BVPSMS: A Batch Verification Protocol for End-to-End Secure SMS for Mobile Users

Neetesh Saxena *Member, IEEE,* Hong Shen *Member, IEEE,* Nikos Komninos *Member, IEEE,* Kim-Kwang Raymond Choo *Senior Member, IEEE,* and Narendra S. Chaudhari *Senior Member, IEEE*

Abstract—Short Message Service (*SMS*) is a widely used communication medium, including by mobile applications, such as banking, social networking, and e-commerce. Applications of *SMS* services also include real-time broadcasting messages, such as notification of natural disasters (e.g. bushfires and hurricane) and terrorist attacks, and sharing the current whereabouts to a group of friends, such as notifying urgent business meeting information, transmitting quick information in the battlefield to multiple users, notifying current location to our friends, and sharing market information. However, traditional *SMS* is not designed with security in mind (*e.g.* messages are not securely sent). It is also possible to extract International Mobile Subscriber Identity (*IMSI*) of the mobile user. In literature, there is no known protocol that could enable secure transmission of *SMS* from one user to multiple users simultaneously. In this paper, we introduce a batch verification Authentication and Key Agreement (*AKA*) protocol, *BVPSMS*, which provides end-to-end message security over an insecure communication channel between different Mobile Subscribers (*MS*s). Specifically, the proposed protocol securely transmits *SMS* from one *MS* to multiple *MS* simultaneously. The reliability of the protocol is discussed along with an algorithm to detect malicious user request in a batch. We then evaluate the performance of the *BVPSMS* protocol in terms of communication and computation overheads, protocol execution time, and batch and re-batch verification times. The impacts of the user mobility, and the time, space, and cost complexity analysis are also discussed. We also present a formal proof of the security of the proposed protocol. To the best of our knowledge, this is the first provably-secure batch verification *AKA* protocol, which provides end-to-end security to the *SMS* using symmetric keys.

Index Terms—Authentication, Batch Verification, Mobile Subscriber, SMS, Symmetric Key Cryptosystem.

1 Introduction

Cellular and mobile telecommunication industries are one of the fastest growing industries globally, partly due to the capability to provide a wide range of services to the Mobile Subscribers (*MS*s), such as health surveillance [1], health financing and health worker performance [2], and Short Message Service (*SMS*)-based web search [3]. However, the challenge for the server to handle multiple authentication requests at one time or in a very short time period (*e.g.* during the first few minutes of a major incident, such as a natural disaster or terrorist attack) is an area that has attracted the attention of researchers in recent years.

- *N. Saxena is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, USA. E-mail: mr.neetesh.saxena@ieee.org*
- *H. Shen is with the Department of Computer Science, University of Adelaide, Australia. E-mail: hong.shen@adelaide.edu.au*
- *N. Komninos is with the Department of Computer Science, City University London, UK. E-mail: nikos.komninos.1@city.ac.uk*
- *K.-K. R. Choo is with the School of Information Technology and Mathematical Sciences, University of South Australia, Australia, and School of Computer Science, China University of Geosciences, Wuhan, China. E-mail: raymond.choo@fulbrightmail.org*
- *N. S. Chaudhari is with the Discipline of Computer Science and Engineering, Indian Institute of Technology, Indore, India and also with the Visvesvarya National Institute of Technology, Nagpur, India. E-mail: narendra@vnit.ac.in*

Manuscript received XXXX XX, 20XX; revised XXX XX, 20XX.

1.1 Research Problem

When an *SMS* is sent from one *MS* to another, the information contained in the *SMS* is transmitted as plaintext. *SMS* may also contain confidential information such as PIN number and a link to a login page. Transmission of such confidential information as plaintext over an insecure network can be targeted by an adversary (*e.g.* intercepting, reading and modifying the *SMS* before it reaches the *SMS*-Center (*SMSC*). Traditional *SMS* service does not have a mechanism to transmit the message securely from one *MS* to another *MS* or to a group of *MS*s. The *EasySMS* is the only protocol available in the literature that enables secure transmission of *SMS* from one *MS* to another [4]. However, there is no such protocol exists in the literature that can securely delivers an *SMS* to multiple recipients simultaneously. This is surprising, as in our increasingly interconnected society, there are various situations where secure transmission of batch *SMS* can play a crucial role, such as sending urgent business meeting information to the employees or to the members of the political parties, military services like simultaneous and quick transmission of secure information in the battlefield, notifying current location to our friends or family members when a person is in trouble (especially helpful for girls), sharing market information, crowd-sourcing information, human flesh search engine of notifying other users about a corrupted public servant by secure *SMS*, and in some cases life-saving (*e.g.* notifying residents in remote areas of a fast spreading bushfire, an earthquake or a volcano eruption, or notifying all residents and users in the vicinity of an area to stay indoor due to an ongoing terrorist attack). In many of these applications, we should not compromise on security for the capability for batch dissemination of *SMS*. For example, without an end-to-end (batch) *SMS* security mechanism in place, a malicious attacker (*e.g.* hacktivist or ideologically-motivated individual) could hijack and replace a batch *SMS* from the local authorities with one that will create social unrest (*e.g.* messages inciting racial hatred). In addition, the protocol should be sufficiently lightweight, suitable for deployment on resource-constrained devices (e.g. limited battery) [5].

1.2 Existing Solutions

Several batch verification-based solutions have been designed for different applications. For example, a number of protocols have been proposed for the value added services in Vehicular Ad-hoc Networks (*VANET*) [6], [7], [8], public-private key-based vehicular communication system [9], [10], [11], and digital signatures in batch to achieve high efficiency [12], [13]. Several *SMS*-based wireless protocols [14], [15], [16], [17], lightweight *AKA* [18] and *SMS*-based attacks and their countermeasures are discussed in [19]. Also, the protocols in [20], [21], [22], [23] are designed to provide *SMS* security based on asymmetric key cryptography with the exception in [22]. Other protocols in the literature include [24], [25] designed for the Global System for Mobile Communications (*GSM*), [26], [27], [28], [29], [30] for the Universal Mobile Telecommunications System (*UMTS*), and [31], [32] for the Long-Term Evolution (*LTE*) networks. However, all these protocols do not consider simultaneous multiple authentication requests using *SMS*. Group Authentication and Key Agreement (*AKA*) protocols are also available in the *LTE* network [33], [34]. However, these protocols do not consider *SMS* as a communication medium and require additional cost and storage for a group setup. Recently, a solution for user privacy in mobile telephony was proposed using the predefined multiple International Mobile Subscriber Identities (*IMSI*) for each Universal Subscriber Identity Module (*USIM*) [35]. However, this solution requires a large storage space, generates a huge overhead for pseudo-identities, and utilizes significant bandwidth for sending *IMSI*s to each *MS*.

A literature review suggests that there is no known batch verification-based protocol that provides end-to-end *SMS* security to many *MS*, although we observe that commercially available applications, such as *SMSzipper*, *TextSecure*, *moGile Secure SMS*, and *CryptoSMS* provide the facility to send secure *SMS*. However, there are a number of limitations in these software solutions, such as (i) the need to install them on the phone's memory/memory card, (ii) the need to provide a secret key to the *SMS* recipient, and (iii) the inability to support sending of an *SMS* to many users simultaneously. Moreover, the security of the communications may also be affected by malware installed or vulnerabilities on the client devices. Therefore, a preferred solution is to develop a protocol that provides end-to-end security.

1.3 Our Contribution

In this paper, we propose a secure and efficient batch verification-based *AKA* protocol, hereafter referred to as *BVPSMS*, which enables the transmission of an *SMS* to multiple recipients at any one time. *BVPSMS* uses symmetric keys, since symmetric key encryptions are significantly faster than asymmetric key encryptions. The proposed protocol has the following contributions:

1) The *BVPSMS* protocol:
   - provides mutual authentications between the sender *MS* and the Authentication Server (*AS*), and between each recipient MS_i and the *AS*.
   - maintains message confidentiality and integrity using *AES-CTR* and Message Authentication Code (*MAC*), respectively, during the messages transmission over an insecure network.
   - allows the sending of only one of n-pieces of the secret code of the key by sender *MS* to each recipient *MS*. It has the following advantages: (i) sending a partial code to each recipient *MS* improves the overall security of the system, and (ii) reduces the total communication overhead generated by the protocol.
2) Our protocol is secure against replay attack, Man-in-the-Middle (*MITM*) attack, impersonation attack, *SMS* disclosure, and *SMS* spoofing.
3) Each user's original identity is kept secret during the authentication over the network. It protects the user against *IMSI* tracing and *ID*-theft attacks.

We compare our protocol with four other related protocols (*ABAKA*, *RAISE*, *SPECS*, and *b-SPECS+*). In a batch authentication when number of requests are 5, 10, 20, 50, 100, and the findings are as follows:

1) During first time (fresh) authentication, *i.e.*, *BVPSMS**, reduces 6.1%, 23%, 12.5%, and 46.52% of the communication overhead as compared to *ABAKA*, *RAISE*, *SPECS*, and *b-SPECS+*, respectively, and is equal of the *BLS* protocol. However, *BLS* does not provide mutual authentication, user privacy, integrity protection, and offers only partial resilience to impersonation attack.
2) During each subsequent authentication, *i.e.*, *BVPSMS***, lowers the communication bandwidth by 79.27%, 89.83%, 80.69%, and 88.2% in comparison to *ABAKA*, *RAISE*, *SPECS*, and *b-SPECS+*, respectively.

In addition, findings from the simulations (*i.e.* execution time, verification time, and re-batch verification time) demonstrate the utility of our protocol in a real-world cellular network deployment.

The remainder of the paper is organized as follows. Section 2 describes the system and threat models for *SMS* security. Section 3 presents our proposed protocol. Section 4 presents the reliability analysis of the proposed protocol, a malicious request detection algorithm, and the impact on user mobility. The security analysis and the performance evaluation of the *BVPSMS* protocol are presented in Sections

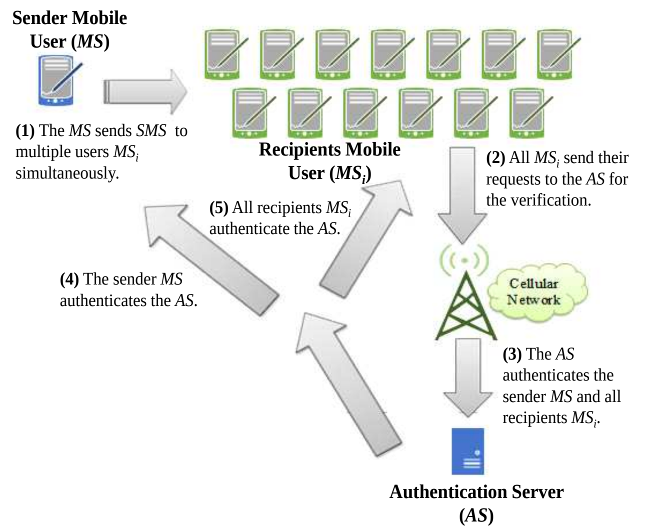


Fig. 1: Batch authentication requests from the *MS* to the *AS*.

5 and 6, respectively. Formal proofs of *BVPSMS* using *BAN-Logic* and *Proverif* are outlined in Section 7. Finally, section 8 concludes this work.

