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 for 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 other users, 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 of the mobile user. In the 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 protocol. BVPSMS, which provides end-to-end message security over an insecure communication channel between different mobile subscribers. Specifically, the proposed protocol securely transmits SMS from one mobile user to multiple users 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 proposed 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 then present a formal security proof of the proposed protocol. To the best of our knowledge, this is the first provably-secure batch verification protocol that delivers end-to-end SMS security 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], Short Message Service (SMS)-based web search [3] and end-to-end communications [4], [5]. 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 [6]. 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) [7], [8]. 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 [9] and SmartSMS [10] are the only available protocols in the literature that enable secure transmission of SMS from one MS to another [9]. However, 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 a number of situations where secure transmission of batch SMS can play a crucial role, such as sending urgent business meeting information to two or more 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, 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 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) [11]. We will address such security issues and consider them in the system and threat models.

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) [12], [13], [14], public-private key-based vehicular communication system [15], [16], [17], and digital signatures in batch to achieve high efficiency [18], [19]. Several SMS-based wireless protocols [20], [21], [22], [23], lightweight AKA [7] and SMS-based attacks and their countermeasures are discussed in [8]. Also, the protocols in [24], [25], [26], [27] are designed to provide SMS security based on asymmetric key cryptography with the exception in [26]. Other protocols in the literature include [28], [29] designed for the Global System for Mobile Communications (GSM), [30], [31], [32], [33], [34], [35] for the Universal Mobile Telecommunications System (UMTS), and [36], [37] 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 [38], [39]. 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) [40]. 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.

The SSM protocol [41] and the Secure Extensible and Efficient SMS (SEESMS) protocol [42], [43] do not provide end-to-end SMS security for mobile users, rather they provide security between a mobile user and the server, which is just a subpart of the problem of this paper and also was the same in EasySMS. These protocols in addition to the protocol in [44] do not focus the communication and computation overheads, prevention from attacks, and the bandwidth utilization. The additional drawback with SEESMS is that it was not suitable for resource constraint devices like mobile phones. There are other existing protocols as well for securing SMS between a mobile user and the server: SecureSMS [45], SK-SIM [46], and SMSSec [47], however, they do not provide SMS security between the end users. Similarly, the VAS-AKA [48] protocol does not provide end-to-end SMS security for mobile users, rather it secures the communication between a mobile user and the service-provision server in order to deliver value added services.

There are mainly two approaches of designing authentication system, namely peer-to-peer approach and server-based approach. The peer-to-peer security approach recommended in [49] is based on the public key cryptosystem, which requires a public-private key pair for each user. It does not require a central authority or a central server for authentication, but the overall processing is relatively slow in comparison to a symmetric key cryptosystem. On the other hand, [50] and [51] mentioned that the Authentication Server (AS) is a better strategy, as it provides faster processing and supports lightweight authentication for a large number of users. However, a single point of failure can be an issue if there is only one AS deployed in the system. In this paper, we selected the server-based approach for authenticating user requests as it supports the existing cellular infrastructure, where an AS is a part of the Authentication Center (AuC).

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 Batch Verification Authentication and Key Agreement Protocol (BVPSMS), which enables the transmission of an SMS to multiple recipients at any one time. The BVPSMS uses symmetric keys, since symmetric key encryptions are significantly faster than asymmetric key encryptions and as shown in [52], [53], they consume less energy. The proposed protocol has the following contributions:

1) The BVPSMS protocol:
   a) provides mutual authentication between the sender MS and the Authentication Server (AS), and between each recipient MS, and the AS.
   b) maintains message confidentiality and integrity using AES with Counter (AES-CTR) and Message Authentication Code (MAC), respectively, during messages transmission over an insecure network.
   c) 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+.

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.

1.4 Organization

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 5 and 6, respectively. Finally, section 7 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) [54], [55]: 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 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 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) 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
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>Z</td>
<td>Variable</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-Acctd</td>
<td>Recipient 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>HMAC-SHA-256 is used to generate DK</td>
<td>–</td>
</tr>
<tr>
<td>fA()</td>
<td>AES-CTR is used to generate TID</td>
<td>–</td>
</tr>
<tr>
<td>f2()</td>
<td>HMAC-SHA1 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>
<tr>
<td>→</td>
<td>Right arrow</td>
<td>–</td>
</tr>
</tbody>
</table>

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, such as A5/1 or A5/2. 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.

