i=1}^{m}$ Z$'_i$, R$'$ = (ReqNo $\oplus$ P), (Z$'$ - R$'$), Q = (m-1)*$\sum_{i=1}^{m}$ Y$_i$, m*f$_3$(), m*SIMcode$_i$

Operations to verify the MS by the AS = f$_1$(), f$_2$(), E$\{\}_{DK_1}$, D$\{\}_{DK_1}$, m*f$_3$(), m*SIMcode$_i$

Operations to verify the AS by the MS$_i$ = P$_i$ = m*(DK$_i$ $\oplus$ IMSI$_i$), m*f$_3$()

Operations to verify the AS by the MS = D$\{\}_{DK_1}$

It is considered that all functions are of single unit cost. Then,

Operations to verify the MS$_i$ by the AS = m*f$_1$(), m*f$_2$(), (2m+1)*T$_{mul}$, (3m-3)*T$_{add}$, T$_{sub}$, m*f$_3$(), m*SIMcode$_i$

Operations to verify the MS by the AS = f$_1$(), f$_2$(), E$\{\}_{DK_1}$, D$\{\}_{DK_1}$, m*f$_3$(), m*SIMcode$_i$

Operations to verify the AS by the MS$_i$ = m*T$_{mul}$, m*f$_3$()

Operations to Verify the AS by the MS = D$\{\}_{DK_1}$

Time to verify the MS$_i$ by the AS = m*273.41 + m*15 + (2m+1)*0.030322 + (3m-3)*0.000933 + 0.000933 + m*221.60 + m*0.89 = 0.028456 + m*510.96 milliseconds = 0.000028 + 0.51*m seconds

Time to verify the MS by the AS = 273.41 + 15 + 8.8 + 9.1 + m*221.60 + m*0.89 = 306.31 + m*222.49 milliseconds = 0.3 + 0.22*m seconds

Time to verify the AS by the MS$_i$ = m*0.030322 + m*221.60 = m*221.63 milliseconds = 0.22*m seconds

Time to verify the AS by the MS = 9.1 milliseconds = 0.0091 seconds

Thus, Total delay in Phase-1= 0.309128 + m*0.95 seconds

BOPSMS Phase-2: Operations to verify the MS by the MS$_i$ = r*D$\{\}_{DK_1}$, r*E$\{\}_{DK_1}$

Operations to verify the MS$_i$ by the MS = r*D$\{\}_{DK_1}$

Time to verify the MS by the MS$_i$ = r*(9.1 + 8.8) = r*17.9 milliseconds = r*0.02 seconds

Time to verify the MS$_i$ by the MS = r*9.1 milliseconds = r*0.009 seconds (= r*0.01 seconds)

Total verification delay in Phase-2 = r*0.03 seconds

9.7.3 Re-batch Verification Delay

The time delay for re-batch verification can be estimated as:
Re-batch computation = [P = (m-1-t)*$\sum_{i=1}^{m}$ P$_i$, Z$'$ = (m-1-t)*$\sum_{i=1}^{m}$ Z$'_i$, R$'$ = (ReqNo $\oplus$ P), (Z$'$ - R$'$), Q = (m-1-t)*$\sum_{i=1}^{m}$ Y$_i$]
Operations required for re-batch verification = (m-1-t)*T$_{add}$ + (m-1-t)*T$_{add}$ + T$_{mul}$ + (m-1-t)*T$_{add}$ + T$_{sub}$ = 3(m-1-t)*T$_{add}$ + T$_{mul}$ + T$_{sub}$
Time delay for re-batch verification = 3*(m-1-t) *0.000933 + 0.030322 + 0.000933 = 0.028456 + (m-t)*0.002799 milliseconds (= 0.000028 + (m-t)*0.000002 seconds), where t is the number of malicious MS$_i$ that have been removed in the re-batch authentication. Figure 9.7(a) shows the BOPSMS protocol execution time, when authentication requests m = 10, 20, 50, 100 and subsequent authentication requests r = 1, 2, 5, 10. The verification time for phase-1 and phase-2 of BOPSMS protocol can be observed from Figure 9.7(b) and Figure 9.7(c) respectively when m = 10, 20, 50, 100 for phase-1 and r = 1, 2, 5, 10, 50 for phase-2. Figure 9.7(d) illustrates re-batch verification time for the BOPSMS protocol, when m = 10, 20, 50, 100 and invalid requests in a batch t = 2, 4, 6, 8, 10.

9.8 Formal Proof of BOPSMS Protocol

In order to formally proof the authentication process of BOPSMS protocol, BAN-Logic symbols are used. The notations of BAN-Logic are described in section 3.7.4.

1. *The Formal Messages in BOPSMS Protocol:*

   Phase-1: (1) MS $\rightarrow$ MS$_i$: Ta, ReqNo, TID$_1$, Actcode$_1$, MAC1$_1$

   (2) MS$_i$ $\rightarrow$ AS: Ta, ReqNo, Actcode$_1$, TID$_1$, MAC1$_1$, Tk, Y$_i$,Z$_i$, TID$_i$, MAC2$_i$, Actcode$_i$; DK$_1$= f$_1$(Ta)$_{SK_1}$, DK$_i$= f$_1$(Tk)$_{SK_i}$, MS $\overset{SK_1}{\leftrightarrow}$ AS, MS$_i$ $\overset{SK_i}{\leftrightarrow}$ AS

   (3) AS $\rightarrow$ MS: $\{$T$_{m+1}$, ReqNo, ExpT$\}_{DK_1}$

   (4) MS $\rightarrow$ AS: $\{$T$_{m+1}\}_{DK_1}$

   (5) AS $\rightarrow$ MS$_i$: P$_i$, MAC3$_i$, $\{$T$_{m+2}$, ReqNo, ExpT, DK$_1\}_{DK_i}$

   (6) AS $\rightarrow$ MS: $\{$ReqNo, Ta$\}_{DK_1}$

   Phase-2: (1) MS $\rightarrow$ MS$_i$: $\{$TID$_1$, ReqNo, T$_j\}_{DK_1}$

(2) MS$_i$ → MS: $\{$ReqNo, T$_j\}_{DK_1}$

2. *Security Assumptions:*

   It is assumed that SK$_i$ key is shared between the MS$_i$ and the AS.

