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 (MSs). Specifically, the proposed protocol securely transmits SMS from one MS to multiple MSs 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 (MSs), 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.

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 MSs. 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 Identifications (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 IMSIs 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, 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 MSs 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 MSs 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 MSs. 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→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 MSs, 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.

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Size (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MS</td>
<td>Mobile station referring user</td>
<td>–</td>
</tr>
<tr>
<td>UE</td>
<td>User Equipment referring user</td>
<td>–</td>
</tr>
<tr>
<td>AS</td>
<td>Authentication server referring AuC</td>
<td>–</td>
</tr>
<tr>
<td>IMSI</td>
<td>International mobile subscriber identity</td>
<td>128</td>
</tr>
<tr>
<td>TID</td>
<td>Temporary identity</td>
<td>128</td>
</tr>
<tr>
<td>ReqNo</td>
<td>Request number</td>
<td>8</td>
</tr>
<tr>
<td>SK</td>
<td>Shared secret key between MS and AS</td>
<td>128</td>
</tr>
<tr>
<td>DK</td>
<td>Delegation key generated from SK</td>
<td>128</td>
</tr>
<tr>
<td>H/MAC</td>
<td>Hash/message authentication code</td>
<td>64</td>
</tr>
<tr>
<td>T/T1</td>
<td>Timestamp</td>
<td>64</td>
</tr>
<tr>
<td>K</td>
<td>Random number</td>
<td>128</td>
</tr>
<tr>
<td>Y/P/Q/R</td>
<td>Variable</td>
<td>128</td>
</tr>
<tr>
<td>Z</td>
<td>Signature generated by the MS</td>
<td>128</td>
</tr>
<tr>
<td>SIMcode</td>
<td>SIM card activation code of SK key</td>
<td>64</td>
</tr>
<tr>
<td>S-Actcode</td>
<td>Sender generated code of the SK key</td>
<td>64</td>
</tr>
<tr>
<td>Actcode</td>
<td>Recipient generated code of the SK key</td>
<td>64</td>
</tr>
<tr>
<td>ExpT</td>
<td>Expiry time</td>
<td>64</td>
</tr>
<tr>
<td>f1(·)</td>
<td>HMACSHA256 is used to generate DK</td>
<td>–</td>
</tr>
<tr>
<td>fX(·)</td>
<td>AES-CTR is used to generate TID</td>
<td>–</td>
</tr>
<tr>
<td>f8(·)</td>
<td>HMACSHA1 is used to generate MAC</td>
<td>–</td>
</tr>
<tr>
<td>E{DK}</td>
<td>Encryption function with DK key</td>
<td>–</td>
</tr>
<tr>
<td>D{DK}</td>
<td>Decryption function with DK key</td>
<td>–</td>
</tr>
<tr>
<td>⊕</td>
<td>Bitwise XOR operation</td>
<td>–</td>
</tr>
</tbody>
</table>

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 MSs. 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 clear-text, 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 untraceability.

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.
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.

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

### 3.3 Detailed Description

Although the protocol is capable of supporting concurrent threads of different MSs sending their authentication requests with SMs to several different MSs. For simplicity, in this section, we present a scenario where a MS sends multiple SMs to different MSs. This scenario can be easily extended with multiple sender MSs. 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 MACs.

**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 $\in \mathbb{Z}_p^t$ (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 MSs 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 SMSs 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$_n\}$ such that: (i) knowledge of at least $k$ pieces of S-Actcode$_i$, helps AS in the computation required to generate the final SIMcode$_i$ that is similar to S-Actcode$_i$’s hash, and (ii) knowledge of any $k$-1 pieces of S-Actcode$_i$ cannot help in the reconstruction of the final SIMcode$_i$’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$_i$ after receiving at least $k$-S-Actcode$_i$. The AS will then match the computed hashed SIMcode$_i$ with the stored pre-computed hash of SIMcode$_i$ of the sender MS. Once the hashed SIMcode$_i$ is known to the AS, it retrieves SK$_i$ key and derives a delegation key DK$_i$ of the sender MS. This entire process takes $k$ points to define a polynomial of degree $k$-1 in a finite field $F$ 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 $b_0 = \text{SIMcode}_1$. 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:

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

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$_i = H(\text{SIMcode}_i)$ and is sent to the AS. We use first 64 bits of $H(\cdot)$ function as Actcode$_i$, which is SHA256.