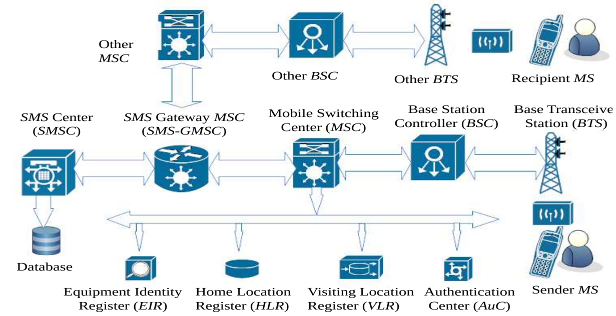
2 System and Threat Models

In this section, we present the system and threat models.

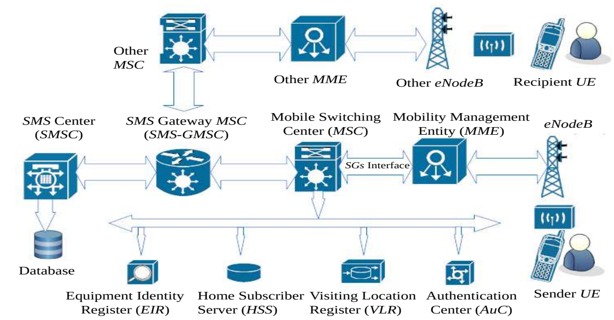
2.1 System Model

We introduce a scenario where the *MS* sends an *SMS* to multiple *MS*s simultaneously. Upon receiving the *SMS*, each *MS* sends its authentication request to the *AS* for identity verification of the sender *MS*. The system model allows many such concurrent executions (*e.g.* several *MS* sending *SMS* to multiple recipients *MS*. A scenario is shown in Figure 1 where multiple *MS*s send their authentication requests to the *AS* for the identity verification of sender *MS* at the same time. The *AS* handles the received authentication requests and authenticates all the *MS*s. The authentication request may be single or multiple. However, it would be uncommon to have only a single request at any point of time. When an *SMS* is sent from the sender *MS* to the recipient *MS* over the *2G/3G* (*GSM/UMTS*) networks, it follows the path shown in Figure 2(a) [36], [37]: Sender *MS*→Base Transceiver Station (*BTS*)→Base Station Controller (*BSC*)→Mobile Switching Center (*MSC*)→*SMS*-Gateway *MSC* (*SMS-GMSC*)→*SMS*-Center (*SMSC*)→*SMS-GMSC*→*MSC*→*BSC*→*BTS*→Recipient *MS*. Similarly, Figure 2(b) and Figure 2(c) show a path of *SMS* transmission over the *SGs* and *IP/IMS* in *4G* (*LTE*) networks. It is challenging for the *AS* to verify and authenticate a large number of *MS*s, based on its capacity to handle requests in an efficient way.

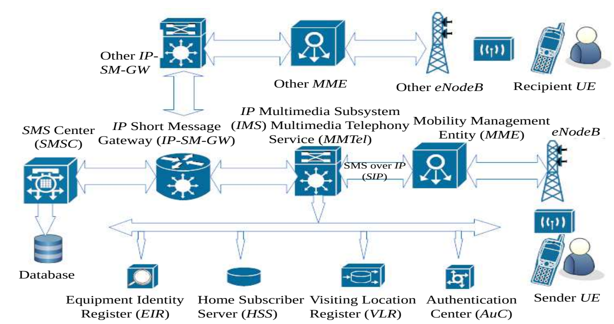
If the server can only handle one request at a time, then it requires a queue to manage all incoming requests. However, managing such a queue will result in increased overheads, time, and cost of authentication. In fact, the approach used for the authentication must be very efficient to handle all the requests in a very short time. To more efficiently handle multiple authentication requests, one solution is to perform a batch authentication for all incoming requests. However, there may be one or more malicious requests generated by the adversary. In such a case, we need to first identify the malicious requests and remove the identified malicious requests from the batch, then perform re-batch authentication. This comes at an additional cost to the re-batch authentication. However, the cost of authenticating each user is reduced. The notations used in the paper are presented in Table 1.



(a) *SMS* transmission in *2G/3G* (*GSM/UMTS*) system.



(b) *SMS* transmission over *SGs* in *4G* (*LTE*) system.



(c) *SMS* transmission over *IP/IMS* in *4G* (*LTE*) system.

Fig. 2: *SMS* transmission from the sender *MS*/*UE* to receiver *MS*/*UE* in cellular systems.

2.2 Threat Model

We consider a threat model with three categories of the mobile users, namely honest majority, semi-honest majority, and dishonest majority. In the *honest majority* scenario, the legitimate and honest *MS* and the *AS* behaves as per protocol specifications, while a few (no more than half the total)

TABLE 1: Notations

Symbol	Description	Size (bits)
MS	Mobile station referring user	–
UE	User Equipment referring user	–
AS	Authentication server referring *AuC*	–
IMSI	International mobile subscriber identity	128
TID	Temporary identity	128
ReqNo	Request number	8
SK	Shared secret key between *MS* and *AS*	128
DK	Delegation key generated from *SK*	128
H/MAC	Hash/message authentication code	64
$T/T_i/T_1$	Timestamp	64
K	Random number	128
$Y/P/Q/R$	Variable	128
Z	Signature generated by the *MS*	128
SIMcode	SIM card activation code of *SK* key	64
S-Actcode	Sender generated code of the *SK* key	64
Actcode	Recipient generated code of the *SK* key	64
ExpT	Expiry time	64
$f_1()$	*HMACSHA256* is used to generate *DK*	–
$f_2()$	*AES-CTR* is used to generate *TID*	–
$f_3()$	*HMACSHA1* is used to generate *MAC*	–
$E\{\}_{DK}$	Encryption function with *DK* key	–
$D\{\}_{DK}$	Decryption function with *DK* key	–
$\oplus$	Bitwise *XOR* operation	–

MS send incorrect outputs to the *AS* in a *semi-honest MS* scenario. However, in the *dishonest MS* (malicious *MS* to the network) scenario, majority of the *MS* (more than half) send fabricated information to the *AS*. Furthermore, malicious *MS* computes the required functions in a probabilistic polynomial time with auxiliary information. We do not consider these scenarios for the *AS*, as malicious *AS* does not have the correct keys in its database. Therefore, we consider only the trusted *AS* scenario where the *AS* always sends correct information to all the *MS*s. We also remark that an adversary can delay some or all the messages between the *MS* and the *AS* under a public channel.

In this paper, we consider two variations of the adversary models: non-adaptive and adaptive variations. In a non-adaptive or static variation of the model, a set of corrupted users are fixed, while in the adaptive variant, the adversary can choose any corrupted users in any numbers during run time. Furthermore, the adversary can choose any input for corrupted users. We also consider *passive as well as active adversaries* in the network.

i) Security and Privacy Attacks, and Integrity Violations: The threat model describes different scenarios to capture various attacks in which a malicious *MS* can access the authentic information or misguide legitimate *MS*. Since the *SMS* is sent in plaintext, network operators can eavesdrop on the *SMS* content at the *SMSC*. This leads to *SMS* disclosure and spoofing attacks. Currently, Over-the-Air (*OTA*) interface between the *MS* and the *BTS* is protected by a weak encryption algorithm. Hence, the adversary can compromise the messages in order to capture the information contained in the *SMS*. The unencrypted messages are sent over the Signaling System (*SS7*) networks, which does not secure the transmission medium.

ii) Security Goals: Our security goals are as follows:

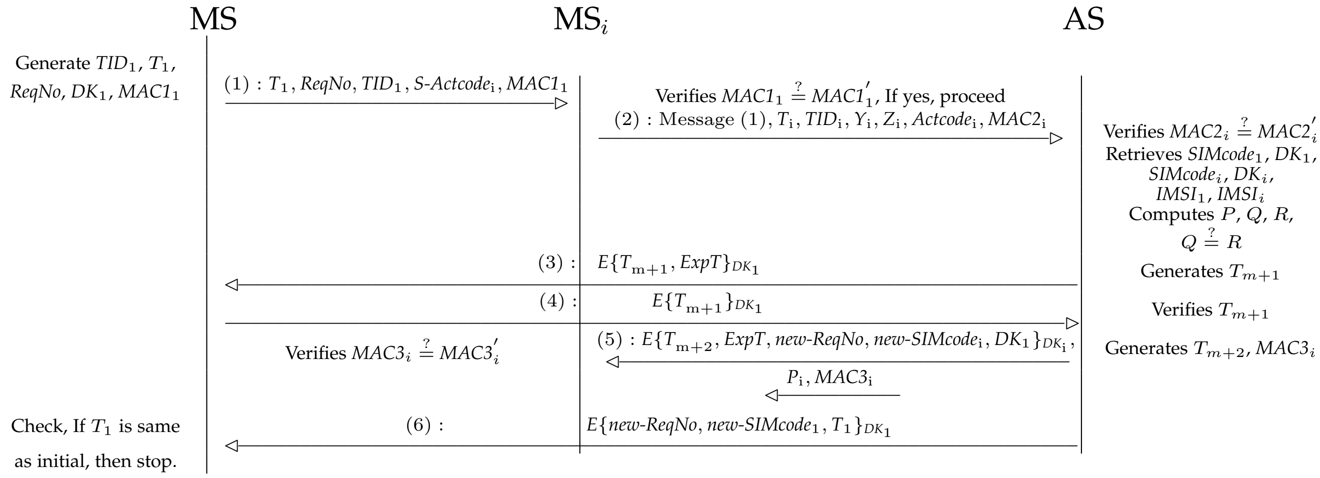
1) *Mutual Authentication:* The proposed protocol must provide mutual authentication between each *MS* and the *AS*.
2) *Data Confidentiality and Message Integrity:* These are two key properties to prevent the leakage or abuse of user data.
3) *Other Security Properties:* The protocol should be secured against the following attacks:
    a) *Eavesdropping and Impersonation Attacks:* The adversary can eavesdrop the communication between the user and the server. The adversary may also pretend itself as legitimate user or the server and perform impersonation attacks.
    b) *MITM Attacks:* An adversary can perform *MITM* attack when the *MS* is connected to the *BTS* and eavesdrops the session initiated by a legitimate *MS*. If *IMSI* is sent in cleartext, the adversary can compromise the system/user by tracing the user. Commercially available software, such as *IMSI* catcher can be used to capture the user's *IMSI* over a weak or unencrypted network.
    c) *Replay Attacks:* The attacker may fraudulently delay the conversation between both *MS*, and captures or reuses the authenticated information contained in the previous messages to facilitate or conduct a replay attack.
4) *Session Key Security, Forward Secrecy, and Non-linkability:* It is common practice not to send the session key over the network in a plaintext. The system must also defeat *known key* attacks and maintain forward secrecy. The protocol should be able to handle key generation, transmission, and its usage. The adversary must not be able to link current session information (messages and keys) with previous sessions, *i.e.*, *non-linkability*.
5) *Privacy Preservation and Untraceability:* The original identity of each *MS* must be protected during its transmission over the network. Such *privacy preservation* helps to secure the system against *MITM* attacks and user *untreaceability*.