   (1) MS$_i$ has the secure key SK$_i$ and MS$_i$ $|\equiv$ MS$_i$ $\overset{SK_i}{\leftrightarrow}$ AS

   (2) AS has the secure key SK$_i$ and AS $|\equiv$ MS$_i$ $\overset{SK_i}{\leftrightarrow}$ AS

3. *Security Analysis:*

   Phase-1: (1) MS → MS$_i$: MS $|\equiv$ #(Ta); MS$_i$ $\triangleleft$ Ta, ReqNo, Actcode$_1$, f$_2$(IMSI$_1$, Ta)$_{DK_1}$, MAC1$_1$

   (2) MS$_i$ → AS: MS$_i$ $|\equiv$ #(Tk); AS $\triangleleft$ Ta, ReqNo, Actcode$_1$, TID$_1$, MAC1$_1$, Tk, (k$_i$ $\oplus$ IMSI$_i$), ((k$_i$ + DK$_i$ $\oplus$ ReqNo) mod m), TID$_i$, MAC2$_i$, Actcode$_i$; DK$_1$= f$_1$(Ta)$_{SK_1}$, DK$_i$= f$_1$(Tk)$_{SK_i}$, MS $\overset{SK_1}{\leftrightarrow}$ AS, MS$_i$ $\overset{SK_i}{\leftrightarrow}$ AS

   (3) On receiving message (2), the AS computes and compares MAC1$_1$?= MAC1$_1$ and MAC2$_i$?= MAC2$_i$. If they hold, then computes SIMcode$_1$, SIMcode$_i$, DK$_i$ = f$_1$(Tk)$_{SK_i}$, and IMSI$_i$ = f$_2$(TID$_i$, Tk)$_{DK_i}$ along with DK$_1$ = f$_1$(Ta)$_{SK_1}$ and IMSI$_1$ = f$_2$(TID$_1$, Tk)$_{DK_1}$. Thereafter, the AS checks whether Q = $\sum_{i=1}^{m}$(Y$_i$)==R. If yes, then all the MS$_i$ are successfully verified by the AS. Then, AS → MS: $\{$T$_{m+1}$, ReqNo, ExpT$\}_{DK_1}$

   (4) MS generates DK$_1$ and MS → AS: $\{$T$_{m+1}\}_{DK_1}$

   (5) On receiving message (4), the AS checks the stored T(k+1) with the received one. Then, AS → MS$_i$: (DK$_i$ $\oplus$ IMSI$_i$), $\{$T$_{m+2}$, ReqNo, ExpT, f$_1$(Ta)$_{SK_1}\}_{DK_i}$, MAC3$_i$

   (6) On receiving message (5), the MS$_i$ computes and compares MAC3$_i$?= MAC3$'_i$. If both are equal, then MS$_i$ computes (DK$_i$ $\oplus$ IMSI$_i$) and compares with the received one. Then, AS → MS: $\{$ReqNo, Ta$\}_{DK_1}$. The MS checks #Ta, if it is same as the used in message (1), then the authentication is completed.

   Phase-2: (1) MS → MS$_i$: $\{$TID$_1$, ReqNo, Ts$\}_{DK_1}$, then MS$_i$ checks the TID$_i$ and ReqNo along with Ts <= ExpT. If it holds, then retrieve corresponding DK$_1$.

   (2) MS$_i$ → MS: $\{$ReqNo, Ts$\}_{DK_1}$, On receiving at MS, if ReqNo and Ts match, then the authentication is successful.

4. *Message Meaning Rule:*

   (1) $\frac{MS|\equiv(MS\overset{SK_1,DK_1}{\leftrightarrow}AS)\wedge(MS_i\overset{SK_i,DK_i}{\leftrightarrow}AS),MS\triangleleft f_2(IMSI_1,Ta)_{DK_1}}{MS|\equiv AS\upharpoonright f_2(IMSI_1,Ta)_{DK_1}}$

(2) $\frac{AS|\equiv(AS\overset{SK_1,DK_1}{\leftrightarrow}MS)\wedge(AS\overset{SK_i,DK_i}{\leftrightarrow}MS_i),AS\triangleleft f_2(TID_1,Ta)_{DK_1}}{AS|\equiv MS\upharpoonleft f_2(TID_1,Ta)_{DK_1}}$

5. *Nonce/Timestamp Verification Rule:*

   (1) $\frac{MS|\equiv\#(Ta)\wedge\#(T(k+1)),MS|\equiv AS\upharpoonleft f_2(IMSI_1,Ta)_{DK_1}}{MS|\equiv AS|\equiv f_2(IMSI_1,Ta)_{DK_1}}$

   (2) $\frac{AS|\equiv\#(T(k+1))\wedge\#(T(k+2)),AS|\equiv MS\upharpoonleft f_2(TID_1,Ta)_{DK_1}}{AS|\equiv MS|\equiv f_2(TID_1,Ta)_{DK_1}}$

6. *Jurisdiction Rule:*

   (1) $\frac{MS|\equiv AS\Rightarrow f_2(IMSI_1,Ta)_{DK_1},MS_i\triangleleft MS\upharpoonleft f_2(IMSI_1,Ta)_{DK_1}}{MS|\equiv AS|\equiv MS_i}$

   (2) $\frac{AS|\equiv MS\Rightarrow f_2(TID_1,Ta)_{DK_1},AS\triangleleft AS\upharpoonleft f_2(TID_1,Ta)_{DK_1}}{AS|\equiv MS|\equiv MS_i}$

   (3) $\frac{MS_i|\equiv AS\Rightarrow f_2(IMSI_i,Tk)_{DK_i},AS\triangleleft MS_i\upharpoonleft f_2(IMSI_i,Tk)_{DK_i}}{MS_i|\equiv AS|\equiv MS}$

   (4) $\frac{AS|\equiv MS_i\Rightarrow f_2(TID_i,Tk)_{DK_i},AS\triangleleft AS\upharpoonleft f_2(TID_i,Tk)_{DK_i}}{AS|\equiv MS_i|\equiv MS}$

7. *Protocol Goals:*

   a) Mutual authentication:

   MS $|\equiv$ MS$_i$ $|\equiv$ AS $\wedge$ AS $|\equiv$ MS$_i$ $|\equiv$ MS $\rightarrow$ MS $|\equiv$ AS $\wedge$ MS$_i$ $|\equiv$ AS Thus, mutual authentication between MS-AS and MS$_i$-AS hold.

   b) Session key agreement:

   The DK$_1$ key between the MS and the AS, and the DK$_i$ keys between each MS$_i$ and the AS provide session key agreement.