1) **Security Goals:** Our security goals are as follows:

   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. The half-open connection requests lead to flood-based Denial of Service (DoS) attacks [56]. There are cellular network protocols that protect the system against a DoS attack, such as Secure-AKA in UMTS network [57].

   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 \(f_3()\) 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 \(f_3()\) and \(E/D()\). However, inputs for \(f_2()\) and \(E/D()\) are different. Modern mobile devices are fairly capable of computing these functions [9]. The structure of these functions as follows:

\[ f_1() \text{ Function: A one-way function, such as one-way hash function HMACSHA-256, which takes input message of 512 bits with SK key and generates 256 bits of hash code, out of which first 128 bits are used as the DK key.} \]

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

\[ f_3() \text{ Function: It is used to generate MAC codes, which can be implemented by a one-way MAC function, such as HMAC-SHA1 that takes as input a multiple of 512 bits message with DK key and generates 160 bits of hash code, where the first 64 bits are used as MAC.} \]

\[ E/D()_{DK} \text{ Function: It is used to encrypt and decrypt the transmitted messages over the network. AES-CTR with DK key is used for this purpose. The Modified AES (MAES) [9] with 256 bits of DK key can also be used as an alternative. However, a key expand function is required to generate 256 bits of DK key from 128 bits.} \]

3.3 Detailed Description

Although the protocol is capable of supporting concurrent threads of different MSs sending their authentication requests with SMSs to several different MSs. For simplicity, in this section, we present a scenario where a MS sends multiple SMSs 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.

The proposed protocol uses a temporary identity (TID) for each user involved in the authentication process. The idea is to protect the user’s original identity (international mobile subscriber identity IMSI, and not a mobile number) over an insecure network. A mobile user can be traced by its IMSI number. A mobile number is used just to connect with a user, but it is never sent or exchanged over the network. In other words, this mapping to recognize the actual user takes place only with the IMSI number. Relevant authorities, such as safety and security department, can trace the original user with the help of the cellular providers, but an attacker or other legitimate users cannot trace a user over the network during communication.
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 ∈ ℤₚ (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 [58]. Hence, our scheme uses n ≤ 1.05 × k. The generation and transmission of activation code S-Actcode by the sender MS and retrieval of actual SIMcode by the AS are motivated by Shamir Secret Sharing Scheme [59] as follows:

The goal is to divide the hash of secret SIMcode of the sender MS into n-pieces as {S-Actcode₁, S-Actcode₂, ..., S-Actcodeᵢ} (i = 1, 2, ..., n) such that: (i) knowledge of at least k pieces of S-Actcodeᵢ helps AS in the computation required to generate the final SIMcode, say SIMcodeᵢ’s hash, and (ii) knowledge of any k-1 pieces of S-Actcodeᵢ cannot help in the reconstruction of the final SIMcodeᵢ’s hash (considering all possible values are equally likely). Therefore, the sender MS sends S-Actcodeᵢ to n-recipients MSᵢ. All n-MSs (in the ideal case) or at least k out of n-recipients MSᵢ (in case of error or network failure) forward their Actcodeᵢ to the AS along with the received S-Actcodeᵢ (part of sender MS). The AS obtains the actual hashed SIMcodeᵢ after receiving at least k-S-Actcodeᵢ. The AS will then match the computed hashed SIMcodeᵢ with the stored pre-computed hash of SIMcodeᵢ of the sender MS. Once the hashed SIMcodeᵢ is known to the AS, it retrieves SK₁ key and derives a delegation key DK₁ of the sender MS. This entire process takes k points to define a polynomial of degree k-1 in a finite field ℙ of size p where 0 < k ≤ n < p, SIMcodeᵢ < p, and p is a large prime.

The sender MS chooses at random k-1 positive integers {b₁, b₂, ..., bᵦ₋₁} with b₁ < p, and computes a polynomial

\[ f(x) = b₀ + b₁x + b₂x² + ... + bᵦ₋₁x^{k-1}, \]

where b₀ = SIMcode₁. The sender MS generates n S-Actcodeᵢ points \((xᵢ, yᵢ)\) as \((i, f(i) \mod q)\) using the Lagrange basis polynomial, where \(q > n, q > bᵢ\). On receiving the message (from at least k-recipients MSᵢ), the AS reconstructs a polynomial by computing \(f(x)\) as:

\[ f(x) = \sum_{i=1}^{k} yᵢlᵢ(x), \] where \(lᵢ(x) = \prod_{1≤j≤k \atop j≠ᵢ} (x - xᵢ)/(xᵢ - xⱼ). \]

