Secure Algorithms for SAKA Protocol in the GSM Network

Neetesh Saxena
Department of Computing and Informatics
Faculty of Science and Technology
Bournemouth University, UK
Email: nsaxena@bournemouth.ac.uk

Narendra S. Chaudhari
Discipline of CSE,
Indian Institute of Technology, Indore, India and
Visvesvarya National Institute of Technology, Nagpur, India
Email: nsc0183@gmail.com

***Abstract*—This paper deals with the security vulnerabilities of the cryptographic algorithms A3, A8, and A5 existing in the GSM network. We review these algorithms and propose new secure algorithms named NewA3, NewA8, and NewA5 algorithms with respect to the A3, A8, and A5 algorithms. Our NewA5 algorithm is based on block ciphers, but we also propose NewA5 algorithm with Cipher Feedback, Counter, and Output Feedback modes to convert block cipher into stream cipher. However, stream cipher algorithms are slower than the block cipher algorithm. These new algorithms are proposed to use with a secure and efficient authentication and key agreement (AKA) protocol in the GSM network. The proposed architecture is secure against partition attack, narrow pipe attack, collision attack, interleaving attack, and man-in-the-middle attack. The security analysis of the proposed algorithms are discussed with respect to the cryptanalysis, brute force analysis, and operational analysis. We choose the NewA3 and NewA8 algorithms for challenge-response and key generation, respectively. Furthermore, the NewA5 is suitable for encryption as it is efficient than the existing A5/1 and A5/2 algorithms. In case when stream cipher algorithms are required to use, our new algorithms, NewA5-CTR, NewA5-CFB, and NewA5-OFB can be used for specific applications. These algorithms are completely secure and better than the existing A5/1 and A5/2 in terms of resistant to attacks.**

***Index Terms*—GSM, COMP, Collision Attack, Narrow Pipe Attack, Partition Attack.**

TABLE I
SYMBOLS AND ABBREVIATIONS

Symbol	Description	Size (bits)
MS	Mobile Station/User	–
HLR	Home Location Register	–
VLR	Visiting Location Register	–
$IMSI$	International Mobile Subscriber Identity	128
$ServiceRequest$	Location Update Request/Call Attempt	8
$UserProfile$	User Mobile/Reference Number	40
VLR_ID	Identification of the VLR	28
$Count$	Integer Number	24
$\mathbb{K}_c$	Session Key	64
LAI	Location Area Identity	40
VLR_C	Certificate of the VLR	var.
$\mathbb{K}_i$	Secret Key shared b/w MS-HLR	128
$\mathbb{DK}$	Delegation Key	128
$RAND$	Random Number	128
MAC	Message Authentication Code	64
$SRES$	Signed Result	32
T	Timestamp	64
$\|\|$	Concatenation	–

TABLE II
CRYPTOGRAPHIC ALGORITHMS

Function	Definition
$A3/NewA3$	Authentication Algorithm
$A8/NewA8$	Key Agreement Algorithm
$A5/NewA5$	Specified for Data Encryption and Decryption

I. INTRODUCTION

Since in last few years, mobile communications have become more and more popular due to its latest technological advancements. Nowadays, people can communicate with each other from anywhere anytime 24x7. The security measures of first generation (1G) were taken into account during the design of second-generation (2G) digital cellular systems. Even the global system for mobile communication (GSM) network has several security features, but these features have several security and performance limitations ([1], [2], [3], [4]). In particular, the GSM was implemented in more than 70 countries around the world till 2002 ([2]). Global penetration based on total connections is set to exceed 100% in 2013 with mobile subscriber penetration standing at only 45% by the end of 2012, while by 2017, subscriber penetration in developed countries is set to have passed 80% according to [3]. The number of telephone subscribers in India is 1006.96 Million at the end of June, 2015, whereas total wireless subscribers are 980.81 Million, according to [4]. Although we are running towards 3G/4G cellular networks ([20], [21]), Golde et al. in 2013, by performing a practical attacks showed that our basic cellular network, *i.e.*, GSM, is still not secure ([17]). Therefore, it is required to investigate the GSM network security before further moving towards the secure 4G/5G systems.

A. Research Problem

Security algorithms COMP-128, A5/1, and A5/2 used in the GSM network have already been proved vulnerable ([7], [8], [18], [19], [11], [12], [13]). Due to this fact, GSM network provides weaker security to the transmitted information over the network. Even, original GSM authentication and key

agreement (AKA) protocol is inefficient and does not provide secure environment. Hence, there is a strong need to propose and develop a secure and efficient AKA protocol for the GSM network. We have various proposed protocols for the GSM network, which overcome the security issues of the GSM-AKA. According to [16], the SAKA is the most efficient and secure AKA protocol available for the GSM network. Further, the security algorithms are also required to build in order to accommodate sufficient security for the GSM network.

B. Contribution Summary

The following are our contribution towards improving the GSM network security:

- The new algorithms NewA3, NewA8, and NewA5 are proposed and implemented in order to overcome the vulnerabilities existing in the A3/A8 (COMP-128) and A5 algorithms of the GSM network.
- The proposed algorithms are compatible to SAKA protocol, which is one of the existing protocols for the GSM network, and is secure against various attacks, such as man-in-the-middle attack, partitioning attack, collision attack, interleaving attack, and narrow pipe attack ([16]).
- Our analysis shows that the proposed algorithms are more secure as compared to the existing GSM network security algorithms. Further, our NewA5 algorithm is also efficient than A5/1 and A5/2 algorithms.