   MS $|\equiv$ DK$_1$ $\wedge$ #(Ta), since DK$_1$ = f$_1$(Ta)$_{SK_1}$, and

   MS$_i$ $|\equiv$ DK$_i$ $\wedge$ #(Tk), since DK$_i$ = f$_1$(Tk)$_{SK_i}$

   c) Freshness of messages:

   AS $|\equiv$ #(Ta) $\wedge$ MS $|\equiv$ #(Ta), AS $|\equiv$ #(Tk) $\wedge$ MS$_i$ $|\equiv$ #(Tk), AS $|\equiv$ #(T(k+1)) $\wedge$ MS$_i$ $|\equiv$ #(T(k+1))

   Thus, key freshness between the MS and the AS, and between each MS$_i$ and the AS hold.

   d) Privacy between the MS and the AS, and between each MS$_i$ and the AS:

   $\frac{MS|\equiv(MS\overset{DK_1}{\leftrightarrow}AS),MS\triangleleft\{Msg\}DK_1}{MS|\equiv AS\upharpoonleft Msg}\wedge\frac{MS_i|\equiv(MS_i\overset{DK_i}{\leftrightarrow}AS),MS_i\triangleleft\{Msg\}_{DK_i}}{MS_i|\equiv AS\upharpoonleft Msg}$

   $\frac{MS|\equiv(MS\overset{DK_1}{\leftrightarrow}AS),MS\triangleleft f_2(IMSI_1,Ta)_{DK1}}{MS|\equiv AS\upharpoonleft IMSI_1}\wedge\frac{MS_i|\equiv(MS_i\overset{DK_i}{\leftrightarrow}AS),MS_i\triangleleft f_2(IMSI_i,Tk)_{DKi}}{MS_i|\equiv AS\upharpoonleft IMSI_i}$

9.9 Chapter Summary

This chapter presents a BOPSMS protocol for end-to-end secure transmission of SMS in a batch. This protocol provides mutual authentication and hides original IMSI between each MS and the AS. It has been shown that in batch authentication, from the device to the server proposed scheme for the AS lowers transmission bandwidth by 23.81%, 56.17%, and 61.68% in comparison to ABAKA, BLS, and ECDSA-AKA protocols respectively, when when number of authentication requests (m) = 50, 100, 200, 500, and 1000. Further, from the server to the device, the proposed AS scheme reduces the communication bandwidth by 80% and 90.5% as compared to ABAKA and ECDSA-AKA protocols respectively. The reliability analysis of BOPSMS protocol is performed in terms of Hypergeometric distribution probability, in which, out of 1000 (assumption) authentication requests (server can handle 900 at one time), the probability of exact t invalid authentication requests in a batch is maximum (7.36) for t = 3 and minimum (0.0013) for t = 5, when the maximum number of invalid requests are considered as 10.

Chapter 10

Conclusion

10.1 Discussion

In this thesis, authentication and key agreement protocols for the GSM, UMTS, and LTE cellular networks are proposed and analyzed. Additionally, for SMS-based applications in the cellular networks, secure and efficient AKA protocols are proposed for providing the value added services to the mobile users and end-to-end secure transmission of SMS over the network.

It is observed that a secure and efficient SAKA protocol offers bidirectional authentication and better service for the GSM network, when the mobile users are in roaming area. The proposed GSM architecture with SAKA protocol and secure algorithms is free from various security attacks that exist in the original GSM architecture. It is reported that the SAKA protocol is the most efficient protocol in terms of bandwidth utilization as compared to all the existing and proposed GSM authentication protocols from the literature. It also generates lesser communication overhead in comparison to all the exiting GSM authentication protocols.

The work reports that the ES-AKA protocol defeats the security limitations that exist in the original UMTS-AKA protocol and provides better security in the UMTS network. The protocol prevents flood-based DoS attack in the UMTS network, which was not provided by any GSM authentication protocol. An improved cipher algorithm named MAES-128 has been proposed for the UMTS network, which provides faster encryption and decryption than the existing KASUMI and AES algorithms. Thus, the MAES-128 algorithm is suitable for ciphering messages in UMTS network. Further, the problem of DoS attack, where an adversary can have more powerful resources than a normal user, is overcome by a puzzle-based Secure-AKA protocol. The Secure-AKA protocol is the only available protocol in the literature, which solves the issue of DoS attack (flood-based DoS, DDoS, LDoS) and protects the actual identity of the mobile user in the UMTS network. Thus, it reports a significant contribution to the literature. It is also noticed that this protocol saves more bandwidth consumption in comparison to all the existing AKA protocols in the UMTS network with lesser communication and computation overheads.

Furthermore, the existing limitations of EPS-AKA in 4G LTE network, *i.e.*, the clear text transmission of IMSI and KSI$_{ASME}$ over the network, are improved without increasing the communication overhead. This is the first LTE authentication protocol, which protects clear text KSI$_{ASME}$ over the network. The protocol also prevents various security attacks and resolves the synchronization issue that exists in the EPS-AKA protocol with nearby the same bandwidth consumption as in EPS-AKA.