Finally, the AS retrieves the actual SIMcodeᵢ (= b₀) from the computed \(f(x)\). In our protocol, each recipient MSᵢ also generates its own Actcodeᵢ as follows:

\[ E\{\text{new-ReqNo}, \text{new-SIMcodeᵢ}, Tᵢ\} \]
At the MSi: Each MSi generates Actcodei = H(SIMcodei) and is sent to the AS. We use first 64 bits of H() function as Actcodei, which is SHA256.

At the AS: The AS pre-computes H(SIMcodei) from the stored SIMcodei for each MSi, and then verifies Actcodei = H(SIMcodei). Thereafter, the AS extracts SKi key and derives DKi key by referring SIMcodei of each MSi. The delegation keys are special keys that represent the corresponding users and are derived from the original keys. This idea prevents the original keys from being accessed by adversaries over the network. In short, the original keys cannot be derived and comprised by the unauthorized users.

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 SIMcodei. Also, in each such request, n is randomly generated, which is at least 1.05 x k. For example, we can divide the hash of secret SIMcodei into twenty parts (n = 20) of S-Actcodei, and any fifteen parts (k = 15) can sufficiently reconstruct the original SIMcodei. Note that the construction of SIMcodei by an adversary is useless, as it cannot derive or extract meaningful information from SIMcodei and the information sent over the network. Later, in our protocol after verifying sender MS and all recipients MSj, the AS sends a new new-SIMcodei and new-SIMcodej to the MS and all MSj, 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 [60], 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 MSi (including sender MS) computes TIDi as TIDi = f2(IMSIi, Tij)DKi to prevent the transmission of the original IMSI over the network that protects ID-theft, eavesdropping, and MITM attacks. Here, Tij is the current timestamp, DKi is a delegation key, and f2() is a reversible symmetric encryption function (e.g., AES-CTR). The structure of this function may be known; however, DKi key remains secret.

3. Protocol Initialization: Let m be the total number of authentication requests generated by various mobile users MSi (where i = 2, 3, ..., m+1) to the AS at the same time when they receive a request from the sender MS. Initially, each MSi (and sender MS) chooses a random number Kp ∈ Zp (where p is a large prime integer of 128 bits), generates current timestamp Tij, and derives a delegation key DKi, where DKi = fi(Tij)SKi, and fi() is a hash-based MAC function, such as HMACSHA-256. Thereafter, each MSi computes Yij = Kp ⊕ IMSIi and a symmetric-signature Zij = (Kp + DKi ⊕ ReqNo) mod m, where ⊕ 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 MSi (and sender MS) and the AS know the corresponding SKi key as well as the generated DKi key.

4. Protocol Execution: This phase proceeds the execution of the proposed protocol in two parts: a fresh batch authentication and the subsequent authentications. A batch authentication process verifies the identities of the sender and all receiver MS simultaneously. If a batch process contains malicious MS and does not successfully verifies the identities of all users, a re-batch authentication will be performed after detecting the malicious users in the batch. Once a batch authentication is successful, the verified sender and receiver users can directly communicate with each other using a delegation key with a specified expiry time.

Phase-1: Batch Authentication: The proposed protocol performs the following six steps for a batch authentication:

Step 1. [MSi → MSj: Tij, ReqNo, TIDj, S-Actcodej, MAC1j]: The sender MS sends its request as {timestamp Tij, ReqNo, temporary identity TIDj, activation code S-Actcodej, MAC1j} to all targeted MSi (message-1), where MAC1j = fi(Tij, ReqNo, TIDj, S-Actcodej) for each MS. Thereafter, the AS extracts the hashed Zij value (from the received S-Actcodej) and computes MAC1j′ = (MAC1j ⊕ MAC1j). If it verifies, then the respective MSi procedure; otherwise, the connection is terminated by the MSi. The MSi who successfully verifies MAC1j, computes and sends their activation codes Actcodej, temporary identity TIDj, timestamps Tij, variables Yi and Zij, and MAC2j to the AS along with message-1 received from the MS except MAC1j (message-2), where MAC2j = fi(Tij, ReqNo, TIDj, S-Actcodej, Tij, Yi, Zij, Actcodej).