3 Proposed Protocol: BVPSMS

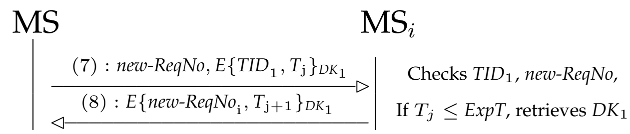
In this section, we present the proposed efficient and secure batch verification-based protocol *BVPSMS* for end-to-end *SMS* security over an insecure network. The *BVPSMS* protocol is illustrated in Figure 3. The following subsections describe our protocol in detail.

3.1 System Assumptions

We make the following assumptions, similar to the traditional cellular network, for our system implementation:
Assumption 1. An *AS* is deployed at the Authentication Center (*AuC*) similar to the traditional cellular network.
Assumption 2. A Secret Key (*SK*) is stored in the *AS*'s database at the *AuC* as well as on the Subscriber Identity Module (*SIM*) card of the *MS* during manufacturing.



(a) Phase-1: protocol execution for mutual authentications.



(b) Phase-2: subsequent authentications.

Fig. 3: BVPSMS protocol (a) phase-1 (b) phase-2.

Assumption 3. The *AS* never discloses the stored secret keys to any other entity in the network. Also, it does not illegally reuse the secret key of any one mobile user to other users.

Assumption 4. The process of generating *Actcode* and retrieval of *SIMcode* (discussed later in user registration subsection) is strictly kept secret and not publicly available. This is a realistic assumption as cellular network algorithms and functions are generally considered intellectual property.

3.2 Definition of the Functions Used

Our protocol uses different functions with standard notations, such as $f_1()$, $f_2()$, $f_3()$, and $E/D\{\}$, similar to used by existing cellular network authentication protocols. In the protocol, $f_1()$ and $f_3()$ functions are two different *HMAC* functions to avoid any collision generated with the same input. We also consider *AES* with Counter mode (*AES-CTR*) to implement $f_2()$ and $E/D\{\}$. However, inputs for $f_2()$ and $E/D\{\}$ are different. Modern mobile devices are fairly capable of computing these functions [4]. The structure of these functions as follows:

$f_1()$ ***Function:*** A one-way function, such as one-way hash function *HMACSHA256*, which takes input message of 512 bits with *SK* key and generates 256 bits of hash code, out of which first 128 bits are used as the *DK* key.

$f_2()$ ***Function:*** Any reversible symmetric encryption function, such as *AES-CTR* where the plaintext and shared key generate the ciphertext, and then ciphertext and the same key are able to produce the original plaintext. The key used in the function is *DK*, derived from the *SK* key at *MS* as well as at *AS*.

$f_3()$ ***Function:*** It is used to generate *MAC* codes, which can be implemented by a one-way *MAC* function, such as *HMACSHA1* that takes as input a multiple of 512 bits message with *DK* key and generates 160 bits of hash code, where the first 64 bits are used as *MAC*.

$E/D\{\}_{DK}$ ***Function:*** It is used to encrypt and decrypt the transmitted messages over the network. *AES-CTR* with *DK* key is used for this purpose. The Modified *AES* (*MAES*) [4] with 256 bits of *DK* key can also be used as an alternative. However, a key expand function is required to generate 256 bits of *DK* key from 128 bits.

3.3 Detailed Description

Although the protocol is capable of supporting concurrent threads of different *MS*s sending their authentication requests with *SMS*s to several different *MS*s. For simplicity, in this section, we present a scenario where a *MS* sends multiple *SMS*s to different *MS*s. This scenario can be easily extended with multiple sender *MS*s. The physical security (any personal access by the end user/mobile operator/adversary) of the *AS* is assumed secure, similar to the existing traditional cellular networks. Hence, it is almost impossible to extract the secret key *SK* of a mobile user. Readers should not confuse the *AuC* with the *SMSC*. At the *SMSC*, mobile operator can easily access the content of each message. The *AuC* is secured against any personal access, and the keys stored at the *AuC* can only be accessed by the protocol during its execution. Therefore, the *AS* is secure against any personal access.

We describe our protocol in four different parts: user registration, pseudo-identity generation, protocol initialization, and protocol execution. The protocol maintains message integrity between each *MS* and the *AS* using *MAC*s.

1) User Registration: When a user requests for a new

SIM, the operator activates *SIM* card by establishing a connection between the *SIM* card and the *AS*. The *AS* generates a random *SIMcode* $\in \mathbb{Z}_p^*$ (where p is a large prime), stores *SIMcode* in its database as a label to the secret key *SK*, and also sends *SIMcode* to the *SIM* card during first use (*e.g.* when the card is activated). On receiving *SIMcode*, the *SIM* card stores it in the memory. The *Actcode* is a one-time activation code sent to the *AS* instead of the actual *SIMcode*, when requesting for the authentication. The purpose of this code is to help the *AS* to verify *SIMcode* and retrieve *SK* key from its database that belongs to a user requesting for the authentication. The *AS* sends a random *new-SIMcode* to all involved *MS*s for subsequent authentication request.

In the proposed protocol, when a mobile user activates this module to send an *SMS* to multiple users, an automatic signal is sent to the respective *AS*, which sends a random k to the user's device encrypted by its *DK* key. The user decrypts k and chooses n by its own. The selection of n is based on the average number of *SMS*s dropped by the network per unit time. Although there is no guarantee that an *SMS* will actually be delivered to the recipient, but delay or complete loss of a message is uncommon, typically affecting less than 5 percent of messages [38]. Hence, our scheme uses n is at least $1.05 \times k$. The need to generate and transmit activation code *S-Actcode* by the sender *MS* and retrieval of actual *SIMcode* by the *AS* is motivated by Shamir secret [39] as follows:

The goal is to divide the hash of secret *SIMcode* of the sender *MS* into n-pieces as $\{$*S-Actcode*$_1$, *S-Actcode*$_2$, ..., *S-Actcode*$_i\}$ such that: (i) knowledge of at least k pieces of *S-Actcode*$_i$ helps *AS* in the computation required to generate the final *SIMcode*, say *SIMcode*$_1$'s hash, and (ii) knowledge of any k-1 pieces of *S-Actcode*$_i$ cannot help in the reconstruction of the final *SIMcode*$_1$'s hash (considering all possible values are equally likely). Therefore, the sender *MS* sends *S-Actcode*$_i$ to n-recipients MS_i. All n-MS_i (in the ideal case) or at least k out of n-recipients MS_i (in case of error or network failure) forward their *Actcode*$_i$ to the *AS* along with the received *S-Actcode*$_i$ (part of sender *MS*). The *AS* obtains the actual hashed *SIMcode*$_1$ after receiving at least k-*S-Actcode*$_i$. The *AS* will then match the computed hashed *SIMcode*$_1$ with the stored pre-computed hash of *SIMcode*$_1$ of the sender *MS*. Once the hashed *SIMcode*$_1$ is known to the *AS*, it retrieves SK_1 key and derives a delegation key DK_1 of the sender *MS*. This entire process takes k points to define a polynomial of degree k-1 in a finite field $\mathbb{F}$ of size p where $0 < k \leq n < p$, *SIMcode*$_1 < p$, and p is a large prime.

The sender *MS* chooses at random k-1 positive integers $\{b_1, b_2, ..., b_{k-1}\}$ with $b_i < p$, and computes a polynomial $f(x) = b_0 + b_1 x + b_2 x^2 + ... + b_{k-1} x^{k-1}$, where b_0 = *SIMcode*$_1$. The sender *MS* generates n *S-Actcode*$_i$ points (x_i, y_i) as $(i, f(i)\ mod\ q)$ using the Lagrange basis polynomial, where $q > n$, $q > b_i$. On receiving the message (from at least k-recipients MS_i), the *AS* reconstructs a polynomial by computing $f(x)$ as:

$$f(x) = \sum_{i=1}^{k} y_i l_i(x), \text{ where } l_i(x) = \prod_{\substack{1 \leq j \leq k \\ j \neq i}} (x - x_j)/(x_i - x_j).$$

Finally, the *AS* retrieves the actual *SIMcode*$_1$ (= b_0) from the computed $f(x)$. In our protocol, each recipient MS_i also generates its own *Actcode*$_i$ as follows:

At the MS_i*:* Each MS_i generates *Actcode*$_i$ = $H($*SIMcode*$_i)$ and is sent to the *AS*. We use first 64 bits of $H()$ function as *Actcode*$_i$, which is *SHA256*.

At the AS: The *AS* pre-computes $H($*SIMcode*$_i)$ from the stored *SIMcode*$_i$ for each MS_i, and then verifies *Actcode*$_i \stackrel{?}{=} H($*SIMcode*$_i)$. Thereafter, the *AS* extracts SK_i key, and derives DK_i key by referring *SIMcode*$_i$ of each MS_i.

We keep the selection of k points dynamic by the *AS* in each attempt to increase the difficulty of an adversary in correctly guessing the different pieces of the secret *SIMcode*$_1$. Also, in each such request, n is randomly generated, which is is at least $1.05 \times k$. For example, we can divide the hash of secret *SIMcode*$_1$ into twenty parts (n = 20) of *S-Actcode*$_i$, and any fifteen parts (k = 15) can sufficiently reconstruct the original *SIMcode*$_1$. Note that the construction of *SIMcode*$_1$ by an adversary is useless, as it cannot derive or extract meaningful information from *SIMcode*$_1$ and the information sent over the network. Later, in our protocol after verifying sender *MS* and all recipients MS_i, the *AS* sends a new *new-SIMcode*$_1$ and *new-SIMcode*$_i$ to the *MS* and all MS_i, respectively, for subsequent authentication request.

2) Pseudo-Identity Generation: The generation of *TID* and retrieval of *IMSI* are not publicly available. We consider *IMSI* 128-bit as defined in the *3GPP* specifications [40], according to which the length of the compressed *IMSI* and encrypted *IMSI* shall be 64 bits (8 octets) and 128 bits (16 octets), respectively. We use an encryption function to generate a temporary identity of each participating user. Each MS_i (including sender *MS*) computes TID_i as $TID_i = f_2(IMSI_i, T_i)_{DK_i}$ to prevent the transmission of the original $IMSI_i$ over the network that protects *ID*-theft, eavesdropping, and *MITM* attacks. Here, T_i is the current timestamp, DK_i is a delegation key, and $f_2()$ is a reversible symmetric encryption function (*e.g. AES-CTR*). The structure of this function may be known; however, DK_i key remains secret.