Further, in this work, we have considered the application of SMS security in cellular networks by proposing the SecureSMS protocol, which prevents various security attacks, when the mutual authentication is performed between the MS and the AS to provide the secure delivery of value added services. It is observed that the SecureSMS protocol saves more than half of the total bandwidth during the authentication than the existing SMSSec and PK-SIM protocols. The protocol is also efficient as it produces lower overheads. The AES and ECDSA algorithms have been tested on Nokia 6300 successfully. The proposed scheme for SecureSMS maintains authentication, confidentiality, integrity, and non-repudiation security services. The solution to this problem has been extended by proposing the EXTVAS-AKA protocol, which provides secure delivery of value added services in a batch. It is found that EXTVAS-AKA execution time and verification delay at MS and at AS increase proportionally to the number of authentication requests in a batch. Further, it is noticed that the re-batch verification delay is lower, if the number of malicious authentication requests are large. This protocol lowers the bandwidth than the ABAKA, B LS, and ECDSA-AKA protocols.

Another application considered is to provide end-to-end secure transmission of SMS between mobile users. This is achieved by our proposed EasySMS protocol, which efficiently sends Secure Message in both the scenarios, when the sender and the receiver mobile users are in same network area or in different network area. This protocol securely transmits the secret key to the recipient user and efficiently utilizes the bandwidth than SMSSec and PK-SIM protocols. Further, the MAES cipher algorithm provides faster encryption and decryption in comparison to the other existing algorithms like DES, TripleDES with 2 keys, TripleDES with 3 keys, AES, Twofish, RC2, RC6, and CAST6. This work adds to the existing state of the knowledge as the EasySMS is the first symmetric key cryptography-based protocol, which delivers end-to-end secure SMS. This work is further extended for the scenario, when a user sends secure SMS messages to multiple recipients simultaneously. To meet this objective, the BOPSMS protocol is proposed, which keeps the original user identity private during the authentication. The proposed approach for the server in BOPSMS to handle multiple requests reduces the bandwidth utilization in comparison to ABAKA, BLA, and ECDSA-AKA protocols. The communication and computation overheads of BOPSMS increases as the number of target MS grows. The BOPSMS protocol is more efficient in the scenario, when a mobile user sends number of SMS messages to a group of multiple recipients within a session as the subsequent authentication process takes very less execution time in comparison to the first time authentication. This symmetric key cryptography-based protocol is the first protocol that ensures the secure transmission of SMS in a batch and enables users to send multiple SMS simultaneously.

10.2 Future Scope of the Work

This section briefly describes the future scope of the thesis work. The work carried out can be extended in the following directions:

1. *Attacks Analysis on GSM Phone using Open Source Software*

   Two projects are known to us, in order to implement and observe GSM protocol stack with open source software: OpenBSC and OsmocomBB. Unfortunately, there is no open source implementation available for the UMTS and LTE networks. The developers of OsmocomBB project are working very hard towards incorporating the functionalities of 3G in this project. The extension of this work may enable the possibility of various other vulnerabilities and attacks, and can open the doors to think differently with the analysis of open GSM stack.

2. *Real-time Simulation of Proposed AKA Protocols in Cellular Networks*

   For in-depth study of the proposed protocols, the source code of existing AKA protocols need to be altered in order to implement and simulate our proposed protocols in the real cellular networks that require infrastructure (telecommunication laboratory).

3. *Secure SMS Based Mobile Banking for India*

   The service of SMS can be extended in secure mobile banking for the people who live in the rural part of India. Typically, they do not have java enabled mobile phones and have limited or no Internet facility. Thus, it is a research direction to secure SMS and its banking environment against existing threats and attacks by providing the security services authentication, confidentiality, integrity, non-repudiation.

4. *Efficient and Secure Solution for Healthcare Monitoring*

   The proposed solution can be connected to a secure information system through SMS for healthcare monitoring, where the objectives are to manage e-records and remote monitoring of patient′s vital signs Diabetes, Hypertension, ECG etc., in rural areas that have limited or no communications infrastructure. It will enable mobile-monitoring through secure SMS in emergency scenarios and making remote diagnosis possible.

Bibliography

[1] Lee C.H., Hwang M.S., Yang W.P. (1999), Enhanced privacy and authentication for the global system for mobile communications, Wireless Networks, 5, 231-243.

[2] Harn L., Lin H. (1995), Modifications to enhance the security of GSM, 5th National Conference on Information Security, Taipei, Taiwan, pp. 74-76.

[3] Lin H., Harn L. (1995), Authentication protocols for personal communication system, ACM Special Interest Group on Data Communications (SIGCOMM′95), Cambridge, USA, pp. 256-261.

[4] Brumley B. (2004), A3/A8 and COMP128, T-79.514 Special Course on Cryptology, pp. 1-18. www.tcs.hut.fi/ Studies/T-79.514/slides/S5.Brumley-comp128.pdf

[5] Briceno M., Goldberg I., Wager D. (1999), A pedagogical implementation of the GSM A5/1 and A5/2 voice privacy encryption algorithms. http://www.cryptome.org/gsm-a512.htm

[6] Rao J.R., Rohatgi P., Scherzer H., Tinguely S. (2002), Partitioning attacks: or how to rapidly clone some GSM cards, IEEE Symposium on Security and Privacy (S&P′02), California, USA, pp. 31-41.

[7] Biryukov A., ShamirA., Wagner D. (2001), Real time: cryptanalysis of A5/1 on a PC, 8th International Workshop on Fast Software Encryption, LNCS 1978, Yokohama, Japan, pp. 1-18.

[8] Wagner D. (2009), GSM cloning, Smartcard Developer Aassociation and ISAAC Security Research Group.

[9] Barkan E., Biham E., Keller N. (2003), Instant cipher text only cryptanalysis of GSM encrypted communication, 23rd Annual International Cryptology Conference (CRYPTO′03), LNCS 2729, California, USA, pp. 600-616.

[10] Biham E., Dunkelman O. (2000), Cryptanalysis of the A5/1 GSM stream cipher, INDOCRYPT, LNCS 1977, Kolkata, India, pp. 43-51.