We propose and implement new GSM algorithms compatible with the SAKA protocol, as this protocol improves the major weaknesses of the GSM network. Table I lists the various symbols with their definitions and sizes, which are used in this paper. Table II represents various functions defined and implemented in this paper. The definitions of the proposed algorithms, *i.e.*, NewA3, NewA8, and NewA5 are same as the A3, A8, and A5 listed in Table II.

C. Organization of the Paper

The rest of paper is organized as follows: Section II reviews the existing GSM security architecture with its security algorithms. Section III describes proposed new algorithms for the GSM network and analyzes their simulation results obtained from the implementation. Further, Section IV discusses the security and performance analysis of new algorithms. Finally, Section V summarizes the conclusion of this work.

II. Review of Existing GSM Security Architecture

The security architecture of the GSM network consists of three algorithms as shown in Figure 1. The first algorithm is A3 that is used for the challenge-response mechanism in order to authenticate the user. The second algorithm is A8 that is used to generate the key through key generation mechanism and uses the output of the A3 algorithm. The third algorithm is A5 that is used to encrypt/decrypt the message information. However, the challenge-response mechanism is a

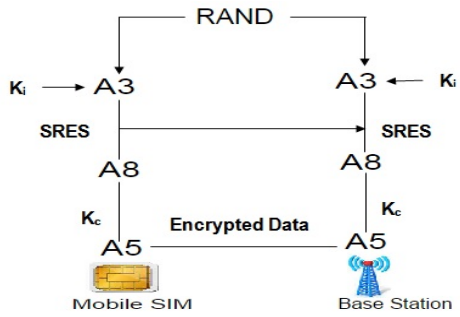

Fig. 1. GSM Authentication and Encryption.

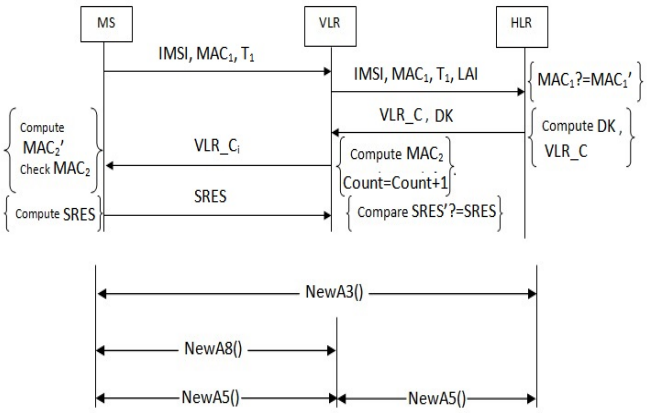

Fig. 2. SAKA Authentication Protocol.

simple authentication mechanism and is insecure ([5]). Most of the mobile operators implement a single COMP-128 keyed hash function instead of using the A3 and A8 algorithms together, which generates the 32-bit signed response (SRES) and secret key 54-bit $\mathbb{K}_c$ in a single function ([10]). The A8 algorithm generates a 64-bit $\mathbb{K}_c$, which is stronger than the $\mathbb{K}_c$ (54-bit) generated by the COMP-128. The authentication of subscriber identity module (SIM) of a user depends upon a shared secret key between a SIM and the authentication center (AuC) called $\mathbb{K}_i$. This secret key $\mathbb{K}_i$ is embedded into a SIM card at the time of manufacturing, and is also replicated at the AuC. When the AuC authenticates a SIM, it generates a random number as RAND and sends it to the subscriber. Both, the AuC and the SIM feed $\mathbb{K}_i$ and RAND values into the COMP-128 and a value known as SRES is generated at both ends. If the value of SRES generated at the SIM card matches with the AuC's SRES, the SIM is successfully authenticated and the user is verified. Thereafter, the AuC as well as the SIM compute a secret key called $\mathbb{K}_c$ by feeding $\mathbb{K}_i$ and RAND values into the A8 algorithm. In 1997, a leaked document led to the publication of the COMP-128, which is used in place of the A3/A8 ([6]).

In 1998, Marc Briceno, Ian Goldberg, and David Wagner

published an attack on COMP-128 with that it is possible to find out the secret key $\mathbb{K}_i$ ([7]). A partition attack was also launched on COMP-128 ([8]). The A5/1, which is deemed strong encryption and used in western countries, was reverse engineered sometimes ago ([9], [10]). Several time-memory trade-off attacks against A5/1 have been proposed. Most notably, the recent attack by Biryukov, Shamir, and Wagner that breaks A5/1 in seconds using huge pre-computation time and memory ([18], [19]). Later, Tim Guneysu et al. described time-memory trade-off technique that can be used for attacking the popular A5/1 algorithm used in the GSM voice encryption ([22]). The A5/2 has been cracked by Wagner and Goldberg with a methodology requiring five clock cycles and making A5/2 almost useless ([11], [12]). Biham's attack is feasible with the current technology, which suggests that the A5/1 scheme should be replaced ([13]). A high speed reconfigurable architecture is proposed, which implements A5/1 and A5/2 algorithms ([14]). Furthermore, a modified version of the A5/1 is also presented. However, it is based on linear feedback shift register (LFSR) ([15]). The cell phones that use GSM, store the data on a SIM card, which often stores upto 64 Kilobytes (KB) or 128 KB of private data.