3. Protocol Initialization: Let m be the total number of authentication requests generated by various mobile users MS_i (where i = 2, 3, ..., m+1) to the *AS* at the same time when they receive a request from the sender *MS*. Initially, each MS_i (and sender *MS*) chooses a random number $K_i \in \mathbb{Z}_p^*$ (where p is a large prime integer of 128 bits), generates current timestamp T_i, and derives a delegation key DK_i, where $DK_i = f_1(T_i)_{SK_i}$ and $f_1()$ is a hash-based *MAC* function, such as *HMACSHA256*. Thereafter, each MS_i computes $Y_i = K_i \oplus IMSI_i$ and a symmetric-signature $Z_i = (K_i + DK_i \oplus ReqNo)\ mod\ m$, where $\oplus$ is a bitwise *XOR* operation. Each mobile user generates a valid symmetric-signature and fulfills the security properties with Assumption 3, such as *authenticity* (the signer itself signs the associated message with its key), *unforgeability* (only the signer can generate a valid symmetric-signature for the associated message, assuming an honest *AS*), *non-reusability* (generated symmetric-signature cannot be reused), *non-repudiation* (signer cannot deny the signing of a message, *i.e.*, symmetric-signature, with a honest *AS*), and *integrity* (ensures that content has not been modified). Note that in symmetric key cryptography, both parties know the shared secret key. If they send messages to a third party, then it is difficult to determine the sender of the message received by a third party. In such a scenario, only two parties are involved. In other words, only the MS_i (and sender MS) and the AS know the corresponding SK_i key as well as the generated DK_i key.

4. Protocol Execution: The execution of *BVPSMS* is divided into two phases: fresh authentication and subsequent authentication.

Phase-1: Batch Authentication: The proposed protocol performs the following six steps:
Step 1. [*MS* $\rightarrow$ *MS*$_i$: T_1, *ReqNo*, *TID*$_1$, *S-Actcode*$_i$, *MAC1*$_1$]: The sender MS multicasts its authentication request as T_1, *ReqNo*, TID_1, *S-Actcode*$_i$, and $MAC1_1 = f_3(T_1, ReqNo, TID_1, S\text{-}Actcode_i)$ to all targeted MS_i (message-1), where $f_3()$ is a hash-based *MAC* function, such as *HMACSHA1*.
Step 2. [*MS*$_i$ $\rightarrow$ *AS*: T_1, *ReqNo*, *TID*$_1$, *S-Actcode*$_i$, *MAC1*$_1$, *Actcode*$_i$, *TID*$_i$, T_i, Y_i, Z_i, *MAC2*$_i$]: On receiving the request, all recipients MS_i compute $MAC1_1^{'}$ and verify $MAC1_1 \stackrel{?}{=} MAC1_1^{'}$. If it verifies, then the respective MS_i proceeds; otherwise, the connection is terminated by the MS_i. The MS_i who successfully verify $MAC1_1$, compute and send their activation codes *Actcode*$_i$, temporary identity TID_i, timestamps T_i, variables Y_i and Z_i, and $MAC2_i$ to the AS along with message-1 received from the MS except $MAC1_1$ (message-2), where $MAC2_i = f_3(T_1, ReqNo, TID_1, S\text{-}Actcode_i, T_i, TID_i, Y_i, Z_i, Actcode_i)$.
Step 3. [*AS* $\rightarrow$ *MS*: $E\{T_{m+1}, \boldsymbol{ExpT}\}_{\boldsymbol{DK_1}}$]: On receiving the message, the AS computes $MAC2_i^{'}$ for all the received messages from different MS_i and compares $MAC2_i \stackrel{?}{=} MAC2_i^{'}$. If the verification returns false, then the AS terminates the connection for the MS_i. Otherwise, the AS extracts the hashed *SIMcode*$_i$ and computes DK_i key from the respective *Actcode*$_i$ and SK_i, and $IMSI_i = f_2(TID_i, T_i)_{DK_i}$ from the received TID_i for all valid MS_i. The AS also computes the hashed *SIMcode*$_1$, extracts SK_1 key, derives DK_1 key, and retrieves $IMSI_1$. 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 $\sum_{i=1}^{m}(Y_i \stackrel{?}{=} R)$ is true at the AS, all MS_i are successfully verified by the AS. Otherwise, one/more MS_i are malicious, which requires a re-batch authentication.

Re-batch Authentication Process: In a re-batch authentication, the AS finds all invalid MS_i using a detection algorithm and removes all invalid MS_i from the batch. The AS detects malicious MS_i using an algorithm "Malicious_Requests_Detection", discussed in Section 4. After removing malicious MS_i from a batch, the AS re-computes $P = \sum_{i=1}^{m-t}(DK_i \oplus IMSI_i)$ and $R = \sum_{i=1}^{m-t}(Z_i \oplus IMSI_i) - (ReqNo \oplus P)$, where t is the total number of malicious MS_i. Thereafter, the AS compares $\sum_{i=1}^{m-t}(Y_i \stackrel{?}{=} R)$, and ensures that all legitimate MS_i are authenticated. Finally, the AS sends $E\{T_{m+1}, ExpT\}_{DK_1}$ to the sender MS (message-3), where *new-ReqNo* is a new request number assigned by the AS for subsequent request.
Step 4. [*MS* $\rightarrow$ *AS*: $E\{T_{m+1}\}_{\boldsymbol{DK_1}}$]: The MS replies $E\{T_{m+1}\}_{DK_1}$ as an acknowledgment to the AS (message-4).
Step 5. [*AS* $\rightarrow$ *MS*$_i$: P_i, $E\{T_{m+2}, \boldsymbol{new\text{-}ReqNo}, \boldsymbol{ExpT}, \boldsymbol{new\text{-}SIMcode}_i, \boldsymbol{DK}_1\}_{\boldsymbol{DK_i}}$, *MAC3*$_i$]: The AS decrypts the message as $D\{E\{T_{m+1}\}_{DK_1}\}_{DK_1}$ and verifies T_{m+1}. Furthermore, the AS sends all P_i to the respective MS_i along with $E\{T_{m+2}, new\text{-}ReqNo, new\text{-}SIMcode_i, ExpT, DK_1\}_{DK_i}$ and $MAC3_i$ (message-5), where $MAC3_i = f_3(P_i, E\{T_{m+2}, new\text{-}ReqNo, new\text{-}SIMcode_i, ExpT, DK_1\}_{DK_i})$. On receiving the message, all MS_i compute $MAC3_i^{'}$ and compare $MAC3_i \stackrel{?}{=} MAC3_i^{'}$. If it holds, all MS_i compute $P_i^{'}$ and compare $P_i \stackrel{?}{=} P_i^{'}$, where $P_i^{'} = (DK_i \oplus IMSI_i)$. If it verification returns true, the AS is verified by all MS_i. Otherwise, the particular MS_i terminates the connection.
Step 6. [*AS* $\rightarrow$ *MS*: $E\{T_1, \boldsymbol{new\text{-}ReqNo}, \boldsymbol{new\text{-}SIMcode}_1\}_{\boldsymbol{DK_1}}$]: Finally, the AS sends $E\{T_1, new\text{-}ReqNo, new\text{-}SIMcode_1\}_{DK_1}$ to the MS (message-6), where T_1 (first timestamp) shows the completion of authentication process. Thereafter, both ends can communicate with secure messages encrypted by *AES-CTR* with 128 bits key.

Phase-2: Subsequent Authentications: Any subsequent request made by sender MS within a pre-determined expiry time of DK_1 executes as follows:
Step 7. [*MS* $\rightarrow$ *MS*$_i$: *new-ReqNo*, $E\{\boldsymbol{TID}_1, T_j\}_{\boldsymbol{DK_1}}$]: The MS sends *new-ReqNo*, $E\{TID_1, T_j\}_{DK_1}$ to all respective MS_i (message-7).
Step 8. [*MS*$_i$ $\rightarrow$ *MS*: E{*new-ReqNo*$_i$, $T_{j+1}\}_{DK_1}$]: All MS_i check *new-ReqNo*, retrieve the corresponding DK_1 from their memory, and decrypt the received message. Furthermore, if $T_j \leq ExpT$, all respective MS_i compute another request number $new\text{-}ReqNo_i = f_3(new\text{-}ReqNo, TID_1, T_j)$ and sends it to the MS along with T_{j+1} (message-8). The same *new-ReqNo*$_i$ is computed by each MS_i. However, T_{j+1} is different for each MS_i. Thereafter, the MS retrieves the message and stores *new-ReqNo*$_i$ in its memory for the subsequent request.

4 Discussion

In this section, we discuss the reliability of the *BVPSMS* protocol, an algorithm to detect malicious requests, and the impact of user mobility [8], [41].

Proposition 1. *Hypergeometric distribution probability helps to predict malicious requests in a batch and determines the reliability of the proposed protocol.*

If we can determine the approximate number of malicious user requests involved in the process, Hypergeometric distribution probability can help us determining the probability in detecting malicious requests in our system. Deploying an Intrusion Detection System (*IDS*), such as presented in [42] in the cellular network, can identify the suspicious malicious users. Let N_{MS} be the maximum number of authentication requests generated by mobile users at any point of time. Realistically, some of these requests may be malicious, denoted as N_{IN}. Also, we assume that N_{AS} is the maximum capacity of the AS to authenticate requests at any point of time. For the statistical analysis, we assume that N_{MS} = 100, N_{AS} = 50, and N_{IN} = 10% of the N_{MS}, *i.e.*, 10. Let *Prob*$\{t\}$ is the probability when t malicious

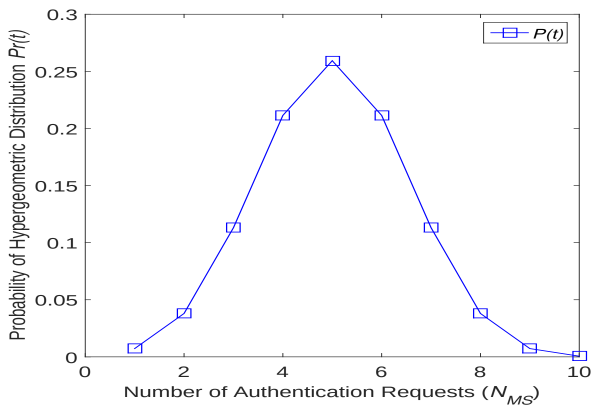


Fig. 4: Reliability analysis of our protocol when t = [1-10].

authentication requests are sent to the *AS*. The probability of Hypergeometric distribution [43] is as follows:

$$Prob\{t\} = \frac{\binom{N_{MS}-N_{IN}}{N_{AS}-t}\binom{N_{IN}}{t}}{\binom{N_{MS}}{N_{AS}}}, where\ t = 1, 2, ..., 10.$$

This indicates that ($N_{AS} - t$) valid requests are sent out of ($N_{MS} - N_{IN}$). Figure 4 shows the probability of Hypergeometric distribution when N_{MS} = 100, N_{AS} = 50, N_{IN} = 10, and malicious requests are t = 1, 2, 3, ..., 10. This probability is maximum (0.25) for t = 5 (half of t), and minimum (0.00059) for t = 10 (last of t values).

Proposition 2. *There exists an algorithm that detects malicious requests in a batch.*

In practice, few of the mobile participants may be dishonest or malicious. A dishonest mobile participant will always lie about the true secret value. Our scheme assumes that all shares lie on a single polynomial of degree at most $k - 1$. This might not hold if the sender mobile user is dishonest or malicious and sends bad shares to some of the mobile recipients. However, our system model has a honest sender mobile user. But a mobile user participant who lies about his share can cause reconstructing incorrect value of the secret (hash of *SIMcode*) at the server. Our scheme is a fault-tolerant scheme that allows the hash of *SIMcode* to be correctly reconstructed, even in the presence of a certain number of corrupted shares.