[11] Al-tawil K., Akrami A., Youssef H. (1998), A new authentication protocol for GSM network, 23^{rd} IEEE Annual Conference on Local Computer Network, Boston, Massachusetts, USA, pp. 21-30.

[12] Lo C.C., Chen Y.J. (1999), A secure communication architecture for GSM networks, IEEE Pacific Rim Conference on Communnications, Computers and Signal Processing, Victoria, Canada, pp. 221-224.

[13] Lo C.C., Chen Y.J. (1999), Secure communication mechanisms for-GSM networks, IEEE Transactions on Consumer Electronics, 45 (4), 1074-1080.

[14] Lee C.C., Hwang M.S., Yang W.P. (2003), Extension of authentication protocol for GSM, IEE Proceedings of Communication, 150(2), 91-95.

[15] Chang C.C., Lee J.S., Chang Y.F. (2005), Efficient authentication protocols of GSM, Computer Communication, 28(8), 921-928.

[16] Bocan V., Cretu V. (2006), Threats and countermeasures in GSM networks, Journal of Networks, 1(6), 18-27.

[17] Fanian A., Berenjkoub M., Gulliver T.A. (2009), A new mutual authentication protocol for GSM networks, Canadian Conference on Electrical and Computer Engineering (CCECE′09), Newfoundland, Canada, pp. 798-803.

[18] Fanian A., Berenjkoub M., Gulliver T.A. (2010), A symmetric polynomial based mutual authentication protocol for GSM networks, IEEE Wireless Communications and Networking Conference (WCNC′10), Sydney, Australia, pp. 1-6.

[19] Yubo S., Xili H., Zhiling L. (2011), The GSM/UMTS phone number catcher, 3^{rd} International Conference on Multimedia Information Networking and Security, Shanghai, China, pp. 520-523.

[20] Southern E., Ouda A., Shami A. (2011), Solutions to security issues with legacy integration of GSM into UMTS, 6^{th} International Conference on Internet Technology & Secured Transactions, Abu Dhabi, pp. 614-619.

[21] Firoozjaei M.D., Vahidi J. (2012), Implementing geo-encryption in GSM cellular network, 9^{th} International Conference on Communications (COMM), Bucharest, Romania, pp. 299-302.

[22] Lin Y.B., Chang M.F., Hsu M.T., Wu L.Y. (2005), One-pass GPRS and IMS authentication procedure for UMTS, IEEE Journal on Selected Areas in Communications, 23(6), 1233-1239.

[23] Cheng K.M., Chang T.Y., Lo J.W. (2010), Cryptanalysis of security enhancement for a modified authentication key agreement protocol, International Journal of Network Security, 11(1), 55-57.

[24] Chang C.C., Hwang K.F., Lin I.C. (2003), Security enhancement for a modified authenticated key agreement protocol, International Journal of Computational and Numerical Analysis and Applications, 3(1), 1-7.

[25] Seo D., Sweeney P. (2006), Simple authenticated key agreement algorithm, Electronics Letters, 35(13), 1073-1074.

[26] Godor G., Imre S. (2006), Novel authentication algorithm public key based cryptography in mobile phone systems. International Journal of Computer Science and Network Security, 6(2B), 126-134.

[27] Akhtar M.N., Minhas A.A., Ahmad J. (2010), A novel security algorithm for universal mobile telecommunication system, International Journal of Multimedia Ubiquitous Engineering, 5(1), 1-18.

[28] Al-saraireh J., Yousef S. (2006), A new authentication protocol for UMTS mobile networks, EURASIP Journal on Wireless Communications and Networking, 2006(2), 1-10.

[29] Ou H.H., Hwang M.S., Jan J.K. (2010), A cocktail protocol with the authentication and key agreement on the UMTS, Journal of Systems and Software, 83(2), 316-325.

[30] Lee C.C., Chen C.L., Ou H.H., Chen L.A. (2013), Extension of an efficient 3GPP authentication and key agreement protocol, Wireless Personal Commications, 68(3), 861-872.

[31] Huang C.M., Li J.W. (2005), Authentication and key agreement protocol for UMTS with low bandwidth consumption, 19^{th} IEEE International Conference on Advanced Information Networking and Applications (AINA$'$05), Taiwan, pp. 392-397.

[32] Chun I.E., Ho P.H., Chen H.Y. (2007), Nested one-time secret mechanisms for fast mutual authentication in mobile communications, IEEE Wireless Communications and Networking Conference (WCNC$'$07), Hong Kong, pp. 2714-2719.

[33] Al-saraireh J., Yousef S. (2006), Extension of authentication and key agreement protocol (AKA) for universal mobile telecommunication system

(UMTS), International Journal of Theoretical and Applied Computer Sciences, 1(1), 109-118.

[34] Wu S., Zhu Y., Pu Q. (2010), Security analysis of a cocktail protocol with the authentication and key agreement on the UMTS, IEEE Communications Letters, 14(4), 366-368.

[35] Huang Y.L., Shen C.Y., Shieh S.W. (2011), S-AKA: a provable and secure authentication key agreement protocol for UMTS networks, IEEE Transactions on Vehicular Technology, 60(9), 4509-4519.

[36] Peng C., Li C.Y., Tu G.H., Lu S., Zhang L. (2012), Mobile data charging: new attacks and countermeasures, ACM Conference on Computer and Communications Security (CCS'12), Raleigh, USA, pp. 195-204.

[37] Vintila C.E., Patriciu V.V., Bica I. (2011), Security analysis of LTE access network, 10^{th} International Conference on Networks (ICN'11), St. Maarten, Netherlands Antilles, pp. 29-34.

[38] Bikos A.N., Sklavos N. (2013), LTE/SAE security issues on 4G wireless networks, IEEE Security & Privacy, 11(2), 55-62.

[39] Hussain S., Hamid Z., Khattak N.S. (2006), Mobility management challenges and issues in 4G heterogeneous networks, 1^{st} International Conference on Integrated Internet Ad Hoc and Sensor Networks (InterSense$'$06), France, Article No. 14, pp. 1-6.