Step 2. [AS → MSi: Tij, ReqNo, TIDj, S-Actcodej, MAC1j, Actcodej, TIDj, Yij, Zij, MAC2j]: On receiving the request, all recipients MSi compute MAC1j′ and verify MAC1j′ = MAC1j′. If it verifies, then the respective MSi proceeds; otherwise, the connection is terminated by the MSi. The MSi who successfully verify MAC1j, compute and send their activation codes Actcodej, temporary identity TIDj, timestamps Tij, variables Yi and Zij, and MAC2j to the AS along with message-1 received from the MS except MAC1j (message-2), where MAC2j = fi(Tij, ReqNo, TIDj, S-Actcodej, Tij, Yij, Zij, Actcodej).

Step 3. [AS → MSj: E{Tm+1, ExpT}DKj]: On receiving the message, the AS computes MAC2j′ for all the received messages from different MSi and compares MAC2j′ = MAC2j′. If the verification returns false, the AS terminates the connection for the MSi.

Re-batch Authentication Process: In a re-batch authentication, the AS finds all invalid MSi using a detection algorithm “Malicious Requests Detection” (discussed in Section 4) and removes all invalid MSi from the batch. After removing malicious MSi from a
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 [14], [48].

**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 the one presented in [61] in the cellular network, can identify the suspicious malicious users. Let \( N_M \) 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 \( N_M \) is the maximum capacity of the AS to authenticate requests at any point of time. For the statistical analysis, we assume that \( N_M = 100 \), \( N_M = 50 \), and \( N_M = 10 \) of the \( N_M \), i.e., 10. Let \( \text{Prob}(t) \) be the probability when \( t \) malicious authentication requests are sent to the AS. The probability of Hypergeometric distribution [62] is as follows:

\[
\text{Prob}(t) = \frac{\binom{N_M - t}{N - t}}{\binom{N_M}{N}}, \quad \text{where} \quad t = 1, 2, \ldots, 10.
\]

This indicates that \( (N_M - t) \) valid requests are sent out of \( (N_M - N_M) \). Figure 4 shows the probability of Hypergeometric distribution when \( N_M = 100, N_M = 50 \), and malicious requests are \( t = 1, 2, 3, \ldots, 10 \). 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 \( k \). This might not hold if the sender mobile user is dishonest.

**Algorithm 1 Malicious_Requests_Detection**

Input: The AS receives a set \( \text{AR} \) of \( m \)-authentication requests \( \{R_1, R_2, R_3, \ldots, R_m\} \) at any time.