At the AS: The AS pre-computes $H(\text{SIMcode}_i)$ from the stored SIMcode$_i$ for each MS$_i$ and then verifies Actcode$_i = H(\text{SIMcode}_i)$. 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$_i$. Also, in each such request, $n$ is randomly generated, which 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 $\text{TID}_i = f_2(\text{IMSI}_i, T_i)_{\text{DK}_i}$, to prevent the transmission of the original IMSI 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 $f_2(\cdot)$ is a reversible symmetric encryption function (e.g. AES-CTR). The structure of this function may be known; however, DK$_i$ 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 $K_i \in \mathbb{Z}_p^t$ (where $p$ is a large prime integer of 128 bits), generates current timestamp $T_i$, and derives a delegation key DK$_{ij}$ where $DK_{ij} = f_1(T_i)_{SK_j}$, and $f_1(\cdot)$ is a hash-based MAC function, such as HMACSHA256. Thereafter, each MS$_i$ computes $Y_i = K_i \oplus \text{IMSI}_i$ and a symmetric-signature $Z_i = (K_i + \text{DK}_i \oplus \text{RegNo}) \mod m$, 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 (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 → MS]: T_i, ReqNo, TID_i, S-Actcode_i, MAC1_i: The sender MS multicasts its authentication request as T_i, ReqNo, TID_i, S-Actcode_i, and MAC1 = f_3(T_i, ReqNo, TID_i, S-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 → AS]: T_i, ReqNo, TID_i, S-Actcode_i, MAC1_i, Actcode_i, TID_i, T_i, Y_i, Z_i, MAC2_i: On receiving the request, all recipients MS_i compute MAC1_i and verify MAC1_i = MAC_i. 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_i 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 (message-2), where MAC2_i = f_3(T_i, ReqNo, TID_i, S-Actcode_i, T_i, TID_i, T_i, Y_i, Z_i, Actcode_i).

**Step 3.** [AS → MS]: E{T_{m+1}, ExpT}\_DK_i: On receiving the message, the AS computes MAC2_i = MAC2_i and compares MAC2_i = 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 verifies IMSI_i. 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 \oplus 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-1}(DK_i \oplus IMSI_i) and R = \sum_{i=1}^{m-1}(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 \oplus R) and ensures that all legitimate MS_i are authenticated. Finally, the AS sends E{T_{m+1}, ExpT}\_DK_i to the sender MS (message-3), where new-ReqNo is a new request number assigned by the AS for subsequent request.

**Step 4.** [MS → AS]: E{T_{m+1}}\_DK_i: The MS replies E{T_{m+1}}\_DK_i as an acknowledgment to the AS (message-4).

**Step 5.** [AS → MS]: P_1, E{T_{m+2}, new-ReqNo, ExpT, new-SIMcode_i, DK_i}, MAC3_i: The AS decrypts the message as D(E{T_{m+1}}\_DK_i), and verifies T_{m+1}. Furthermore, the AS sends all P_i to the respective MS_i along with E{T_{m+2}, new-ReqNo, new-SIMcode_i, ExpT, DK_i}_DK_i, and MAC3_i (message-5), where MAC3_i = f_3(P_i, E{T_{m+2}, new-ReqNo, new-SIMcode_i, ExpT, DK_i}_DK_i). On receiving the message, all MS_i compute MAC3_i and compare MAC3_i = MAC3_i. If it holds, all MS_i compute P_i and compare P_i = 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 → MS]: E{T_1, new-ReqNo, new-SIMcode_i}_DK_i: Finally, the AS sends E{T_1, new-ReqNo, new-SIMcode_i}_DK_i, 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_i executes as follows:

**Step 7.** [MS → MS_i]: new-ReqNo, E{TID_i, T_j}_DK_i: The MS sends new-ReqNo, E{TID_i, T_j}_DK_i, to all respective MS_i (message-7).