[40] Matos A., Sargento S., Aguiar R. (2007), Embedding identity in mobile environments, 2^{nd} ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture (MobiArch$'$07), Article No. 6, Kyoto, Japan, pp. 1-8.

[41] Kϕien G.M. (2011), Mutual entity authentication for LTE, 7^{th} International Conference of Wireless Communication and Mobile Computing (IWCMC$'$11), Turkey, pp. 689-694.

[42] Hadiji F., Zarai F., Kamoun A. (2009), Authentication protocol in fourth generation wireless networks, 6^{th} IEEE and IFIP International Conference on Wireless & Optical Communication Networks, Cairo, Egypt, pp. 1-4.

[43] Gu L., Gregory M.A. (2011), A green and secure authentication for 4^{th} generation mobile network, Australasian Telecommunication Network & Applications Conference (ATNAC$'$11), Melbourne, Australia, pp. 1-7.

[44] He D., Wang J., Zheng Y. (2008), User authentication scheme based on self-certified public-key for next generation wireless network, International Symposium on Biometrics and Security Technologies (ISBAST′08), Islamabad, Pakistan, pp. 1-8.

[45] Zheng Y., He D., Tang X., Wang H. (2005), AKA and authorization scheme for 4G mobile networks based on trusted mobile platform, International Conference on Information, Communications and Signal Processing, Bangkok, pp. 976-980.

[46] Purkhiabani M., Salahi A. (2012), Enhanced authentication and key agreement procedure of next generation evolved mobile n/w, International Journal of Information & Electrical Engineering, 2(1), 69-77.

[47] Choudhury H., Roychoudhury B., Saikia D.K. (2012), Enhancing user identity privacy in LTE, 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communication, Liverpool, UK, pp. 949-957.

[48] Lo J.L.C., Bishop J., Eloff J.H.P. (2008), SMSSec: an end-to-end protocol for secure SMS, Computers & Security, 27(5-6), 154-167.

[49] Rongyu H., Guolei Z., Chaowen C., Hui X., Xi Q., Zheng Q. (2009), A PK-SIM card based end-to-end security framework for SMS, Computer Standards and Interfaces, 31(4), 629-641.

[50] Hassinen M. (2006), Java based public Key infrastructure for SMS messaging, International Conference on Information and Communication Technologies (ICTTA′06), Damascus, Syria, pp. 88-93.

[51] Wu S., Tan C. (2009), A high security framework for SMS, International Conference on Biomedical Engineering and Informatics (BMEI′09), Hangzhou, China, pp. 1-6.

[52] De S.A., Castiglione A., Cattaneo G., Cembalo M., Petagna F., Ferraro P.U. (2010), An extensible framework for efficient secure SMS, International Conference on Complex, Intelligent and Software Intensive Systems, Krakow, Poland, pp. 843-850.

[53] Castiglione A., Cattaneo G., Cembalo M., De S.A., Faruolo P., Petagna F., Ferraro P.U. (2012), Engineering a secure mobile messaging framework, Computers & Security, 31(6), 771-781.

[54] Toorani M., Shirazi A.B. (2008), SSMS-a secure SMS messaging protocol for the m-payment systems, 13th IEEE Symposium on Computers and Communications, Marrakech, Morocco, pp. 700-705.

[55] Hassinen M., Hypponen K. (2005), Strong mobile authentication, 2^{nd} International Symposium on Wireless Communication Systems, Siena, Italy, pp. 96-100.

[56] Horn G., Martin K.M., Mitchell C.J. (2002), Authentication protocols for mobile network environment value-added services, IEEE Transactions on Vehicular Technology, 51(2), 383-392.

[57] Castiglione A., De P.R., De S.A. (2009), Do you trust your phone?, 10^{th} International Conference on E-Commerce and Web Technologies (EC-Web′09), LNCS 5692, Linz, Austria, pp. 50-61.

[58] Huang J.L., Yeh L.Y., Chien H.Y. (2011), ABAKA: An anonymous batch authenticated and key agreement scheme for value-added services in vehicular ad hoc networks, IEEE Transactions on Vehicular Technology, 60(1), 248-262.

[59] Zhang C., Lu R., Lin X., Ho P.H., Shen X. (2008), An efficient identity based batch verification scheme for vehicular sensor networks, 27^{th} IEEE Conference on Computer Communications (INFOCOM), Phoenix, USA, pp. 816-824.

[60] Zhang C., Lin X., Lu R., Ho P.H., Shen X. (2008), An efficient message authentication scheme for vehicular communications, IEEE Transactions on Vehicular Technology, 57(6), 3357-3368.

[61] Lin X., Sun X., Ho P.H., Shen X. (2007), GSIS: A secure and privacy preserving protocol for vehicular communications, IEEE Transaction on Vehicular Technology, 56(6), 3442-3456.

[62] Boneh D., Gentry C., Lynn B., and Shacham H. (2003) Aggregate and verifiably encrypted signatures from bilinear maps, EUROCRYPT, Warsaw, Poland, pp. 416-432.

[63] Yeh L.Y., Huang Y.L., Joseph A.D., Shieh S.W., Tsaur W.J. (2012), A batch-authenticated and key agreement framework for P2P-based online social networks, IEEE Transactions on Vehicular Technology, 61(4), 1907-1924.

[64] Mondal P., Desai P., Ghosh S. (2013), An efficient SMS-based framework for public health surveillance, IEEE Point-of-Care Healthcare Technologies (PHT′13), Bangalore, India, pp. 244-247.

[65] Wu J., Stinson D.R. (2011), Three improved algorithms for multipath key establishment in sensor networks using protocols for secure message transmission, IEEE Transactions on Dependable and Secure Computing, 8(6), 929-937.

[66] Calandriello G., Papadimitratos P., Hubaux J.P., Lioy A. (2011), On the performance of secure vehicular communication systems, IEEE Transactions on Dependable and Secure Computing, 8(6), 898-912.

[67] Zhou Y., Zhu X., Fang Y. (2010), MABS: multicast authentication based on batch signature, IEEE Transactions on Mobile Computing, 9(7), 982-993.