Output: Returns a set of malicious requests \( \text{MR} \), 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_{m/2}\};
        AR_2 = \{R_{m/2+1}, R_{m/2+2}, R_{m/2+3}, ..., R_m\};
        batch verification (AR_1, \lfloor m/2 \rfloor);
        batch verification (AR_2, \lceil m/2 \rceil);
        if (m == 1 && batch verification (AR, m) != 1) then
            returns MR = \{IMSI_i\}
```

Fig. 4: Reliability analysis of our protocol when \( t = [1-10] \).
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 MSi 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 ($S$-$Actcode_i$, $i = 1, 2, ..., t$), 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 [63], it is shown that in any such scheme, the following lower bound must be satisfied: $\lfloor V \rfloor \geq \lfloor |S-Actcode| - 1 \rfloor / \epsilon + 1$, where $\lfloor |V| \rfloor$ 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 $S-Actcode = (S-Actcode_1, S-Actcode_2, ..., S-Actcode_t)$ is a shared secret. We can identify up to $t$ malicious users such that $\lfloor V \rfloor = |S-Actcode|/\epsilon$ [64]. 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 [65]. 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.

One of the challenges of the cellular networks is to provide a reliable and secure service to a mobile user when he/she moves to a roaming area. The proposed protocol works only if the AS receives at least $k$ messages from the participated users. 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, and 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 one or more 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 the sender MS to restart its request.

## 5 Security Analysis

This section describes provable security of the proposed protocol by achieves the security goals outlined in Section 2.2 along with a formal verification of the proposed protocol. Provably secure aims to give an guarantee that the proposed protocol is secure under a system model (described in Section 2.1) and cannot be broken by a class of adversaries under a threat model (described in Section 2.2).

The proposed protocol will not affect network and service control compliances with the relevant regulation in different countries with respect to spam and marketing. A sender can only send an SMS to a receiver if the receiver already agreed to participate and the identities of both users are verified by the authentication server. In most of these services, the recipient knows the actual mobile number (individual or service number) of the sender. The proposed protocol hides the actual identity of each user over the network. Hence, it is unlikely that spam messages could be sent using this protocol. This section further analyses different security properties fulfilled by the proposed protocol.

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

The BVPSMS protocol provides mutual authentication between the AS and the MSi, and between the AS and the MSs. The AS authenticates all MSs by verifying $Y_i = Y_i$ while each MSi authenticates the AS by comparing $P_i = P_i$. The sender MS authenticates the AS by decrypting the received message-3 using $DK_i$ while the MS is authenticated by the AS by verifying $T_{m+1}$. $MS \equiv MS_i \equiv AS \rightarrow MS \equiv MS_i \land MS \equiv AS$.

**Property 2. Secure Session Keys.** The protocol initiates a secure session key establishment between all MS/MSi, and the AS. In fact, Adversary $A$ will not be successful in obtaining $SK_i/SK_i$ or $DK_i/DK_i$, 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 MSi. $A$ is unable to generate $DK_i$ as it does not know the $SK_i$, key and the key generation function $f_i()$. Since each $S$-$Actcode_i/Actcode_i$ is sent over the network only once, the protocol is secure even if $A$ is able to capture $S$-$Actcode_i/Actcode_i$. Moreover, $A$ cannot derive any relation among captured $S$-$Actcode_i/Actcode_i$, as SIMcodei/SIMcodei are randomly generated at the AS. Moreover, after each authentication, new-SIMcodei/new-SIMcodei are sent to each involved MS/MSi. If $A$ modifies Actcodei in message-2, the
computed MAC2 will not match with the received MAC2 at the AS. Hence, the MS will terminate the connection.

**Property 3. User Privacy.** Adversary A cannot trace the original identity of the MS/MSi. In fact, A is not able to identify the actual user, even if it captures the TID1/TID2 of a mobile user.

If adversary A is able to retrieve the original identity (IMSI) of a user, it can perform an impersonation and session hijacking attacks [66]. Our protocol preserves identity anonymity and untraceability guarantee that besides the identity anonymity and untraceability properties. The user’s anonymity and untraceability guarantee that besides the user and the AS ∈ \{networks\}, no one including the operator: (i) can retrieve the actual identity of the user, and (ii) is able to identify previous sessions involving that user.

**Untraceability.** Our protocol satisfies untraceability as A cannot distinguish whether two TIDs correspond to the same MS/MSi or two different MS/MSi:

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

In our protocol, privacy of each MSi (including MS) is ensured. Each TIDi is computed from the original IMSIi as TIDi = f2(IMSIi, Ti)DKi before a message is sent by each MSi over the network. We implement f2() using AES-CTR with DKi key since no practical full attack has revealed against on AES. As TIDi is used by each MSi over the network, A is unable to trace the original identity of the user.