Algorithm 1 *Malicious_Requests_Detection*

Input: The *AS* receives a set (*AR*) of m-authentication requests (R_i) as *AR* = $\{R_1, R_2, R_3, ..., R_m\}$ at any time.
Output: Returns a set of malicious requests (*MR*), otherwise returns True.

if (batch verification (*AR*, m) == 1) **then** returns True.
else
  while (batch verification (*AR*, m) != 1) **do**
    $AR_1 = \{R_1, R_2, R_3, ..., R_{\lceil m/2 \rceil}\}$;
    $AR_2 = \{R_{\lceil m/2 \rceil+1}, R_{\lceil m/2 \rceil+2}, R_{\lceil m/2 \rceil+3}, ..., R_m\}$;
    batch verification (AR_1, $\lceil m/2 \rceil$);
    batch verification (AR_2, $m - \lceil m/2 \rceil$);
    if (m == 1 && batch verification (*AR*, m) != 1) **then**
returns *MR* = $\{IMSI_i\}$

We propose an algorithm to detect malicious requests of the MS_i in a batch in at most $log\ m$ verification rounds (O($log\ m$)). The proposed algorithm, based on binary search approach, is explained as Algorithm 1. Only the hash-based search complexity is better than binary search. The hash-based searching is useful when you know the data, and even more efficient when the data is in sorted order (O(1)). However, in our protocol, the *AS* neither knows the actual data nor stores any data until it is verified. In such case, the proposed algorithm for malicious detection is suitable. Note that "batch verification (*AR*, m)" is the batch verification process at the *AS* involving P, R, and Y_i as explained in our protocol. Each invalid MS_i is placed on a black-list and can only be removed once the predefined time is over. During this period, the request from particular MS_i is discarded.

More generally, if there can be t malicious users with faked shares ($S\text{-}Actcode_i^{'}$, $i = 1, 2, ..., t$), we can show that the secret can be recovered and the malicious users identified if $k + 2t$ shares are available for reconstruction. In other words, we need at least $k + t$ honest shares available (in addition to the t possible malicious users) in order to recover the secret (hash of *SIMcode*) and identify the malicious users. We assume that there are t cheaters or malicious users participating at any time, where $t \leq k/2$. In any secret sharing cheater or malicious identification scheme, the optimal cheating threshold is $k = 2t + 1$. In [44], it is shown that in any such scheme, the following lower bound must be satisfied: $|V| \geq (|S-Actcode|-1)/\epsilon+1$, where $|V|$ exactly matches the above bound is said to be optimal. Let $k = 2t+1$, $p = 1/\epsilon$ and $|S-Actcode| = p^i$, where $i > 1$ and $S-Actcode = (S-Actcode_1, S-Actcode_2, ..., S-Actcode_i)$ is a shared secret. We can identify up to t malicious users such that $|V| = |S-Actcode|/\epsilon^{3n}$ [45]. Now, we assume that j ($n \geq j \geq t$) number of participants are involved in a secret reconstruction out of n. Then, we have $j-t$ legitimate shares in a secret reconstruction. When $j - t > t$ ($j \geq t+1$), there are $\binom{j-t}{t}$ cases that will construct the legitimate secret [46]. This attack of not being able to reconstruct the secret succeeds only when $j - t < t$.

Proposition 3. *There is a sustainable impact of mobility when a user moves out of range of the home AS.*

It is also assumed that the *AS*s are deployed at different geographic locations similar to traditional cellular networks, and are interconnected to each other with a pre-shared secret key between each pair of the *AS*s. When a roaming mobile user requests for an *SMS* service, the corresponding *AS* of that area handles the request, sends the request message encrypted with pre-shared key to the home *AS* of the user. The protocol execution takes place at the home *AS* and the result is returned securely to the roaming *AS* securely. Finally, the roaming *AS* grants/revokes *SMS* service to the respective mobile user. Also, if few *MS*s are out of network, the *AS* will verify whether it has received at least k messages from different *MS*s. If it holds, the *AS* proceeds, otherwise the *AS* waits for a timeout period. If the *AS* still does not receive k messages, it discards the connections, and notifies to the sender *MS* to restart his/her request.

5 Security Analysis

This section achieves the security goals outlined in Section 2.2.

Property 1. *The proposed protocol provides mutual authentication between all MS/MS_i and the AS.*

The *BVPSMS* protocol provides mutual authentications between the *AS* and the *MS*, and between the *AS* and the MS_i. The *AS* authenticates all MS_i by verifying $\sum_{i=1}^{m}(Y_i \stackrel{?}{=} R)$ while each MS_i authenticates the *AS* by comparing $P_i \stackrel{?}{=} P_i^{'}$. The sender *MS* authenticates the *AS* by decrypting the received message-3 using DK_1 while the *MS* is authenticated by the *AS* by verifying T_{m+1}.

Property 2. *The BVPSMS protocol initiates a secure session key establishment between all MS/MS_i and the AS. In fact, Adversary $\mathcal{A}$ will not be successful in obtaining SK_1/SK_i or DK_1/DK_i key, even if it captures S-$Actcode_i$/$Actcode_i$ of a MS.*

A unique DK_i key is used within the expiry of a session for each authentication between the *AS* and each MS_i. $\mathcal{A}$ is unable to generate DK_1/DK_i key as it does not know the SK_i key and the key generation function $f_1()$. Since each *S-$Actcode_i$/$Actcode_i$* is sent over the network only once, the protocol is secure even if $\mathcal{A}$ is able to capture *S-$Actcode_i$/$Actcode_i$*. Moreover, $\mathcal{A}$ cannot derive any relation among captured *S-$Actcode_i$/$Actcode_i$*, as $SIMcode_1/SIMcode_i$ are randomly generated at the *AS*. Moreover, after each authentication, *new-$SIMcode_1$/new-$SIMcode_i$* are sent to each involved *MS*/MS_i. Furthermore, if $\mathcal{A}$ modifies $Actcode_i$ in message-2, the computed $MAC2_i^{'}$ will not match with the received $MAC2_i$ at the *AS*. Hence, the MS_i will terminate the connection.

Property 3. *Adversary $\mathcal{A}$ cannot trace the original identity of the MS/MS_i. In fact, $\mathcal{A}$ is not able to identify the actual user, even if it captures the TID_1/TID_i of a mobile user.*

Our protocol preserves identity anonymity and untraceability properties.

Untraceability: Our protocol satisfies untraceability as $\mathcal{A}$ cannot distinguish whether two *TID*s correspond to the same *MS*/MS_i or two different *MS*/MS_i.

$Verify(publicChannel)[(IMSI_1, IMSI_2)|TID_i|MS/MS_i|AS] \approx Verify(publicChannel)[IMSI_1|IMSI_2|TID_i|MS/MS_i|AS]$.

In our protocol, privacy of each MS_i (including *MS*) is ensured. Each TID_i is computed from the original $IMSI_i$ as $TID_i = f_2(IMSI_i, T_i)_{DK_i}$ before a message is sent by each MS_i over the network. We implement $f_2()$ using *AES-CTR* with DK_i key since no practical full attack has revealed against on *AES*. As TID_i is used by each MS_i over the network, $\mathcal{A}$ is unable to trace the original identity of the user.

IND-ANO: Indistinguishability under Anonymous Identity: Our protocol is *IND-ANO* as no adversary $\mathcal{A}$ at time t can distinguish between two chosen identities TID_1 and TID_2 with a negligible ϵ advantage.

$Pr[\mathcal{A}(TID_1) = 1] - Pr[\mathcal{A}(TID_2) = 1] \leq \epsilon$. $\mathcal{A}$ cannot distinguish and relate TID_i and other messages with ID_i, as each TID_i is used only once over the network. For all subsequent requests, a different *new-$ReqNo_i$* is used each time when the sender *MS* connects to the MS_i. The MS_i sends an encrypted *new-$ReqNo_i$* to the *MS* that will be used for the next authentication within a session. Hence, untraceability and identity anonymity are ensured, as $\mathcal{A}$ cannot trace TID_i, $SIMcode_i$, and *new-ReqNo* to link with users, and also $IMSI_i$ would not be revealed to $\mathcal{A}$ and intermediate operators.

Property 4. *Adversary $\mathcal{A}$ cannot link current session information with previous sessions. Moreover, our protocol maintains perfect forward secrecy and Indistinguishability under Chosen Plaintext Attack (IND-CPA).*

The MS_i (including *MS*) and the *AS* generate fresh DK_i keys with unique timestamps, TID_i, $Actcode_i$, and K_i. Therefore, $\mathcal{A}$ cannot retrieve the information based on linkability among users.

Forward Secrecy: Our protocol maintains forward secrecy as no $\mathcal{A}$ could obtain past keys and generate future keys.

The SK_i and DK_i keys are never sent over the network, and a new DK_i key is used in each fresh session to encrypt $IMSI_i$ using *AES-CTR*. Even compromising current DK_i will not allow $\mathcal{A}$ to obtain or generate past and future keys. Also, the past keys cannot be used for future sessions, as both ends generate a fresh DK_i key.

IND-CPA: Our protocol is *IND-CPA* secure as no adversary $\mathcal{A}$ in time t can distinguish between two chosen messages msg_1 and msg_2, and has no or negligible advantage. $\underset{DK_i \leftarrow SK_i}{Pr}[\mathcal{A}(msg_1) = 1] - \underset{DK_i \leftarrow SK_i}{Pr}[\mathcal{A}(msg_2) = 1] \leq \epsilon$.

Assuming that $\mathcal{A}$ has unlimited access to the encrypted data using a random oracle, the messages encrypted by the same key in our protocol generate different ciphertexts. Even encrypting the same plaintext with the same key generates different ciphertext, as at least one of the input parameters of the message is always different. The MS_i generates TID_i as $f_2(IMSI_i, T_i)_{DK_i}$, where T_i changes for each fresh message. We use *AES-CTR* as $f_2()$ that encrypts successive values of a counter with *AES*, and regurgitates concatenation of the encrypted blocks. *AES-CTR* stream never includes twice the same block and is *IND-CPA*.

Property 5. *The proposed protocol defeats SMS disclosure, SMS spoofing, replay, MITM, and impersonation attacks between the MS/MS_i and the AS. Also, the protocol provides security protection over the air and SS7 channel. Furthermore, adversary $\mathcal{A}$ cannot compromise message confidentiality and integrity.*

BVPSMS provides mutual authentication between the *AS* and the *MS*/MS_i by verifying $(\sum_{i=1}^{m} X_i) \stackrel{?}{=} R$, and $P_i \stackrel{?}{=} P_i^{'}$. This process prevents the system against impersonation attack. Furthermore, transmitted messages are securely encrypted using *AES-CTR*, which protects the system against *SMS* disclosure and *MITM* attack. $\mathcal{A}$ is unable capture actual *IMSI* using *IMSI* catcher, as each *MS*/MS_i sends its *TID* over the network. It also prevents *SMS* spoofing. Furthermore, a timestamp value sent with each message protects the system against replay attack. Our protocol provides end-to-end *SMS* security from the sender *MS* to all recipients MS_i over *OTA* interface and *SS7* channel, as each confidential message is encrypted using strong encryption. Moreover, message integrity (message content and its threshold delivery in time) is maintained, as $T_{receive} \leq T_{generate} + T_{threshold}$ and

TABLE 2: Requirements vs. Protocols

Prevention Goals	*ABAKA* [8]	*RAISE* [6]	*SPECS* [11]	*b-SPECS+* [9]	*BVPSMS*
Mutual Authentication	Yes	No	Yes	Yes	Yes
User Privacy	Yes	Yes	No	No	Yes
Integrity Protection	No	Yes	No	No	Yes
Replay Attack	Yes	Yes	No	No	Yes
MITM Attack	Yes	Yes	Yes	Yes	Yes
Impersonation Attack	Yes	Partial	No	Yes	Yes

*MAC*s are used for verification. The messages received after the threshold time will be lapsed.