[68] Akinyele J.A., Green M., Hohenberger S. (2013), Machine-generated algorithms, proofs and software for batch verification of digital signature schemes, ACM conference on Computer and Communications Security (CCS′12), Raleigh, USA, 474-487.

[69] Yang C.C., Tang Y.L., Wang R.C. (2005), A secure and efficient authentication protocol for anonymous channel in wireless communications, Applied Mathematics and Computation, 169(2), 1431-1439.

[70] Traynor P., Enck W., McDaniel P., Porta T.L. (2009), Mitigating attacks on open functionality in SMS-capable cellular networks. IEEE/ACM Transactions on Networking, 17 (1), 40-53.

[71] Stach J.F., Park E.K. and Makki K. (1999), Performance of an enhanced GSM protocol supporting non-repudiation of service, Computer Communications, 22(7), 675-680.

[72] SDA releases GSM voice-privacy algorithm A5/1, The Smartcard Developer Association (SDA), http://www.scard.org/gsm/

[73] Briceno M., Goldberg I., Wagner D. (1999), A pedagogical implementation of the A5/1, The Smartcard Developer Association (SDA). http://www.scard.org/gsm/a51.html

[74] Ekdahl P., Johansson T. (2003), Another attack on A5/1, IEEE Transactions on Information Theory, 49(1), 284-289.

[75] Guneysu T., Kasper T., Novotny M., Paar C. (2008), Cryptanalysis with COPACOBANA, IEEE Transactions on Computers, 57(11), 1498-1513.

[76] Bajaj N. (2011), Enhancement of A5/1 using variable feedback polynomials of LFSR, International Conference on Emerging Trends in Networks and Computer Communications (ETNCC′11), Udaipur, India, pp. 55-60.

[77] Peyrin T., Sasaki Y., Wang L. (2012), Generic related-key attacks for HMAC, Advances in Cryptology (ASIACRYPT′12), Beijing, China, pp. 580-597.

[78] Stallings W. (2011) Cryptography and Network Security, 5th Edition, Pearson Education.

[79] Mitchell C.J., Dent A.W. (2010), International standards for stream ciphers: a progress report, Royal Holloway, University of London, pp. 1-12.

[80] Handschuh H., Paillier P. (2000), Reducing the collision probability of alleged Comp128. In: Quisquater J. J., Schneier B. (ed) Smart card research and applications, LNCS 1820, pp. 380-385.

[81] Hash collision Q&A (2005), Cryptography research inquiries about the hash collision attacks at CRYPTO-2004 conference (16 February 2005), http://web.archive.org/web/20080717103333/http://www.cryptography.com/cnews/hash.html

[82] Golde N., Redon K., Seifert J.P. (2013), Let me answer that for you: exploiting broadcast information in cellular networks, 22nd USENIX Security Symposium, Washington, USA, pp. 33-48.

[83] Yang G., Gerla M., Sanadidi M.Y. (2004), Defense against low rate tcp-targeted denial-of-service attacks, 9th International Symposium on Computers and Communications (ISCC′04), Alexandria, Egypt, pp.345350.

[84] Ptz S., Schmitz R., Martin T. (2001), Security mechanisms in UMTS, Datenschutz und Datensicherheit, 25, 1-10.

[85] 3rd Generation partnership project (2001) Technical specification group services and system aspects, 3G security, specification of the 3GPP confidentiality and integrity algorithms, document 2: KASUMI specification. http://www.3gpp.org/DynaReport/35202.htm

[86] Matsui M. (1997), Block encryption algorithm MISTY, Fast Software Encryption, LNCS 1267, Haifa, Israel, pp. 64-74.

[87] Biham E., Dunkelman O., Keller N. (2005), A related-key rectangle attack on the full KASUMI, Advances in Cryptology (ASIACRYPT′05), LNCS 3788, Chennai, India, pp. 443-461.

[88] McKay K. (2005) Trade-offs between energy and security in wireless networks. Thesis, Worcester Polytechnic Institute.

[89] Biryukov A., Dunkelman O., Keller N., Khovratovich D., Shamir A. (2010), Key recovery attacks of practical complexity on AES-256 variants with up to 10 rounds, Advances in Cryptology (EUROCRYPT′10), LNCS 6110, French Riviera, France, PP. 299-319.

[90] Dunkelman O., Keller N. (2010), The effects of the omission of last rounds mixcolumns on AES, Journal Information Processing Letters, 110(8-9), 282-287.

[91] Lu C.F., Kan Y.S., Chiang H.L., Yang C.H. (2003), Fast implementation of AES cryptographic algorithms in smart cards, 37th IEEE Annual International Carnahan Conference on Security Technology, Taipei, Taiwan, pp. 573-579.

[92] Abliz M., Znati T. (2009), A guided tour puzzle for denial of service prevention, Annual Computer Security Applications Conference (ACSAC′09), Hawaii, USA, pp. 279-288.

[93] Feng W., Kaiser E., Luu A. (2005), The design and implementation of network puzzles, IEEE International conference on Computer Communication (INFOCOM′05), Miami, USA, 4, pp. 2372-2382.

[94] Juels A., Brainard J. (1999), Client puzzles: A cryptographic countermeasure against connection depletion attacks, Network and Distributed System Security Symposium (NDSS′99), San Diego, USA, pp. 151-165.

[95] Rivest R.L., Shamir A., Wagner D.A. (1996), Time-lock puzzles and timed-release crypto, Technical Report TR-684, MIT Laboratory for Computer Science.

[96] Stebila D., Berkant U. (2009), Towards denial-of-service-resilient key agreement protocols, 14th Australasian Conference on Information Security and Privacy, LNCS 5594, Brisbane, Australia, pp. 389-406.

[97] Stebila D., Kuppusamy L., Rangasamy J., Boyd C., Nieto J.G. (2011), Stronger difficulty notions for client puzzles and denial-of-service-resistant protocols, Topics in Cryptology (CT-RSA′11), LNCS 6558, San Francisco, USA, pp. 284-301.