**Indistinguishability under Anonymous Identity.** Our protocol is indistinguishable under anonymous identity as no adversary A at time t can distinguish between two chosen identities TID1 and TID2 with a negligible ε advantage.

\[
Pr[A(TID_1) = 1] - Pr[A(TID_2) = 1] \leq \varepsilon.
\]

A cannot distinguish and relate TIDi and other messages with IDi as each TIDi is used only once over the network. For all subsequent requests, a different new-ReqNo is used each time when the sender MS connects to the MSi. The MSi sends an encrypted new-ReqNo 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 TIDi, SImcodei, and new-ReqNo to link with users, and also IMSIi would not be revealed to A and intermediate operators.

**Property 4. Defeating Linkability and IND-CPA Attacks.** A cannot link current session information with previous sessions. Our protocol maintains perfect forward secrecy and Indistinguishability under Chosen Plaintext Attack (IND-CPA). The protocol achieves fairness and guarantees that “no MS is malicious or legitimate has an advantage” and maintains correctness under honest, semi-honest, and dishonest majority scenarios.

The MSi (including MS) and the AS generate fresh DKi keys with unique timestamps, TIDi, Actcodei, and Ki. Therefore, A cannot retrieve the information based on linkability.

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

The SKi and DKi keys are never sent over the network, and a new DKi key is used in each fresh session to encrypt IMSIi using AES-CTR. Even compromising current DKi will not allow A to obtain or generate past and future keys. Also, the past keys cannot be used for future sessions, as endpoints generate a fresh DKi key.

**IND-CPA:** Our protocol is IND-CPA secure as no adversary A in time t can distinguish between two chosen messages msg1 and msg2, and has no or negligible advantage.

\[
\frac{Pr[D_{K'i \leftarrow SK_i'} A(msg_1) = 1]} - \frac{Pr[D_{K'i \leftarrow SK_i'} A(msg_2) = 1]} \leq \varepsilon.
\]

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 MSi generates TIDi as f2(IMSIi, Ti)DKi, where Ti changes for each fresh message. We use AES-CTR as f2() 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.

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/MSi and the AS learn each others’ information. However, the MS and the MSi cannot learn any information about each other, as one user is unable to obtain DKi keys belonging to other users. Users are also unable to derive IMSIi/TIDi of each others, as each DKi is secret. Also, A cannot generate a valid symmetric-signature Si as it does not know the correct SKi and/or DKi keys, and Ki is randomly generated by each MSi. Our protocol also maintains IND-CPA; therefore, no MSi has an advantage over others. The proposed protocol fairly works under all three scenarios. We consider these scenarios only for the MSi, not for the AS. The reason is that the AS keeps SKi keys of all MSi 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 time discussed in Section 6.2.3. Our protocol maintains security properties under these scenarios, such as IND-CPA, forward secrecy and fairness.

**Property 5. Defeating Other Security Attacks.** Our protocol defeats SMS disclosure, SMS spoofing, replay, MITM, and impersonation attacks between the MS/MSi, and the AS. The protocol provides security protection over-the-air and SS7 channel. A cannot compromise message confidentiality and integrity. The protocol is secure against both passive and active corruption attacks in the presence of non-adaptive and/or adaptive adversaries.

BP2SMS provides mutual authentication between the AS and the MS/MSi by verifying \((\sum_{i=1}^{m} X_i) \oplus R_i = P_i \oplus P'_i\). It prevents the system against impersonation attack. Transmitted messages are securely encrypted using AES-CTR, which protects the system against SMS disclosure and MITM attacks. A is unable capture initial IMSI using IMSI catcher as each MS/MSi sends its TID over the network. It also prevents SMS spoofing as A cannot impersonate the server as a legitimate user because TID changes each time a user communicates. Furthermore, a timestamp value sent with each message protects the system against replay attack. Our protocol provides end-to-end SMS security from the MS to all MSi over OTA interface and SS7 channel, as each confidential message is encrypted using AES-CTR.
with a 128-bit key. Moreover, message integrity (message content and its threshold delivery in time) is maintained, as \( T_{receive} \leq T_{generate} + T_{threshold} \) and MACs are used for verification and the messages received after will be lapsed.

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.

Table 2 lists the security and privacy requirements achieved by the existing protocols. These protocols are secure against MITM attack, but do not provide integrity protection to the messages with the exception of RAISE [12]. However, RAISE [12] does not provide mutual authentication and is only partially secure against impersonation attacks. We observe that user privacy is preserved in ABACA [14] and RAISE [12], but both SPECS [17] and b-SPECS+ [15] suffer from replay attack. Our proposed protocol, on the other hand, fulfills all the mentioned requirements.

### 6 Performance Evaluation

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

#### 6.1 Analysis

This subsection analyzes the performance of the BVPSMS protocol. We compare the communication overhead generated by RAISE [12], ABACA [14], SPECS [17], and b-SPECS+ [15] 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., \( \text{ExpT} \). 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 = 