In the original GSM authentication protocol, an authentication request is sent to the visitor location register (VLR) whenever a mobile user/station (MS) moves into a roaming network and asks for new communication service ([16]). The authentication request includes a temporary mobile subscriber identity (TMSI) and a location area identifier (LAI). On receiving the request, the new VLR uses the received TMSI from the old VLR in order to get an international mobile subscriber identity (IMSI) of the MS and then sends IMSI to the home location register (HLR). Thereafter, the HLR generates n-distinct sets of authenticating parameters $\{\text{SRES}_i, \text{RAND}_i, \mathbb{K}_{c_i}\}$ and sends them to the VLR. Next, the VLR sends selected RAND_i to the MS. Once the MS receives RAND_i from the VLR, it computes SRES' = A3(RAND_i, $\mathbb{K}_i$) and a temporary session key $\mathbb{K}_c$' = A8(RAND_i, $\mathbb{K}_i$), respectively, where $\mathbb{K}_c$' is kept secret for the communication. 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 unsuccessful. Otherwise, the VLR can make sure that the MS is legal ([16]).

Here, we discuss the COMP-128, A5/1, and A5/2 GSM algorithms in brief as follows:

COMP-128 Algorithm: In the GSM network, the A3/A8 algorithm is usually implemented together as a COMP-128 algorithm, which was completely private and is used to generate 32-bit signed response (SRES) and 64-bit cipher key ($\mathbb{K}_c$) ([6]). Various input and output parameters of the COMP-128 algorithm are described as follows:
A3 Input: 128-bit RAND random challenge, $\mathbb{K}_i$ 128-bit key;
A3 Output: 32-bit SRES signed response;
A8 Input: 128-bit RAND random challenge, $\mathbb{K}_i$ 128-bit key;
A8 Output: 64-bit $\mathbb{K}_c$ cipher key is used in A5.
Here, RAND[0...15]: challenge from the base station, key[0...15]: SIM's A3/A8 long-term key $\mathbb{K}_i$, simoutput[0...11]: Output of the SIM, out of which simoutput[0...3] is SRES and simoutput[4...11] is $\mathbb{K}_c$.

Note that $\mathbb{K}_c$ is 74...127 bits of the COMP-128 output followed by 10 zeros. Therefore, the A5 is keyed with only 54 bits of entropy. This represents a deliberate weakness of the key used for the voice privacy.

A5/1 Algorithm: The A5/1 is a stream cipher used in the GSM standard ([18]). Several time-memory trade-off attacks against the A5/1 have been proposed. Most notably the recent attack by Biryukov, Shamir, and Wagner, which can break the A5/1 in seconds using huge pre-computation time and memory ([19]). Later, Tim Guneysu et al. describe time-memory trade-off techniques that can be used for attacking this algorithm used in GSM voice encryption ([22]). The A5/1 stream cipher is a binary linear feedback shift register (LFSR)-based key stream generator. All of the registers used in the A5/1 are first initialized zero. Thereafter, 64-bit secret session key $\mathbb{K}$ and 22-bit frame number F are XOR'ed in parallel into the least significant bits of the three registers. In the next step, all LFSRs are clocked for 100 clock cycles according to majority rule. However, no output is produced during these cycles. Finally, all three LFSRs are clocked according to the majority rule to generate 228 bits of key stream sequence ([**?**]).

A5/2 Algorithm: The A5/1 and A5/2 algorithms were reverse-engineered from a GSM handset and were published by [18]. The A5/2 is built from four LFSRs of lengths 19, 22, 23, and 17 bits denoted by R_1, R_2, R_3, and R_4, respectively ([12]). Clocking of R_1, R_2, and R_3 are controlled by R_4, and R_4 is regularly clocked in each clock cycle. The clock control mechanisms of both, A5/1 and A5/2, depend on the majority rule. However, input parameters to the clocking control mechanism are given from R_4 in case of the A5/2, whereas in the A5/1 the input parameters are from R_1, R_2, and R_3. In each register, majority of two bits and complementary of a third bit are computed. The results of all the majorities and the right most bit from each register are XOR'ed in order to form the output bit ([11]).

III. Proposed Algorithms and Results

In this section, we propose the NewA3, NewA8 and NewA5 algorithms based on a GSM authentication protocol named SAKA ([16]). This protocol is much better in terms of efficiency and bandwidth utilization as well as is secure against various attacks, including man-in-the-middle attack, replay attack, and impersonation attack.

A. SAKA GSM Authentication Protocol

This subsection focuses on a review of the SAKA GSM protocol ([16]), as shown in Figure 2. The NewA5 algorithm is used to encrypt and decrypt the message using $\mathbb{DK}$ key after the authentication is completed.

TABLE III
PERFORMANCE OF THE COMP(A3/A8) AND NEWA3 ALGORITHMS

Parameters	COMP (A3/A8)	$\mathbb{DK}$	NewA3 MAC	VLR_C
Execution Time (ms)	0.04	211.34	232.42	223.99
Heap memory (kbyte)	1007	2047	2047	2047
Total Memory (kbyte)	3529	6245	6245	6244
CPU Time (ms)	468	686.40	686.40	686.40
Swap Space (kbyte)	2027937	2038063	2036162	2051944
Physical Space (kbyte)	1313505	1230012	1219657	1233516
Virtual Memory (kbyte)	46338	44998	45006	46636

Highlights of the SAKA Protocol: The major highlights of the SAKA protocol are summarized as follows:

- Improves the drawbacks of original GSM authentication protocol, including: not supporting mutual authentication, large bandwidth consumption between the VLR and the HLR, storage space overhead at the VLR, and overloaded HLR with authentication tokens at MS.
- Eliminates the need of synchronization between the MS and its home network.
- Generates lower communication overhead as compared to all other existing and proposed GSM protocols.
- On an average, reduces 56% of the bandwidth consumption during authentication process, which is the maximum reduction of bandwidth by a GSM protocol.