**Step 8.** [MS_i → MS]: E{new-ReqNo, T_{j+1}}_DK_i: All MS_i check new-ReqNo, retrieve the corresponding DK_i from their memory, and decrypt the received message. Furthermore, if T_j ≤ ExpT, all respective MS_i compute another request number new-ReqNo_i = f_3(new-ReqNo, TID_i, 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_M. 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_M = 10% of the N_{MS}, i.e., 10. Let Prob\{t\} is the probability when t malicious
authentication requests are sent to the AS. The probability of Hypergeometric distribution [43] is as follows:

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

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.

We propose an algorithm to detect malicious requests of the MSs 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 \(Y; \) as explained in our protocol. Each invalid MSs is placed on a black-list and can only be removed once the predefined time is over. During this period, the request from particular MSs 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: \(|V| \geq (|S - \text{Actcode}| - 1)/\epsilon + 1\), where \(|V|\) exactly matches the above bound is said to be optimal. Let \(k = 2t + 1, p = 1/\epsilon\) and \(|S - \text{Actcode}| = p^j\), where \(i > 1\) and \(S - \text{Actcode} = (S - \text{Actcode}_1, S - \text{Actcode}_2, ..., S - \text{Actcode}_i)\) is a shared secret. We can identify up to \(i\) malicious users such that \(|V| = |S - \text{Actcode}|/\epsilon\) [45]. Now, we assume that \(j \leq t + 1\) 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 \((j - t)\) 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 ASs 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 ASs. 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 MSs are out of network, the AS will verify whether it has received at least \(k\) messages from different MSs. 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.

![Fig. 4: Reliability analysis of our protocol when t = [1-10].](image-url)
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\textsubscript{i} and the AS.

The BVPSMS protocol provides mutual authentications between the AS and the MS, and between the AS and the MS\textsubscript{i}. The AS authenticates all MS\textsubscript{i} by verifying \( \sum_{i=1}^{m} (Y_i = R) \) while each MS\textsubscript{i} authenticates the AS by comparing \( P_i = P'_i \). The sender MS authenticates the AS by decrypting the received message-3 using DK\textsubscript{i} 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\textsubscript{i} and the AS. In fact, Adversary \( A \) will not be successful in obtaining SK\textsubscript{K}/SK\textsubscript{i} or DK\textsubscript{K}/DK\textsubscript{i} key, even if it captures S-Actcode\textsubscript{K}/Actcode\textsubscript{i} of a MS.

A unique DK\textsubscript{i} key is used within the expiry of a session for each authentication between the AS and each MS\textsubscript{i}. A is unable to generate DK\textsubscript{K}/DK\textsubscript{i} key as it does not know the SK\textsubscript{K} key and the key generation function \( f_1() \). Since each S-Actcode\textsubscript{K}/Actcode\textsubscript{i} is sent over the network only once, the protocol is secure even if \( A \) is able to capture S-Actcode\textsubscript{K}/Actcode\textsubscript{i}. Moreover, \( A \) cannot derive any relation among captured S-Actcode\textsubscript{K}/Actcode\textsubscript{i} as SIMcode\textsubscript{K}/SIMcode\textsubscript{i} are randomly generated at the AS. Moreover, after each authentication, new-SIMcode\textsubscript{K}/new-SIMcode\textsubscript{i} are sent to each involved MS/MS\textsubscript{i}. Furthermore, if \( A \) modifies Actcode\textsubscript{i} in message-2, the computed MAC\textsubscript{2} cannot match with the received MAC\textsubscript{2} at the AS. Hence, the MS\textsubscript{i} will terminate the connection.

Property 3. Adversary A cannot trace the original identity of the MS/MS\textsubscript{i}. In fact, \( A \) is not able to identify the actual user, even if it captures the TID\textsubscript{i}/TID\textsubscript{o} of a mobile user.

Our protocol preserves identity anonymity and untraceability properties.