[98] Rangasamy J., Stebila D., Kuppusary L., Boyd C., Nieto J.G. (2012), Efficient modular exponentiation-based puzzles for denial-of-service protection, Information Security and Cryptology (ICISC′11), LNCS 7259, Seoul, Korea, pp. 319-331.

[99] Tritilanunt S., Boyd C., Foo E., Gonzalez J.M. (2007), Toward non-parallelizable client puzzles, 6th International Conference on Cryptology and Network Security, LNCS 4856, Singapore, pp. 247-264.

[100] Saxena N., Chaudhari N.S. (2012), A secure approach for SMS in GSM network, ACM CUBE International IT Conference & Exhibition (CUBE′12), Pune India, pp. 59-64.

[101] Michalas A., Komninos Nikos, Prasad N.R. (2011), Mitigate DoS and DDoS attack in mobile ad hoc networks, International Journal of Digital Crime and Forensics, 3(1), 1-38.

[102] Baden R., Bender A., Spring N., Bhattacharjee B., Starin D. (2009), Persona: an online social network with user-defined privacy, SIGCOMM Conference on Data Communication, Barcelona, Spain, pp. 135-146.

[103] Zhang M., Fang Y. (2005), Security analysis and enhancements of 3GPP authentication and key agreement protocol, IEEE Transactions on Wireless Communications, 4(2), 734-742.

[104] Caimu T., Dapeng O.W. (2008), An efficient mobile authentication scheme for wireless networks, IEEE Transactions on Wireless Communications, 7(4), 1408-1416.

[105] Aiash M., Mapp G., Lasebae A., Phan R. (2010), Providing security in 4G systems: unveiling the challenges, 6^{th} Advanced International Conference in Telecommunications, Barcelona Spain, pp. 439-444.

[106] Park Y., Park T. (2007), A survey of security threats on 4G networks, IEEE Globecom Workshops, Washington USA, pp. 1-6.

[107] TS 29.272 (2010) MME related interfaces based on diameter.
http://www.3gpp.org/DynaReport/29272.htm

[108] ESTI, 3GPP algorithms, 3GPP confidentiality and integrity, document 2: SNOW 3G specification.
http://www.etsi.org/services/security-algorithms/cellular-algorithms

[109] Khan M.A. (2012), Game dynamics and cost of learning in heterogeneous 4G networks, IEEE Journal on Selected Areas in Communications, 30(1), 198-213.

[110] Graziano D. (2013), (Blog) AT&Ts 4G LTE network found to be fastest in US (17 June 2013), www.Bgr.com/2013/06/17/4g-lte-speeds-att-verizon-sprint-t-mobile

[111] Hardawar D. (2014), (Blog) T-Mobile says it now has the fatest LTE network in the US (8 Jan 2014), http://venturebeat.com/2014/01/08/t-mobile-says-it-now-has-the-fastest-u-s-lte-network-adds-1-6m-new-customers-in-q4/

[112] EE 4G UK - details on coverage, phones, prices, speeds for the 4GEE network (4 Dec 2013), www.whathiwi.com/news/everywhere-confirms-ee-4g-network-launch

[113] Double Speed 4GEE, Explore.ee.co.uk/ee-network/4gee/doublespeed-4gee.

[114] Zimmerman P. (1999) An introduction to cryptography, Network Associates Inc., 88 pages. http://www.pgpi.org/doc/pgpintro/

[115] Saxena N., Chaudhari N.S. (2012), Secure encryption with digital signature approach for short message service, World Congress on Information and Communication Technologies (WICT′12), Trivandrum, Kerala, India, pp. 803-806.

[116] Saxena N., Chaudhari N.S., Prajapati G. L. (2012), An extended approach for SMS security using authentication functions, 7^{th} IEEE Conference on Industrial Electronics and Applications (ICIEA′12), Singapore, pp. 663-668.

[117] Saxena N., Chaudhari N.S. (2011), A secure digital signature approach for SMS security, International journal of Computer Application (IJCA), Special issues on IP Multimedia Communications, 98-102.

[118] Java mobile - start learning, http://www.oracle.com/technetwork/java/javame/javamobile/training/emulator/index.html

[119] Saxena N., Chaudhari N.S. (2013), VAS-AKA: An efficient batch verification protocol for value added services, IEEE International Conference on System, Man and Cybernetics (SMC′13), Manchester, UK, pp. 1560-1565.

[120] Khiabani Y., Wei S., Yuan J., Wang J. (2012), Enhancement of secrecy of block ciphered systems by deliberate noise, IEEE Transactions on Information and Forensics Security, 7(5), 1604-1613.

[121] Wei S., Wang J., Yin R., Yuan J. (2013), Trade-off between security and performance in block ciphered systems with erroneous ciphertexts, IEEE Transactions on Information and Forensics Security, 8(4), 636-645.

[122] Biham E. (1998), Design tradeoffs of the AES candidates, Advances in Cryptology (ASIACRYPT′98), LNCS 1514. http://citeseerx.ist.psu.edu/showciting?cid=2739550

[123] Rischpater R. (2009), Beginning Java^{TM} ME platform, Messaging with Wireless API, part 4, pp. 373-407.

[124] Biham E., Shamir A. (1991), Differential cryptanalysis of DES cryptosystems, Journal of Cryptology, 4(1), 3-72.

[125] Choi J., Kim J., Sung J., Lee S., Lim J. (2005), Related-key and meet-in-the-middle attacks on triple-DES and DES-EXE, International Conference on Computational Science and Its Applications (ICCSA$'$05), Singapore, pp. 567-576.

[126] Altman D., Machin D., Bryant T. (2000) Statistics with Confidence Intervals and Statistical Guidelines, BMJ Books, 254 pages.

[127] Venables W.N., Ripley B.D. (2002) Modern Applied Statistics with S, 4^{th} Edition XI, Springer, 497 pages.

[128] Chong H.K. (2012), Improved differential fault analysis on AES key schedule, IEEE Transactions on Information Forensics and Security, 7(1), 41-50.