We find the SAKA protocol more robust and secure, as it follows the GSM architecture and resolves current existing issues of the GSM network. This paper proposes three new algorithms, named NewA3, NewA8, and NewA5 (secure conversation between the MS and the VLR) to replace A3, A8, and A5 algorithms of the original GSM protocol respectively. In the next subsection, we discuss about new algorithms compatible to the SAKA protocol.

B. COMP(NewA3/NewA8) Algorithm

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

1) NewA3 Algorithm: The NewA3 algorithm is shared between the MS and the HLR only. There are three different cases with different parameters when the NewA3 algorithm is used in the SAKA protocol.

- ***Case 1: Generate $\mathbb{DK}$ key at MS and HLR.***
  Input: 1) T_1 (64 bits): Timestamp, 2) $\mathbb{K}_i$ (128 bits): Secret key shared between the MS and the HLR;
  Output: $\mathbb{DK}$ (128 bits): Delegation key;
- ***Case 2: Generate MAC_1 at MS and HLR.***
  Input: 1) LAI (40 bits): Location area identifier, 2) T_1 (64 bits): Timestamp, 3) $\mathbb{K}_i$ (128 bits): Secret key shared between the MS and the HLR;
  Output: MAC_1 (64 bits): Message authentication code, performs bitwise-XOR between T_1 and LAI (with padding if needed), and the output of it must be used with $\mathbb{K}_i$ key as input to the NewA3 algorithm. Here, LAI must be padded with 24 bits or 3 bytes, *i.e.*, FFF.
- ***Case 3: Generate VLR_C at HLR and MS.***
  Input: 1) MAC_1 (64 bits): Message authentication code, 2) $\mathbb{K}_i$ (128 bits): Secret key shared between the MS and the HLR;
  Output: VLR_C (64 bits): Certificate of the VLR;

We propose a NewA3 algorithm similar to HMACSHA256, which is a secure MAC. The basic structure of the proposed NewA3 algorithm is presented as Algorithm 1.

Algorithm 1 *NewA3 Algorithm*

1: Append zeros (384 zeros) to the left end of key $\mathbb{K}_i$ (128-bit) to create a 512-bit string of $\mathbb{K}_i$.
2: Perform XOR between $\mathbb{K}_i$ (512-bit) and input padding (00110110 repeated 64 times) to produce 512-bit 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) to make it a 512-bit block. This prevents the HMAC against related key-based narrow pipe attack ([23]).
4: Apply hashing to the stream generated in step 3 using an initialization vector (IV) of 256-bit. This process generates a hash code of 256-bit.
5: XOR $\mathbb{K}_i$ (512-bit) with output padding (01011100 repeated 64 times) to produce the 512-bit block S_o.
6: Apply padding to the hash code generated in step 4 with 256-bit (256 zeros) and append the hash code to the S_o.
7: Apply hashing to the stream generated in the step 6 and generate the output 256-bit using IV (256-bit). Now, consider first 128-bit and next 64-bit for the generation of $\mathbb{DK}$ and MAC_1/VLR_C, according to the case 1 and case 2/3, respectively.

Algorithm 2 *NewA8 Algorithm*

1: First three steps are same as in NewA3 algorithm.
2: Apply hashing to the stream generated in step 3 using an initialization vector of 160-bit. This process generates a hash code of 160-bit.
3: XOR $\mathbb{K}_i$ (512-bit) with output padding (01011100 repeated 64 times) to produce the 512-bit block S_o.
4: Apply padding to the hash code generated in step 4 with 160-bit (160 zeros) and append the hash code to the S_o.
5: Apply hashing to the stream generated in the step 6 and generate the output 160-bit using IV (160-bit). Now, consider first 64-bit and next 32-bit for the generation of MAC_2 and SRES, according to case 1 and case 2, respectively.

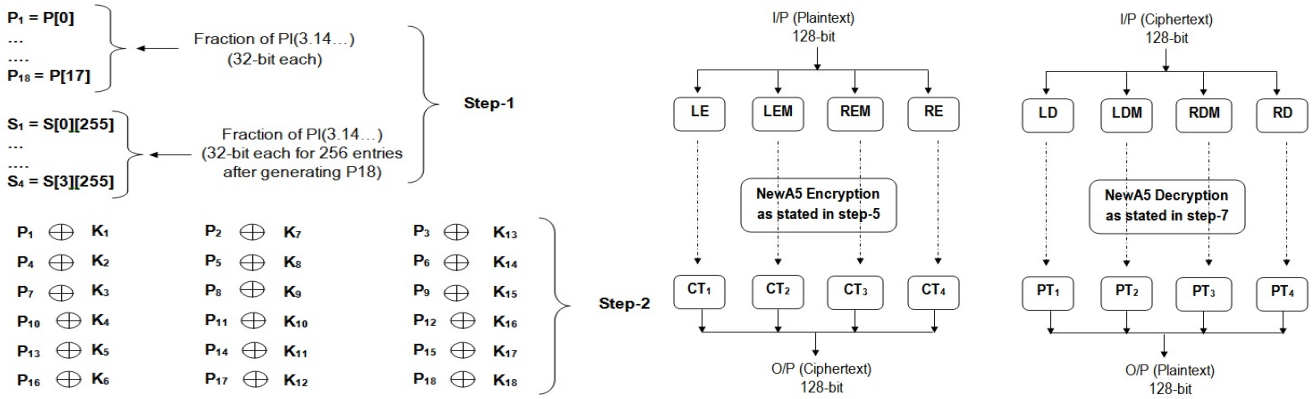

Fig. 3. Various Steps in the NewA5 Algorithm: (a) Initial Steps: 1 to 2, and (b) Final Steps: 3 to 7.