Untraceability: Our protocol satisfies untraceability as \( A \) cannot distinguish whether two TIDs correspond to the same MS/MS\textsubscript{i} or two different MS/MS\textsubscript{i}.

\[ \text{Verify(publicChannel)}[\text{TID}_i, \text{MS/MS}\textsubscript{i}, \text{AS}] \approx \text{Verify(publicChannel)}[\text{IMSI}\textsubscript{i}, \text{MS/MS}\textsubscript{i}, \text{AS}] \]

In our protocol, privacy of each MS\textsubscript{i} (including MS\textsubscript{K}) is ensured. Each TID\textsubscript{i} is computed from the original IMSI\textsubscript{i} as TID\textsubscript{i} = \( f_2(\text{IMSI}\textsubscript{i}, T_i) \) before a message is sent by each MS\textsubscript{i} over the network. We implement \( f_2() \) using AES-CTR with DK\textsubscript{i} key since no practical full attack has revealed against AES. As TID\textsubscript{i} is used by each MS\textsubscript{i} over the network, \( A \) is unable to trace the original identity of the user.

IND-ANO: Indistinguishability under Anonymous Identity: Our protocol is IND-ANO as no adversary \( A \) at time \( t \) can distinguish between two chosen identities TID\textsubscript{1} and TID\textsubscript{2} with a negligible \( \epsilon \) advantage.

\[ Pr[A(\text{TID}_1) = 1] - Pr[A(\text{TID}_2) = 1] \leq \epsilon \]

A cannot distinguish and relate TID\textsubscript{1} and other messages with ID\textsubscript{r}, as each TID\textsubscript{r} is used only once over the network. For all subsequent requests, a different new-ReqNo\textsubscript{r} is used each time when the sender MS connects to the MS\textsubscript{i}. The MS\textsubscript{i} sends an encrypted new-ReqNo\textsubscript{r} to the MS that will be used for the next authentication within a session. Hence, untraceability and identity anonymity are ensured, as \( A \) cannot trace TID\textsubscript{i}, SIMcode\textsubscript{i}, and new-ReqNo to link with users, and also IMSI\textsubscript{i} would not be revealed to \( A \) and intermediate operators.

Property 4. Adversary 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\textsubscript{i} (including MS) and the AS generate fresh DK\textsubscript{i} keys with unique timestamps, TID\textsubscript{i}, Actcode\textsubscript{i}, and K\textsubscript{i}. Therefore, \( A \) cannot retrieve the information based on linkability among users.

Forward Secrecy: Our protocol maintains forward secrecy as no \( A \) could obtain past keys and generate future keys.

The SK\textsubscript{i} and DK\textsubscript{i} keys are never sent over the network, and a new DK\textsubscript{i} key is used in each fresh session to encrypt IMSI\textsubscript{i} using AES-CTR. Even compromising current DK\textsubscript{i} will not allow \( A \) 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\textsubscript{i} key.

IND-CPA: Our protocol is IND-CPA secure as no adversary \( A \) in time \( t \) can distinguish between two chosen messages msg\textsubscript{1} and msg\textsubscript{2}, and has no or negligible advantage.

\[ Pr[D_{\text{R,SK}_i}([\text{msg}_1]) = 1] - Pr[D_{\text{R,SK}_i}([\text{msg}_2]) = 1] \leq \epsilon \]

Assuming that \( A \) 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 TID\textsubscript{i} generates TID\textsubscript{2} as \( f_2(\text{IMSI}\textsubscript{i}, T_i) \), 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\textsubscript{i} and the AS. Also, the protocol provides security protection over the air and SS7 channel. Furthermore, adversary \( A \) cannot compromise message confidentiality and integrity.

BVPSMS provides mutual authentication between the AS and the MS/MS\textsubscript{i} by verifying (\( \sum_{i=1}^{m} X_i = R \)) and \( P_i = P'_i \). 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. \( A \) is unable capture actual IMSI using IMSI catcher, as each MS/MS\textsubscript{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\textsubscript{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 \( T_{\text{receive}} \leq T_{\text{generate}} + T_{\text{threshold}} \) and
MACs 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 $A$.

In passive and active corruption attacks, $A$ obtains complete information held by the corrupted $MS_i$ (while a $MS_i$ still runs protocol correctly) and $A$ 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, $A$ 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/$TID_i$ of each others, as each $DK_i$ is secret. Also, $A$ cannot generate a valid symmetric-signature $S_i$, as it does not know the correct $SK_i$ and/or $DK$, 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)×$m$ + (128+64+8+64+64+128+64+64)×$m$ + (64+64+8+64) + (64+64+8+64+64+128)×$m$ + (8+64+64) = 336+1752×$m$ bits.