Table 2 lists the security and privacy requirements achieved by existing protocols. These protocols are secure against *MITM* attack, but do not provide integrity protection to the messages with the exception of *RAISE* [6]. However, *RAISE* [6] does not provide mutual authentication and is partially secure against impersonation attacks. We remark that user privacy is preserved in *ABAKA* [8] and *RAISE* [6]. The *SPECS* [11] and *b-SPECS+* [9] suffer from replay attack. Thus, our proposed protocol fulfills all the mentioned requirements.

Property 6. *Our protocol is secure against both passive and active corruption attacks in the presence of non-adaptive and/or adaptive adversaries $\mathcal{A}$.*

In passive and active corruption attacks, $\mathcal{A}$ obtains complete information held by the corrupted MS_i (while a MS_i still runs protocol correctly) and $\mathcal{A}$ takes over control of corrupted MS_i, respectively. In both cases, our protocol maintains *IND-CPA* indistinguishability as well as perfect forward secrecy. Moreover, keys are never sent over the network, and delegation keys are generated only for a session. Furthermore, both passive and active adversaries can be non-adaptive (a set of corrupted MS_i is chosen before the protocol starts) or adaptive (a corrupted MS_i is selected at any time during protocol run). In any case, $\mathcal{A}$ acting as corrupted MS_i does not affect the security of the protocol.

Property 7. *The proposed protocol achieves fairness and guarantees that "no MS_i (malicious or legitimate) has an advantage".*

A protocol is said to be fair if it ensures that no user can gain a significant advantage over other users, even if the protocol halts for any reason. In our protocol, the MS/MS_i and the AS learn each others' information. However, the MS and the MS_i cannot learn any information about each other, as one user is unable to obtain DK_i keys belonging to other users. Users are also unable to derive $IMSI_i/TID_i$ of each others, as each DK_i is secret. Also, $\mathcal{A}$ cannot generate a valid symmetric-signature S_i, as it does not know the correct SK_i and/or DK_i keys, and K_i is randomly generated by each MS_i. Our protocol also maintains *IND-CPA*; therefore, no MS_i has an advantage over others.

Property 8. *BVPSMS maintains fairness and correctness under honest, semi-honest, and dishonest majority scenarios.*

Our protocol fairly works under all three scenarios. We consider these scenarios only for the MS_i, not for the AS. The reason is that the AS keeps SK_i keys of all MS_i secret. Hence, it cannot be dishonest or semi-dishonest. The effectiveness of our protocol under all three scenarios can be observed by re-batch verification delay. Our protocol maintains security properties under these scenarios, such as *IND-CPA*, forward secrecy, and fairness.

6 Performance Evaluation

This section presents the performance evaluation of *BVPSMS* in terms of overheads, verification and re-batch verification times, and the time, space, and cost analysis.

6.1 Analysis

This subsection analyzes the performance of the *BVPSMS* protocol. We compare the communication overhead generated by *RAISE* [6], *ABAKA* [8], *SPECS* [11], and *b-SPECS+* [9] along with the *BVPSMS* protocol. There is no batch protocol for *SMS* security in the literature. However, we compare the communication overhead generated by the protocols with our protocol, as all protocols are based on authentication considering the same wireless network communication scenario, and also the flow of information is same in all the protocols. However, the computation overhead and verification delay are different in both types of the protocols because *VANET* protocols have additional devices and road side equipment to communicate information over the network.

6.1.1 Communication Overhead

Let m be the number of recipients MS_i, and r be the number of subsequent multiple authentication requests within the expiry time, *i.e.*, *ExpT*. The communication overhead can be defined as the total number of bits transmitted during the authentication process over the network. The transmission overhead generated by the *BVPSMS* protocol during m-authentication requests can be evaluated as:
Phase-1: Total number of transmitted bits = (1)+(2)+(3)+(4)+(5)+(6) = (128+64+8+64+64)$\times m$ + (128+64+8+64+64+128+64+128+128+64+64)$\times m$ + (64+64+8) + (64) + (128+64+64+8+64+64+128)$\times m$ + (8+64+64) = 336+1752$\times m$ bits.
Phase-2: Total number of transmitted bits = ((7)+(8))$\times r$ = (128+8+64)$\times r$ + (64+8)$\times r$ = 200$\times r$.
Total overhead = 42+(219$\times m$)+(25$\times r$) bytes.

BVPSMS is our original protocol that provides integrity to each message in two phases. Since all the protocols

TABLE 3: Communication Overhead in Batch Authentication by Different Protocols

Protocols	Device-Server (bytes)	Intermediate Authority Server (bytes)	Server-Device (bytes)	Total (bytes)
ABAKA [8]	$84\times m$	–	$80\times m$	$164\times m$
RAISE [6]	$200\times m$	–	–	$200\times m$
SPECS [11]	$48\times m$	$96\times m$	$32\times m$	$176\times m$
b-SPECS+ [9]	$48\times m$	$176\times m$	$64\times m$	$288\times m$
*BVPSMS**	$97\times m$	–	$57\times m$	$154\times m$
*BVPSMS***	$25\times r$	$9\times r$	–	$34\times r$

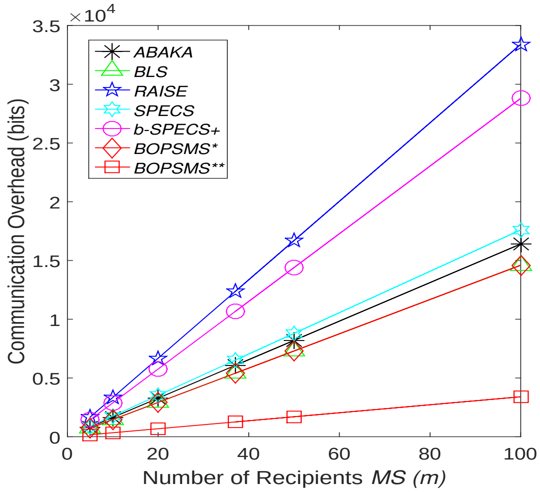


Fig. 5: Communication overhead analysis.

RAISE [6] compared in Table 3 provide no integrity, we use two variants of *BVPSMS* for comparison: *BVPSMS** for fresh authentication without integrity protection (as phase-1), and *BVPSMS*** for each subsequent authentication within the expiry time of DK_1 key (as phase-2). For m-authentication requests, *BVPSMS** generates 154$\times m$ bytes overhead, which is lowest among all the protocols discussed in the paper, while for all subsequent authentication requests, the overhead is only 34$\times r$ bytes.

From Figure 5, it is clear that *BVPSMS** and *BVPSMS*** generate less communication overhead among all protocols. *BVPSMS** reduces the communication overhead by 6.1%, 23%, 12.5%, and 46.52% in comparison to *ABAKA*, *RAISE*, *SPECS*, and *b-SPECS+*, respectively, when m = 5, 10, 20, 50, 100. For any subsequent authentication request, *BVPSMS*** produces significantly low overhead in comparison to all the protocols. It reduces the communication overhead by 79.27%, 89.83%, 80.69%, and 88.2% in comparison to *ABAKA*, *RAISE*, *SPECS*, and *b-SPECS+*, respectively, when r = 5, 10, 20, 50, 100.

6.1.2 Computation Overhead

The computation overhead generated by *BVPSMS* during m-authentication requests is shown in Table 4. We consider all functions as a single unit cost. Then, the computation at the *MS*, MS_i, and *AS* are as follows:

Phase-1: At the *MS* = 8, At the MS_i = 11$\times m$, and At the *AS* = 6+14$\times m$.

Phase-2: At the *MS* = 2$\times r$ and At the MS_i = 2$\times r$.
Total computation overhead = 8 + (11$\times m$) + (6+14$\times m$) + (2$\times r$) + (2$\times r$) = 14+(25$\times m$)+(4$\times r$) bits.

We compute the communication and computation overheads (in bits) generated by our protocol when m = 10, 20, 50, 100; r = 1, 2, 5, 10. For m=100, the generated communication overheads are 2745.875 bytes and 2970.875 bytes, respectively, when r=1 and r=10. Similarly, when m=100, the computation overheads for r=1 and r=10 are 314.75 bytes and 319.25 bytes, respectively. This indicates that our protocol is efficient even when a large number of subsequent authentication requests is executed.

TABLE 4: Computation Overhead in Batch Authentication

Entity Name	Total Computation (Time Computation)
Phase-1	
At the *MS*	$T_{f_1()}$, $T_{f_2()}$, $2T_{D\{\}_{DK_1}}$, $T_{E\{\}_{DK}}$, $T_{f_3()}$, $T_{H()}$, $T_{S\text{-}Actcode}$
At the MS_i	$mT_{f_1()}$, $mT_{f_2()}$, $3mT_{XOR}$, mT_{Add}, $mT_{D\{\}_{DK}}$, $2mT_{f_3()}$, $mT_{Actcode}$, $mT_{H()}$
At the *AS*	$(m+1)T_{f_1()}$, $(m+1)T_{f_2()}$, $(m+1)T_{H()}$, $3mT_{f_3()}$, $(3m-3)T_{Add}$, T_{Sub}, $(2m+1)T_{XOR}$, $(m+2)T_{E\{\}_{DK}}$, $T_{D\{\}_{DK}}$, $(m+1)T_{SIMcode}$
Phase-2	
At the *MS*	$rT_{E\{\}_{DK}}$, $rT_{D\{\}_{DK}}$
At the MS_i	$rT_{D\{\}_{DK}}$, $rT_{E\{\}_{DK}}$

6.2 Simulation

This section presents the simulation results of our protocol in terms of the total execution and verification times. We also perform time, space, and cost analysis of our protocol.

6.2.1 Protocol Execution Time

We implemented a client-server paradigm for our system, where the MS/MS_i are the clients and the *AS* is a server. We performed various operations on an Intel Core i3-2330M 2.20GHz machine with Windows7 OS, 256 MB RAM, using JDK1.7 with J2ME WTK mobile emulator. On average, the execution time to perform addition, *XOR*, and subtraction are T_{add} = 0.0009 milliseconds (ms), T_{xor} = 0.03 ms, and T_{sub} = 0.0009 ms, respectively. We setup the system with 50 MS_i (and one *MS*) transmitting their messages to the server *AS*, when the *MS* sends an *SMS* to these MS_i. The average value of 30 iterations is considered for each result.