2) NewA8 Algorithm: The NewA8 algorithm is shared between the MS and the VLR and is used in two cases.

- ***Case 1: Generate MAC_2 at VLR and MS.***
  Input: 1) VLR_C (64 bits): Certificate of the VLR, 2) Count (24 bits): Integer number, 3) $\mathbb{DK}$ (128 bits): Delegation key;
  Output: MAC_2 (64 bits): Message authentication code, performs bitwise-XOR between VLR_C and Count (with padding if needed), and the output of it must be used with $\mathbb{DK}$ key as input to the NewA8 algorithm. Here, Count must be padded with 40 bits or 5 bytes, *i.e.*, FFFFF.
- ***Case 2: Generate SRES at MS and VLR.***
  Input: 1) Count (24 bits): Integer number, 2) $\mathbb{DK}$ (128 bits): Delegation key;
  Output: SRES (32 bits): Signed response;

Now, we propose a NewA8 algorithm similar to HMAC-SHA1, which is another MAC. Basic structure of the proposed NewA8 is shown in Algorithm 2.

Algorithms Implementation Results: All the results presented in this paper are obtained on Core i3 processor, 256 MB RAM, 320 GB hard disk and Windows7 operating system with JDK1.6 environment. The performance results of the COMP(A3/A8) algorithm with respect to various parameters have been summarized in Table III. The average time to execute the COMP(A3/A8) algorithm is 0.04 milliseconds (ms), while the process CPU time is 468 ms. The performance of NewA3 and NewA8 algorithms with various parameters can be observed from Table III and Table IV, respectively.

Table III shows the statistics of the output parameters $\mathbb{DK}$, MAC, and VLR_C generated by the NewA3 algorithm, whereas Table IV represents the computation of parameters MAC and SRES, generated by NewA8 algorithm. The average time for the NewA3 algorithm to generate $\mathbb{DK}$, MAC (*i.e.*, MAC_1), and VLR_C is 0.21 seconds (s), 0.23 s, and 0.22 s, respectively. Similarly, the average time for the NewA8 algorithm to generate MAC, *i.e.*, MAC_2 and SRES is 0.19 s and 0.18 s, respectively. The HMAC function provides more security than the normal hash function. The HMAC function uses a secret key that makes it more difficult to vulnerable as compared to the hash function in which no key is used. Therefore, the proposed NewA3 (based on HMACSHA256) and NewA8 (based on HMACSHA1) algorithms are more secure than the COMP(NPA3/NPA8) algorithm with SHA256 and COMP(NPA3/NPA8) with SHA1, respectively ([24]). The execution time of the COMP(A3/A8) is lower than the NewA3 and NewA8. The reason is that the COMP(A3/A8) algorithm is a simple and weak challenge-response mechanism. However, the NewA3 and NewA8 are strong algorithms based on HMAC. It is better to use a bit slow secure algorithms than fast insecure algorithms.

C. NewA5 Algorithm

The NewA5 algorithm is shared among the MS, VLR, and HLR. This algorithm is used to encrypt and decrypt the data being sent 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 base transceiver station (BTS) and vice versa (considering data is formed into blocks, and each block is of 128 bits), 2) $\mathbb{DK}$ (128 bits): Delegation key;
Output: E/D Data (128 bits): Encrypted/decrypted data.

The proposed NewA5 algorithm uses its internal structure similar to Blowfish, which is a symmetric algorithm. Blowfish has a 64-bit block size and is a 16-round Feistel cipher that uses large key-dependent S-boxes. Blowfish makes use of key $\mathbb{K}$ that ranges from 32-bit to 448-bit, which are 1 to 14 32-bit words. This key is used to generate 18 32-bit subkeys, which 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, S and P arrays can be stored rather than re-derived from key each time algorithm is used. This requires over 4 KB of memory. The reason to choose Blowfish as a basis for NewA5 algorithm is that till now there is no full attack on Blowfish. However, previously a key recovery-based attack has been found on AES with partial rounds ([26]). The key generation process in Blowfish takes more time than AES, whereas the encryption and decryption process with Blowfish

Algorithm 3 *NewA5 Algorithm*

1: Initialize 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-bit of hexadecimal digits of PI (3.141...) and so on. The key $\mathbb{DK}$ is used to generate 18 32-bit subkeys.

2: Now perform bitwise-XOR of P[0] with the first 32-bit of the key $\mathbb{DK}$, XOR P[3] with the second 32-bits of the $\mathbb{DK}$ key, and so on for all bits of the $\mathbb{DK}$ key (up to P[17]). Repeatedly cycle through the key bits until the entire P-array has been XOR-ed with the key bits.

3: Encrypt the 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. Encrypt the output of step 3 using the modified P and S arrays. Replace P_3 and P_4 with cipher text output. Continue the process, replacing all elements of P-array and then all four S-boxes in order, with the output of the continuously change in algorithm.