**Phase-2:** Total number of transmitted bits = ((7)+(8))×$r$ = (128+8+64)×$r$ + (64+8)×$r$ = 200×$r$. Total overhead = $42+(219+m)+(25+r)$ bytes.

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

<table>
<thead>
<tr>
<th>Protocols</th>
<th>Device-Server (bytes)</th>
<th>Intermediate Authority Server (bytes)</th>
<th>Server-Device (bytes)</th>
<th>Total (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABAKA [8]</td>
<td>84×$m$</td>
<td>–</td>
<td>80×$m$</td>
<td>164×$m$</td>
</tr>
<tr>
<td>RAISE [6]</td>
<td>200×$m$</td>
<td>–</td>
<td>–</td>
<td>200×$m$</td>
</tr>
<tr>
<td>SPECS [11]</td>
<td>48×$m$</td>
<td>96×$m$</td>
<td>32×$m$</td>
<td>176×$m$</td>
</tr>
<tr>
<td>b-SPECS+ [9]</td>
<td>48×$m$</td>
<td>176×$m$</td>
<td>64×$m$</td>
<td>288×$m$</td>
</tr>
<tr>
<td>BVPSMS*</td>
<td>97×$m$</td>
<td>–</td>
<td>57×$m$</td>
<td>154×$m$</td>
</tr>
<tr>
<td>BVPSMS**</td>
<td>25×$r$</td>
<td>9×$r$</td>
<td>–</td>
<td>34×$r$</td>
</tr>
</tbody>
</table>
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×$m$ bytes overhead, which is lowest among all the protocols discussed in the paper, while for all subsequent authentication requests, the overhead is only 34×$r$ 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, $MS_i = 11\times m$, and At the AS = 6×14×$m$.

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

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 the server. We performed various operations on an Intel Core i3-2390M 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 $T_{sub} = 0.009$ 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 $E()_{DK_i}$/$D()_{DK_i}$ 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 HMACSHA1 and HMACSHA256 are 512 bits each (actual input size plus

<table>
<thead>
<tr>
<th>Function</th>
<th>Ext (ms)</th>
<th>TUM (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1()$ = HMACSHA256</td>
<td>185</td>
<td>15204024</td>
</tr>
<tr>
<td>$E()_{DK_i}$/$f_2()$=AES-CTR</td>
<td>13.6</td>
<td>9139681</td>
</tr>
<tr>
<td>$D()_{DK_i}$/$f_3()$=AES-CTR</td>
<td>4.2</td>
<td>9124165</td>
</tr>
<tr>
<td>$f_1()$ = HMACSHA1</td>
<td>172</td>
<td>1521840</td>
</tr>
<tr>
<td>$H()$ = SHA256</td>
<td>20</td>
<td>14321156</td>
</tr>
</tbody>
</table>

Table 5: Computations of Various Used Functions
trailing zeros to make it multiple of 512). Also, the execution time of \textit{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:

\textbf{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.

\textbf{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:

\textbf{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. 

\textbf{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.

\textbf{BVPSMS Phase-1 (Time to verify):} MS\(_i\) by AS = 0.0282 + 391.55 \( \times m \) ms, MS by AS = 236.40 + 172.89 ms, MS 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.

\textbf{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 \( \times (m - 1 - t) \) + 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 \( E/D() \) 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 \( E/D() \) 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 / proverif 1.88
./proverif proofs/sms/BVPSMS.pv
 Query attacker(s[]) --> event(enableEnc)
 Completing...ok, secrecy assumption verified: fact unreachable
```
attacker(kims[11 = v_946])
Starting query attacker(s[]) ==> event(enableEnc)
RESULT attacker(s[]) == true.

- Query event(endMS(x1,x2)) == event(begMS(x1,x2))
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))
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[])
Starting query not attacker(DK[])
RESULT not attacker(DK[]) is true.

- Query not attacker(s[])
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 MSs 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 rebatch 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


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 (Information Technology), University of Piraeus, University of Aegaeon, 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.