Note that protocol execution time is the complete time for mutual authentication between all MS/MS_i and the *AS*. Table 5 shows simulation results obtained for various functions' computations. Here, *Ext*, *TUM*, *Enc*, and *Dec* are the execution time (ms), total used memory (bytes), encryption, and decryption process, respectively. The $f_2()$ is implemented as *AES-CTR*, where encryption (generation of TID_i) took 13.6 ms and decryption (generation of $IMSI_i$) is performed in 4.2 ms. The same results are obtained for $E\{\}_{DK_1}/D\{\}_{DK_1}$ using *AES-CTR*. The $f_1()$ and $f_3()$ are implemented as *HMACSHA256* and *HMACSHA1*, respectively. The output of *HMACSHA1* and *HMACSHA256* are truncated to 64 and 128 bits, respectively because the output of $f_3()$ is 64 bits *MAC*, whereas the output of $f_1()$ is 128 bits, which is DK_i key. The input to the *HAMCSHA1* and *HMACSHA256* are 512 bits each (actual input size plus

TABLE 5: Computations of Various Used Functions

Function	*ExT* (ms)	*TUM* (bytes)
$f_1()$=*HMACSHA256*	185	15204024
$E\{\}_{DK_1}/f_2()$=*AES-CTR*	13.6	9139681
$D\{\}_{DK_1}/f_2()$=*AES-CTR*	4.2	9124165
$f_3()$=*HMACSHA1*	172	15211840
$H()$=*SHA256*	20	14321156

trailing zeros to make it multiple of 512). Also, the execution time of *SIMcode* using a random number generation and hash generation time of $H()$ using *SHA256* are 0.89 ms and 20 ms, respectively.

Total execution time of a single authentication:
Phase-1: Total time = transmission time for all messages in phase-1 + time at the entities (*MS*, MS_i, *AS*) = 2.98 sec. Hence, on average the execution time per user = 1.49 sec.
Phase-2: Total time = transmission time for all messages in phase-2 + time at the *MS*, MS_i, *AS* = 10.7+35.6 = 46.3 ms.
Total execution time of a batch authentication:
Phase-1: Total time = transmission time for all messages in phase-1 + time at the *MS*, MS_i, *AS* = 672.20+$m\times$1330.41 ms.
Phase-2: Total time = transmission time for all messages in phase-2 + time at the *MS*, MS_i, *AS* = $r\times$46.3 ms.

6.2.2 Verification Time

The verification delay in our protocol is evaluated between the MS/MS_i and the *AS*. It is the time estimation between the sent messages and the received response or the completion of the protocol.

***BVPSMS* Phase-1** *(Time to verify): MS_i by AS* = 0.0282+391.55$\times m$ ms, *MS by AS* = 236.40+172.89$\times m$ ms, *AS by MS_i* = 172.03$\times m$ ms, and *AS by MS* 4.2 ms.
Total delay in phase-1 = 240.62+736.47$\times m$ ms.

***BVPSMS* Phase-2** *(Time to verify): MS by MS_i* = 17.8$\times r$ ms, and *MS_i by MS* 4.2$\times r$ ms.
Total verification delay in phase-2 = 32$\times r$ ms.
Therefore, total verification delay in *BVPSMS* = 240.62+736.47$\times m$+32$\times r$ ms.

6.2.3 Re-batch Verification Time

If a batch authentication is not successful, it is expected to execute a re-batch authentication without including the malicious MS_i. After detecting the malicious MS_i, it is required to remove them from the batch and execute a re-batch authentication process. The delay in re-batch verification can be estimated as follows:
Total delay in a re-batch verification = 0.000933$\times$3$(m-1-t)$ + 0.030322 + 0.000933 = 0.028456+0.002799$\times(m-t)$ ms.

6.2.4 Simulation Results

The execution time of the *BVPSMS* protocol is observed when m = 10, 20, 50, 100; r = 1, 2, 5, 10. For m=100, the protocol execution times are 133.75 sec. and 134.17 sec., respectively, when r=1 and r=10, which are actually on average, 1.32 sec. and 1.21 sec. per user, respectively. It is clear that on average, the execution time per mobile user decreases when r increases. The execution time per mobile user also decreases when m increases and r is fixed. On average, the execution times of our protocol are 1.44, 1.38, 1.35, and 1.34 sec., respectively, when r=10 (fix) and m=10, 20, 50, and 100. The verification times for phase-1 and phase-2 of our protocol are also evaluated when m = 10, 20, 50, 100 and r = 1, 2, 5, 10, 50. For m=100, on average the verification time per user for batch authentication is 0.71 sec. Furthermore, for r=10 and r=50, the total verification times are 0.3 sec. and 1.6 sec., respectively, and on average, the verification time for each subsequent authentication per user is 0.03 sec. It is also clear that the increase in r lowers verification time, on average per mobile user. Re-batch verification time is also computed in our protocol when m = 10, 20, 50, 100 and malicious requests t = 2, 4, 6, 8, 10. For m=10, the re-batch verification times are 0.044 ms, 0.03 ms, and 0.028 ms, respectively, when t=2, t=9, and t=10. Similarly, for m=100, the times are 0.22 ms, 0.21 ms, and 0.20 ms, respectively, when t=2, t=9, and t=10.

6.3 Time, Space, and Cost Analysis

In both single and batch authentications, two functions $f_3()$ and $f_1()$ are implemented as *HMAC* functions. The output of *HMACSHA1* and *HMACSHA256* are 160 bits and 256 bits, respectively. The *DK* key requires 128 from 256 bits and a *MAC* needs 64 out of 160 bits. In total, 192 bits are required to be stored. Further, The time complexity of add, subtract, and *XOR* operations are constant, *i.e.*, O(1). The costs for a single authentication (8 operations) and a batch authentication (9$\times m-1$ operations) are also O(1). The time to compute *Actcode*/*SIMcode* is constant, and total cost is O(1). The block cipher algorithm, such as *AES*, works with a fixed input size and has O(1) constant complexity. However, when the algorithm has variable length of input (say $|m|$), the time is O(m). The block size is still fixed (128 bits) as the $f_2()$ and $E/D\{\}_{DK_1}$ are implemented using *AES-CTR*. Therefore, the time complexity is independent of input and is constant O(1). Hence, the costs are O(1) for $f_2()$ and $E/D\{\}_{DK_1}$ in a single authentication (2 operations) as well as batch authentication (2$\times m$ operations). The $IMSI_i$ and TID_i of 128 bits each also need to be stored in the memory. Furthermore, the storage is also required for *HMACSHA1*, *HMACSHA256*, and *AES-CTR* at the MS/MS_i as well as at the *AS*. For a re-batch verification, O(1) is only the extra cost need to be paid (for 3$\times m$-3$\times t$+2 operations). Therefore, the *BVPSMS* protocol is an efficient, secure, and cost effective protocol that requires less storage.

7 Formal Proof

This section presents the formal proof of the proposed scheme using *Proverif*. *Proverif* is an online automated tool to verify whether the logical expressions and the protocol properties are correct and valid with different queries. We perform five adversary queries: (i) Can an adversary successfully recover confidential and useful information from the messages sent over the network?, (ii) Can an adversary successfully compute parameters generated by the MS?, (iii) Can an adversary successfully compute parameters generated by the AS?, (iv) Can an adversary successfully generate *DK* key of the *MS*?, and (v) Can an adversary successfully recover secret key of the *MS*?. Following is the output observed from the *Proverif* tool:
$Neetesh@Neetesh-PC\ /proverif1.88$
$\$./proverif\ proofs/sms/BVPSMS.pv$
– Query attacker(s[]) ==> event(enableEnc)
Completing...ok, secrecy assumption verified: fact unreachable

attacker(kims[!1 = v_946])
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_2080])
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_2500,x2_2501)) ==> event(begAS(x1_2500,x2_2501))
Completing...ok, secrecy assumption verified: fact unreachable attacker(kims[!1 = v_3257])
Starting query event(endAS(x1_2500,x2_2501)) ==> event(begAS(x1_2500,x2_2501))
RESULT event(endAS(x1_2500,x2_2501)) ==> event(begAS(x1_2500,x2_2501)) is true.
– Query not attacker(*DK*[])
Completing...ok, secrecy assumption verified: fact unreachable attacker(kims[!1 = v_4332])
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_5378])
Starting query not attacker(s[])
RESULT not attacker(s[]) is true.

8 CONCLUSION

We proposed a batch verification protocol *BVPSMS* for transmitting secure *SMS* from one *MS* to multiple *MS* recipients. This protocol enjoys several advantages over the related protocols studied in the paper. *BVPSMS* provides mutual authentication between each *MS* and the *AS*. The *AS* efficiently verifies multiple authentication requests sent by different *MS*s at any one time while keeping the original *IMSI* secret during the authentication. We then demonstrated that the protocol is secure against replay attacks, *MITM* attacks, impersonation attacks, *SMS* disclosure and *SMS* spoofing, and also maintains untraceability, forward secrecy, and identity anonymity. The performance results show that in different scenarios, *i.e.*, *BVPSMS*$*$ and *BVPSMS*** when no provision of integrity protection, our protocol incurs a lower communication overhead compared to the protocols studied in this paper. Our evaluation of the protocol using Java demonstrated that the estimated re-batch verification time is almost negligible. The execution and verification times also suggested that our protocol is practical for deployment in real-world cellular networks.

ACKNOWLEDGMENTS

This research work was supported by Tata Consultancy Services Limited, India. We thank the Associate Editor and the anonymous reviewers for their constructive feedbacks.