4: Encrypt the given input starting at the given offset and place the result in the provided buffer starting at the given offset. The input will be an exact multiple of our block size. The plain text is divided into 4 32-bit halves LE, LEM, REM, and RE. For each of 16 rounds do:

5: **for** i=1 to 16 **do**
$LEM_i = LE_i \oplus P_i$ and $RE_i = REM_i \oplus P_i$;
$LE_i = F_1[LEM_i] \oplus LEM_{i-1}$ and $REM_i = F_2[RE_i] \oplus RE_{i-1}$;
$LE_{17} = LEM_{16} \oplus P_{18}$ and $REM_{17} = RE_{16} \oplus P_{18}$;
$LEM_{17} = LE_{16} \oplus P_{17}$ and $RE_{17} = REM_{16} \oplus P_{17}$;

6: Decrypt the given input starting at the given offset and place the result in the provided buffer starting at the given offset. The input will be an exact multiple of block size. The cipher text is divided into 4 32-bit halves LD, LDM, RDM, and RD. The decryption involves the use of subkeys in reverse order, however, algorithm direction is same as was in encryption. For each of 16 rounds do the following:

7: **for** i=1 to 16 **do**
$LDM_i = LD_{i-1} \oplus P_{19-i}$ and $RD_i = RDM_{i-1} \oplus P_{19-i}$;
$LD_i = F_1[LDM_i] \oplus LDM_{i-1}$ and $RDM_i = F_2[RD_i] \oplus RD_{i-1}$;
$LD_{17} = LDM_{16} \oplus P_1$ and $RDM_{17} = RD_{16} \oplus P_1$;
$LDM_{17} = LD_{16} \oplus P_2$ and $RD_{17} = RDM_{16} \oplus P_2$;

is faster than AES. Therefore, we propose that all the keys in P[0...17] and S[0...3][0...255] are stored on to the SIM card. This is possible because nowadays we have 128 KB memory-based SIM cards available. We consider the merits of the Blowfish algorithm and develop the NewA5 algorithm, as shown in Figure 3, in which each block size is 128 bits in size. The NewA5 algorithm uses two Feistel cipher F_1 (for 0...63-bit of data block) and F_2 (for 64...127-bit of data block) 16-round each that provides more computational security to the algorithm. The structure of both Feistel ciphers are as follows:

TABLE IV
PERFORMANCE OF THE NEWA8 ALGORITHM

Parameters	MAC	SRES
Avg. Execution Time (ms)	190.23	189.68
Avg. Heap Memory Used (kbyte)	2048	2048
Avg. Total Memory Used (kbyte)	6244	6244
Process CPU Time (ms)	639.60	592.80
Used Swap Space (kbyte)	2031833	1990553
Used Physical Space (kbyte)	1186557	1172545
Virtual Memory (kbyte)	47562	47575

$F_1(x) = ((S_o, 24 + S_1, 16 \bmod 2^{32}) \oplus S_2, 8) + S_3, 0 \bmod 2^{32}$, and $F_2(x) = ((S_o, 28 + S_1, 21 \bmod 2^{32}) \oplus S_2, 14) + S_3, 7 \bmod 2^{32}$. Various steps of the NewA5 algorithm are presented in Algorithm 3.

Algorithms Implementation Results: The performance of A5/1 and A5/2 algorithms based on different parameters can be observed in Table V along with the runtime performance between the NewA5 and original Blowfish algorithms. We can clearly observe from Table V that the NewA5 algorithm is much faster as compared to the Blowfish, A5/1, and A5/2 algorithms. The average time to encrypt and decrypt the message is (0.065, 0.061, 0.7, 0.006) s, and (0.065, 0.061, 0.8, 0.004) s for the A5/1, A5/2, Blowfish, and NewA5 algorithms, respectively. We can clearly observe that NewA5 algorithm provides better encryption and decryption efficiency.

One major application for stream ciphers is to provide high speed data encryption. It means that block cipher-based keystream generators are not the complete solution. Due to convenience use of block ciphers in various protocols, it is suitable to convert it into a stream-like behavior, which can be obtained via modes of operation in Counter, OFB, and CBC. Since the original A5/1 and A5/2 algorithms were stream ciphers in nature, thus we consider the NewA5 algorithm with three modes of operation, *i.e.*, Cipher Feedback (CFB), Counter (CTR) and Output Feedback (OFB) mode to convert the NewA5 algorithm message blocks into bits of streams. Block ciphers encrypt a fixed size block of plaintext at a time to produce a block of cipher text, whereas stream ciphers encrypt stream data one/more bit(s) at a time. Table 6 presents the performance of the NewA5 algorithm with counter mode (NewA5-CTR), NewA5 with CFB mode (NewA5-CFB), and NewA5 with OFB mode (NewA5-OFB). All the parameters generate almost the same results for all three algorithms (as shown in Table VI). The encryption and decryption can be performed in parallel in CTR mode, therefore, we consider the counter mode to integrate with the NewA5 algorithm for providing better security.

IV. DISCUSSION: SECURITY ANALYSIS

This section discusses security analysis of the attacks found on the GSM algorithms as well as impacts on the NewA3, NewA8 and NewA5 algorithms.

TABLE V
PERFORMANCE OF THE A5/1, A5/2, AND NEWA5 ALGORITHMS VS. BLOWFISH ALGORITHM

Parameters	A5/1	A5/2	NewA5	Blowfish
Enc. Time (ms)	0.065	0.061	0.006	0.7
Dec. Time (ms)	0.065	0.061	0.004	0.8
Heap memory (kbyte)	351	351	378	379
Total Memory (kbyte)	12694	12693	12741	12739

TABLE VI
PERFORMANCE OF THE NEWA5-CTR, NEWA5-CFB, AND NEWA5-OFB ALGORITHMS

Parameters	NewA5-CTR	NewA5-CFB	NewA5-OFB
Enc. Time (ms)	0.019	0.018	0.019
Dec. Time (ms)	0.005	0.004	0.005
Heap Memory (kbyte)	385	385	394
Total Memory (kbyte)	14927	14927	14926
CPU Time (s)	624004	608403	639604
Swap Space (kbyte)	1999302	1192441	1981927
Physical Space (kbyte)	1191845	1191440	1170485
Virtual Memory (kbyte)	48377	48414	48369

A. Attacks on Existing GSM Algorithms

This subsection discusses security aspects of the new proposed algorithms against various attacks, which have been found on existing GSM algorithms.