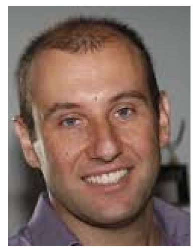
REFERENCES

[1] P. Mondal and P. Desai, "An efficient SMS-based framework for public health surveillance," in *Proc. IEEE PCHT*, 2013, pp. 244-247.
[2] B. DeRenzi, "Improving community health worker performance through automated SMS," in *Proc. 5th ICDT*, 2012, pp. 25-34.
[3] J. Chen, L. Subramanian, and E. Brewer, "SMS-based web search for low-end mobile devices," in *Proc. MobiCom*, 2010, pp. 125-135.
[4] N. Saxena and N. S. Chaudhari, "EasySMS: a protocol for end-to-end secure transmission of SMS," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 7, pp. 1157-1168, Jul. 2014.
[5] Y. Yang , J. Lu, K.-K. R. Choo, and J. K. Liu, "On lightweight security enforcement in cyber-physical systems," *Lightweight Cryptography for Security and Privacy*, LNCS vol. 9542, pp. 97-112, Jan. 2016. [Online]. http://dx.doi.org/10.1007/978-3-319-29078-2_6.
[6] C. Zhang, X. Lin, R. Lu, P. H. Ho, and X. Shen, "An efficient message authentication scheme for vehicular communications," *IEEE Trans. on Vehicular Technology*, vol. 57, no. 6, pp. 3357-3368, 2008.
[7] J. Wu and D. R. Stinson, "Three improved algorithms for multipath key establishment in sensor networks using protocols for secure message transmission," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 6, pp. 929-937, 2011.
[8] J. L. Huang, L. Y. Yeh, and H. Y. Chien, "ABAKA: an anonymous batch authenticated and key agreement scheme for value-added services in vehicular ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 1, pp. 248-262, 2011.
[9] S. J. Horng, S. F. Tzeng, Y. Pan, P. Fan, X. Wang, T. Li, and M. K. Khan, "b-SPECS+: batch verification for secure pseudonymous authentication in VANET," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 11, pp. 1860-1875, Nov. 2013.
[10] G. Calandriello, P. Papadimitratos, J. P. Hubaux, and A. Lioy, "On the performance of secure vehicular communication systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 6, pp. 898-912, 2011.
[11] T. W. Chim, S. M. Yiu, L. C. K. Hui, and V. O. K. Li, "SPECS: secure and privacy enhancing communications schemes for VANETs," *Ad Hoc Networks*, vol. 9, pp. 189-203, 2011.
[12] Y. Zhou, X. Zhu, and Y. Fang, "MABS: multicast authentication based on batch signature," *IEEE Transaction on Mobile Computing*, vol. 9, no. 7, pp. 982-993, Jul. 2010.
[13] J. A. Akinyele, M. Green, and S. Hohenberger, "Machine-generated algorithms, proofs and software for batch verification of digital signature schemes," in *Proc. Cryptology*, 2013, pp. 1-41.
[14] N. Saxena and N. S. Chaudhari, "SecureSMS: a secure SMS protocol for VAS and other applications," *Journal of Systems and Software*, vol. 90, pp. 138-150, 2014.
[15] J. L. Lo, J. Bishop, and J. H. P. Eloff, "SMSSec: an end-to-end protocol for secure SMS," *Computers & Security*, vol. 27, no. 5-6, pp. 154-167, 2008.
[16] H. Rongyu, Z. Guolei, C. Chaowen, X. Hui, Q. Xi, and Q. Zheng, "A PK-SIM card based end-to-end security framework for SMS," *Computer Standards and Interfaces*, vol. 31, no. 4, pp. 629-641, 2009.
[17] C. C. Yang, Y. L. Tang, and R. C. Wang, "A secure and efficient authentication protocol for anonymous channel in wireless communications," *Applied Math. & Computation*, vol. 169, no. 2, pp. 1431-1439, 2005.
[18] P. Gope and T. Hwang, "Lightweight and energy-efficient mutual authentication and key agreement scheme with user anonymity for secure communication in global mobility networks," *IEEE System Journal*, Mar. 2015, accepted.
[19] P. Traynor, W. Enck and P. McDaniel, "Mitigating attacks on open functionality in SMS-capable cellular networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 1, pp. 40-53, 2009.
[20] A. A. Shaikh and N. S. Vani, "An extended approach for securing the short messaging services of GSM using multi-threading elliptical curve cryptography," in *Proc. ICCICT*, 2015, pp. 44-48.
[21] S. Islam, I. U. Haq, and A. Saeed, "Secure end-to-end SMS communication over GSM networks," in *Proc. IBCAST*, 2015, pp. 286-292.
[22] S. N. Karale, K. Pendke, and P. Dahiwale, "The survey of various techniques & algorithms for SMS security," in *Proc. ICIIECS*, 2015, pp. 1-6.
[23] S. Bojjagani and V. N. Sastry, "SSMBP: a secure SMS-based mobile banking protocol with formal verification," in *Proc. WiMob*, 2015, pp. 252-259.
[24] C. C. Chang and J. S. Lee, "Efficient authentication protocols of GSM," *Computer Comm.*, vol. 28, no. 8, pp. 921-928, 2008.
[25] N. Saxena and N. S. Chaudhari, "SAKA: a secure authentication and key agreement protocol for GSM networks," *CSI Transactions on ICT*, vol. 1, no. 4, pp. 331-341, 2013.

[26] N. Saxena and N. S. Chaudhari, “NS-AKA: an improved and efficient AKA protocol for 3G (UMTS) networks,” in *Proc. CSEE*, Mar. 2014, pp. 220-224.
[27] C. C. Lee, C. L. Chen, and H. H. Ou, “Extension of an efficient 3GPP authentication and key agreement protocol,” *Wireless Personal Communications*, vol. 68, no. 3, pp. 861-872, 2013.
[28] Y. L. Huang, C. Y. Shen, and S. W. Shieh, “S-AKA: a provable and secure authentication key agreement protocol for UMTS networks,” *IEEE Trans. on Vehicular Tech.*, vol. 60, no. 9, pp. 4509-4519, 2011.
[29] N. Saxena, J. Thomas, and N. S. Chaudhari, “ES-AKA: an efficient and secure authentication and key agreement protocol for UMTS networks,” *Wireless Personal Communications*, vol. 84, no. 3, pp. 1981-2012, Oct. 2015.
[30] N. Saxena and N. S. Chaudhari, “Secure-AKA: an efficient AKA protocol for UMTS networks,” *Wireless Personal Communications*, vol. 78, no. 2, pp. 1345-1373, Sep. 2014.
[31] F. Hadiji, F. Zarai, and A. Kamoun, “Authentication protocol in fourth generation wireless networks,” in *Proc. IEEE WOCN*, 2009, pp. 36-39.
[32] G. M. Kϕien, “Mutual entity authentication for LTE,” in *Proc. 7^{th} IWCMC*, 2011, pp. 689-694.
[33] Y. W. Chen, T. Wang, K. H. Chi, and C. C. Tseng, “Group-based authentication and key agreement,” *Wireless Personal Communications*, vol. 62, no. 4, pp. 965-979, 2012.
[34] C. Lai, H. Li, R. Lu, and X. Shen, “SE-AKA: a secure and efficient group authentication and key agreement protocol for LTE networks,” *Computer Networks*, vol. 57, no. 17, pp. 3492-3510, 2013.
[35] M. S. A. Khan, and C. J. Mitchell, “Improving air interface user privacy in mobile telephony,” *Cryptography and Security*, Cornell University Library, Apr. 2015. [Online]. http://arxiv.org/abs/1504.03287
[36] N. Saxena and N. S. Chaudhari, “A secure approach for SMS in GSM network,” in *Proc. CUBE*, 2012, pp. 59-64.
[37] N. Saxena and N. S. Chaudhari, “A secure digital signature approach for SMS security,” *International Journal of Computer Applications: SI on IP Multimedia Communications*, pp. 98-102, 2011.
[38] E. Oliver, Design and Implementation of a Short Message Service Data Channel for Mobile Systems, Technical Report CS-2007-42, 2007, 10 pages. [Online]. https://cs.uwaterloo.ca/research/tr/2007/CS-2007-42.pdf.
[39] A. Shamir, “How to share a secret,” *Communications of the ACM*, vol. 22, pp. 612-613, 1979.
[40] WLAN Pseudonym Generation for EAP-SIM/AKA, 3GPP TSG SA WG3 Security, S3-020654, Nov. 2002. [Online]. ftp://www.3gpp.org/tsg_sa/WG3_Security/TSGS3_26_Oxford/Docs/PDF/S3-020654.pdf.
[41] N. Saxena and N. S. Chaudhari, “VAS-AKA: an efficient batch verification protocol for value added services,” in *Proc. IEEE SMC*, 2013, pp. 1560-1565.
[42] A. Eshak, H. M. Harb, and M. Ezz, “Scalable intrusion detection system for cellular networks,” in *Proc. IEEE ASID*, 2013, pp. 1-8.
[43] A. Berkopec, “HyperQuick algorithm for discrete hypergeometric distribution,” *J. of Discrete Algorithms*, vol. 5, no. 2, pp. 341-744, 2007.
[44] K. Kurosawa, S. Obana, and W. Ogata, “t-cheater identifiable (k, n) threshold secret sharing schemes,” in *Proc. Advances in Cryptology*, LNCS 963, 1995, pp. 410-423.
[45] E. Kashefi, D. Markham, M. Mhalla, and S. Perdrix, “Information flow in secret sharing protocols,” *Electronic Proceedings in Theoretical Computer Science (EPTCS)*, vol. 9, pp. 87-97, 2009.
[46] L. Harn and C. Lin, “Detection and identification of cheaters in (t, n) secret sharing scheme,” *Designs, Codes and Cryptography*, vol. 52, no. 1, pp. 15-24, 2009.

Netesh Saxena is working as a Post-Doctoral Researcher at Georgia Institute of Technology, USA. Prior to this, he was with the State University of New York (SUNY) Korea, South Korea as a Post-Doctoral Researcher and a Visiting Scholar at Stony Brook University, USA. He earned his Ph.D. in Computer Science & Engineering from IIT Indore, India. His current research interests include smart grid and V2G security, cryptography, security in the cellular networks, and secure mobile applications.

Hong Shen is currently a tenured Professor (Chair) of Computer Science at University of Adelaide, Australia, and also “1000 People Plan” Professor at Sun Yat-sen University, China. He received the B.Eng. degree from Beijing University of Science and Technology, M.Eng. degree from University of Science and Technology of China, Ph.Lic. and Ph.D. degrees from Abo Akademi University, Finland, all in Computer Science. He was Professor and Chair of the Computer Networks Laboratory in Japan Advanced Institute of Science and Technology (JAIST) during 2001-2006, and Professor (Chair) of Compute Science at Griffith University, Australia, where he taught 9 years since 1992. He has published more than 300 papers including over 100 papers in international journals such as a variety of IEEE and ACM transactions. Prof. Shen received many honours/awards including China National Endowed Expert of “1000 People Plan” and Chinese Academy of Sciences “Hundred Talents”.



Nikos Komninos received his Ph.D. in 2003 from Lancaster University (UK) in Information Security. He is currently a Lecturer (US System: Assistant Professor) in Cyber Security in the Department of Computer Science at City University London. Prior to his current post, he has held teaching and research positions at the University of Cyprus, Carnegie Mellon University in Athens (Athens Information Technology), University of Piraeus, University of Aegean, and University of Lancaster. Since 2000, he has participated, as a researcher or principal investigator, in a large number of European and National R&D projects in the area of information security, systems and network security. He has authored and co-authored more than sixty journal publications, book chapters and conference proceedings publications in his areas of interest.

Kim-Kwang Raymond Choo is an Associate Professor of Cyber Security and Forensics at the University of South Australia, Australia, and a Visiting Expert at INTERPOL Global Complex for Innovation, Singapore. He received the Ph.D. in Information Security in 2006 from Queensland University of Technology, Australia. He was named one of 10 Emerging Leaders in the Innovation category of The Weekend Australian Magazine/Microsofts Next 100 series in 2009, and is the recipient of various awards including a Highly Commended Award in the 2014 Best Chapter in a Book Category by Australia New Zealand Policing Advisory Agency (ANZPAA), a Fulbright Scholarship in 2009, and a 2008 Australia Day Achievement Medallion.

Narendra S. Chaudhari has completed his undergraduate, graduate, and doctoral studies at IIT, Mumbai, India, in 1981, 1983, and 1988 respectively. He has shouldered many senior level administrative positions, a few notable assignments include: Dean - Faculty of Engineering Sciences, Devi Ahilya University, Indore, Coordinator - International Exchange Program, Nanyang Technological University, Singapore, and Dean - Research and Development, Indian Institute of Technology (IIT) Indore. He has more than 300 publications in top quality international conferences and journals. Currently, he is Director of VNIT Nagpur, India and also Mentor Director of newly established Indian Institute of Information Technology (IIIT), Nagpur, India. He is a Fellow and recipient of Eminent Engineer award (Computer Engineering) of IE-India, as well as Fellow of IETE-India, Senior Member of CSI and IEEE, and many other societies.