1) Partitioning Attack: In the case of COMP-128, an important characteristic is that the processor can only address 8-bit. However, the first s-box consists of 512 values, which need an address room of 9-bit. Thus, the table needs to split in minimum two pieces. The IBM engineers made the assumption that the table is just split in the middle ([8]). In our scheme, we propose to implement NewA3 and NewA8 algorithms separately unlike as in the existing GSM network, where most operators implement a single joint algorithm, named COMP-128 instead of two algorithms. Since our NewA3 and NewA8 algorithms are based on HMAC functions and do not use S-boxes in the algorithm. Therefore, our algorithms are secure against partitioning attack.

2) Narrow Pipe Attack: A collision attack is possible because 95% of the collisions in the output originate in the second round of the COMP-128, whereas a collision in the first round is impossible, since it is a one-to-one mapping ([27]). With their analysis, 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 COMP128 ([7]). This fact is called narrow pipe. This attack can compromise the HMAC function theoretically only with related key, whereas the hash functions used with HMAC can be compromised with single and related keys ([23]). Since one of the best features of HMAC is that it uses a hash function as a black box, without any need to change the primitive implementation, our goal is to find a patch that does not affect the hash function definition and could prevent HMAC against this attack. Our proposed solution is to force an extra fixed bit (or byte) before the input message M ([23]). To prevent the NewA3 and NewA8 algorithms against narrow pipe attack, we propose to (1) use different initialization vector (IV) values for the inner and outer hash functions in HMAC, (2) append a zero before the message (M'= 0||M) and then append the result to S_i in both algorithms, where each message M' block is about 512-bit in size, and (3) perform $\oplus$ operation with constant 10101010 to the inner and/or outer hash message input to differentiate the input and output computations. We used first two options in our implementation to prevent HMAC-based algorithms against narrow pipe attack, while the third option may increase the computation, therefore not considered.

3) Collision Attack: The aim of this attack is to compute the secret subscriber-specific authentication key. It exploits a weakness in the COMP-128 that allows information to be deduced about the key when two different challenges to the card (input values to the algorithm) produce the same output ([28]). In the SAKA protocol, at each MS, $\mathbb{DK}$ key is generated by passing a unique key $\mathbb{K}_i$ and a timestamp T_1 to a pre-specified algorithm NewA3. Hence, the possibility to generate same output by two MSs is very low. Since the NewA3 and NewA8 algorithms are based on HMAC functions, hence, are free from collision attack, as collision attack does not affect the security of cryptographic hash/HMAC function.

4) Interleaving Attack: In the interleaving attack, an adversary can derive the information from the current or previous authentication exchanges. The input parameters in different cases of the NewA3 and NewA8 algorithms are (T_1, T_1, MAC_1) and (Count, Count), respectively, for each case, where MAC_1 also depends on T_1. These T_1 and Count change each time an authentication request is generated within an expiry time. Hence, if the other parameters are same for NewA3 and NewA8 algorithms, an adversary cannot derive $\mathbb{DK}$ or $\mathbb{K}_i$ key based on previous authentication exchanges. Therefore, these algorithms are secure against interleaving attack.

5) Man-in-the-middle Attack: The man-in-the-middle attack allows an adversary to access secret information by intercepting and altering the communication between the communicating parties. 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. Hence, our system is free from MITM attack.

B. Time Complexity of the New Algorithms

Algorithmic time complexity of an algorithm depends on the operations performed in its various steps. Our NewA3 algorithm performs various internal operations, such as 2 XOR, 6 ||, selection of required bits (128-bit and 64-bit out of 256-bit), and SHA256. The SHA256 executes 80 rounds with AND, OR, XOR, ROT, ADD, and MOD 2^{32} operations. Since the algorithm's input size is fixed, these operations can be performed in O(1). Similarly, the NewA8 algorithm also

has time complexity of O(1) with extra SHIFT operation. The complexity of the NewA5 algorithm is O(1) as the input size is a fixed block size. However, for the NewA5-CTR, NewA5-CFB, and NewA5-OFB algorithms, it is O(m), where m is size of input message.

V. Conclusion

In this paper, the security considerations related to the GSM wireless security have been taken into account. A set of new secure algorithms, *i.e.*, NewA3, NewA8, and NewA5, has been proposed and implemented in order to overcome the known vulnerabilities against A3, A8, and A5 algorithms existing in the GSM network. Three variants of the NewA5 algorithm have also been proposed and implemented, which convert the block ciphers into the stream ciphers, out of which the NewA5 with counter mode provides parallelism to the encryption and decryption process. The algorithms proposed in this paper are cryptographically strong enough to use in place of the algorithms used in the GSM network.

References

[1] Lo, C.C. and Chen, Y.J. (1999) 'Secure communication mechanisms for GSM networks', *IEEE Transaction on Consumer Electronics*, Vol. 45, No. 4, pp.1074–1080.

[2] Saxena, N. and Payal, A. (2011) 'Enhancing security system of short message service for m-commerce in GSM', *IJCSET*, Vol. 2, No. 4, pp.136–133.

[3] Report (2012) GSMA announces new global research that highlights significant growth opportunity for the mobile industry. http://www.gsma.com/newsroom/press-release/gsma-announces-new-global-research-that-highlights-significant-growth-opportunity-for-the-mobile-industry/.

[4] Press Release No. 09/2015 (2015) Highlights on telecom subscription data as on 30 June 2015, Telecom Regulatory Authority of India, New Delhi. http://trai.gov.in/Content/PressDetails/32259_0.aspx.

[5] Lee, C.H., Hwang, M.S., and Yang,W.P. (1999) 'Enhanced privacy and authentication for the global system for mobile communications', *Wireless Networks*, Vol. 5, pp.231–243.

[6] Brumley, B. (2004) 'A3/A8 and COMP128', T-79.514 Special Course on Cryptology, pp.1–18. [Online]. www.tcs.hut.fi/Studies/T-79.514/slides/S5.Brumley-comp128.pdf

[7] Wagner, Goldberg, and Briceno (1998) 'GSM cloning'. [Online]. http://www.isaac.cs.berkeley.edu/isaac/gsm-faq.html.

[8] Rao, J.R., Rohatgi, P., Scherzer, H., and Tinguely, S. (2002) 'Partitioning attacks: or how to rapidly clone some gsm cards', *Proceedings of the IEEE Symposium on Security and Privacy*.

[9] Biryukov, Shamir, Wagner (2000) 'Real time: cryptanalysis of A5/1 on a PC', Weizmann Inst. Rohovot, Isreal.

[10] Wagner, D. (2009) 'GSM cloning', Smartcard Developer Association and ISAAC security research group. [Online]. www.scard.org/gsm/.

[11] Barkan, E.L., Biham, E., and Keller, N. (2003) 'Instant cipher text only cryptanalysis of GSM encrypted communication', *Proceedings of CRYPTO*, LNCS 2729, pp.600–616.

[12] Briceno, M., Goldberg, I., and Wager, D. (1999) 'A pedagogical implementation of the GSM A5/1 and A5/2 voice privacy encryption algorithms'. [Online]. www.cryptome.org/gsm-a512.htm

[13] Biham, E. and Dunkelman, D. (2000) 'Cryptanalysis of the A5/1 GSM stream cipher', *Proc. of INDOCRYPT*, 1977, pp.43–51.

[14] Longmei, L.W.D.Z.N. (2008) 'Research and implementation of a high-speed reconfigurable A5 algorithm', *Workshop Computational Intelligence and Industrial Application*, pp.93–98.

[15] Bajaj, N. (2011) 'Enhancement of A5/1 using variable feedback polynomials of LFSR', *Proceedings of the ETNCC*, pp.55–60.

[16] Saxena, N. and Chaudhari, N.S. (2013) 'SAKA: a secure authentication and key agreement protocol for GSM network', *CSI Transactions on ICT*, Vol. 1, No. 3, pp.1–13.

[17] Golde, N., Redon, K., and Seifert, J.P. (2013) 'Let me answer that for you: exploiting broadcast information in cellular networks', *Proceedings of USENIX conf. on Security*, pp.33–48.

[18] Briceno, M., Goldberg, I., and Wagner, D. (1999) 'A Pedagogical implementation of the A5/1', The Smartcard Developer Association (SDA). [Online]. http://www.scard.org/gsm/a51.html.

[19] Ekdahl, P. and Johansson, T. (2003) 'Another attack on A5/1', *IEEE Trans. on Information Theory*, Vol. 49, No. 1, pp.284–289.

[20] Saxena, N. and Chaudhari, N.S. (2014) 'Secure-AKA: an efficient AKA protocol for UMTS networks', *Wireless Personal Communications*, Vol. 78, pp.1345–1373.

[21] Saxena, N., Thomas, J., and Chaudhari, N.S. (2015) 'ES-AKA: an efficient and secure authentication and key agreement protocol for UMTS networks', *Wireless Personal Communications*, Vol. 84, No. 3, pp.1981–2012.

[22] Guneysu, T., Kasper, T., Novotny, M., Paar, C., and Rupp, A. (2008) 'Cryptanalysis with COPACOBANA', *IEEE Trans. on Computers*, Vol. 57, No. 11, pp.1498–1513.

[23] Peyrin, T., Sasaki, Y., and Wang, L. (2012) 'Generic related-key attacks for HMAC', *Proceedings of ASIACRYPT*, pp.580–597.

[24] Saxena, N. and Chaudhari, N.S. (2013) 'An enhanced NPA protocol for secure communications in GSM network', *International Journal of Security and Networks (IJSN)*, Vol. 8, No. 1, pp.13–28.

[25] Stallings, W. (2011) 'Cryptography and Network Security', Pearson Education, 5th Edition.

[26] Bogdanov, A.,and Rechberger, C. (2011) 'Biclique cryptanalysis of the full AES', *Proceedings of ASIACRYPT*, pp.344–371.

[27] Handschuh, H. and Paillier, P. (2000) 'Reducing the collision probability of alleged COMP128', *Smart Card Research and Applications*, LNCS 1820, pp.380–385.

[28] Walker, M. and Vedder, K. (1998) 'Attack on COMP 128', ETSI SMG#26, Tdoc SMG 98-0475. quintillion.co.jp/3GPP/GSM/SMG_26/tdocs/P-98-0475.pdf.

[29] Gonzalez, T. (2007) 'A reflection attack on blowfish'. [Online]. http://karbalus.free.fr/sat/docsat/PaperGonzalezTom.pdf.

[30] Mitchell, C.J. and Dent A.W. (2010) 'International standards for stream ciphers: A progress report'. [Online]. http://www.chrismitchell.net/isfsca.pdf.