\[
(1)+(2)+(3)+(4)+(5)+(6) = (128+64+8+64+64)\times m + (128+64+8+64+64+64+128)\times m + (8+64+64) = 336+1752\times m 
\]

**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 except RAISE [12] 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 ABACA, 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 ABACA, RAISE, SPECS, and b-SPECS+, respectively, when \( r = 5, 10, 20, 50, 100 \).

In the proposed protocol, the receiver MS will not be liable for any cost. Only the sender MS will pay the cost for sending the SMSs. Technically, there is no limitation of sending or receiving SMSs in the modern cellular infrastructure, assuming the upload and download data speeds.

### Table 2: Protocols Comparison

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutual Authenti-</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>cation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Privacy</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Integrity</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Replay Protection</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>MITM Attack</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Impersonation</td>
<td>Yes</td>
<td>Partial</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Table 3: Communication Overhead in Batch Authentication by Different 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>ABACA [14]</td>
<td>84 (\times m)</td>
<td>–</td>
<td>80 (\times m)</td>
<td>164 (\times m)</td>
</tr>
<tr>
<td>RAISE [12]</td>
<td>200 (\times m)</td>
<td>–</td>
<td>200 (\times m)</td>
<td></td>
</tr>
<tr>
<td>SPECS [17]</td>
<td>48 (\times m)</td>
<td>96 (\times m)</td>
<td>32 (\times m)</td>
<td>176 (\times m)</td>
</tr>
<tr>
<td>b-SPECS+ [15]</td>
<td>48 (\times m)</td>
<td>176 (\times m)</td>
<td>64 (\times m)</td>
<td>288 (\times m)</td>
</tr>
<tr>
<td>BVPSMS*</td>
<td>97 (\times m)</td>
<td>–</td>
<td>57 (\times m)</td>
<td>154 (\times m)</td>
</tr>
<tr>
<td>BVPSMS**</td>
<td>25 (\times r)</td>
<td>9 (\times r)</td>
<td>–</td>
<td>34 (\times r)</td>
</tr>
</tbody>
</table>
are sufficiently fast. For example, 4G LTE offers typical
download speeds of 14Mbps and 8Mbps, respectively, whereas 4G LTE-Advanced provides typical real
world download speeds of 42Mbps and upload speeds of 30Mbps [67]. At the time of this research, voice over LTE
provides the fastest platform for sending an SMS [68]. The
proposed protocol is efficient and practical as long as
the authentication server is able to handle a batch process
efficiently. The delivery of each SMS is handled by the SMS
center, which keeps any undelivered SMS for 24 hours.
Practically, it can handle a very large number of undelivered
SMSs in the system.

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 computations at
the MS, MSi, and AS are as follows:

**Phase-1:** At MS = 8, MSi = 11 × m, and AS = 6 + 14 × m.

**Phase-2:** At MS = 2 × r and MSi = 2 × r.
Total computation overhead = 2 + (11 × m) + (6 + 14 × m) + 
(2 × r) + (2 × r) = 14 + (25 × m) + (4 × r) bits.

We compute the communication and computation over-
heads (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. The computation
overhead of the proposed protocol is much lower than the
communication overhead, therefore, the proposed protocol
achieves lightweight computations and can support a large
number of mobile users to be authenticated in a batch. This
also indicates that our protocol is efficient even when a large
number of subsequent 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/MSi 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 Tadd = 0.0009 milliseconds (ms), Tcor = 0.03 ms, and
Tsub = 0.0009 ms, respectively. We setup the system with 50
MSs, and one MS) transmitting their messages to the server
AS, when the MS sends an SMS to these MSs. 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/MSi and the AS.
Table 5 shows our 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 f2i
is implemented as AES-CTR, where encryption (generation of TIDi) took 13.6 ms and decryption (generation of IMSi)
performed in 4.2 ms. The AES is a secure algorithm till
date and CTR mode provide parallelism to speed up the
execution. The same results are obtained for E{D}DKi
using AES-CTR. The f1i and f3i are implemented as
HMACSHA-256 and HMAC-SHA1, respectively. The output
of HMAC-SHA1 and HMACSHA-256 are truncated to 64 and
128 bits, respectively because the output of f3i is 64 bits
MAC, whereas the output of f1i is 128 bits, which is DKi
key. The input to the HAMCSHA1 and HMACSHA-256 are
512 bits each (actual input size plus 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.

**TABLE 5: Computations of Various Used Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Ext (ms)</th>
<th>TUM (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>f1i() = HMACSHA-256</td>
<td>185</td>
<td>15204024</td>
</tr>
<tr>
<td>E{DKi}</td>
<td></td>
<td>AES-CTR</td>
</tr>
<tr>
<td>D{DKi}</td>
<td></td>
<td>AES-CTR</td>
</tr>
<tr>
<td>f1i() = HMAC-SHA1</td>
<td>172</td>
<td>15211840</td>
</tr>
<tr>
<td>H() = SHA256</td>
<td>20</td>
<td>14321156</td>
</tr>
</tbody>
</table>
Total execution time of a single authentication:
Phase-1: Total time = transmission time for all messages in phase-1 + time at the entities (MS, MSi, AS) = 2.98 sec. Hence, on average the execution time per user is 1.49 sec.
Phase-2: Total time = transmission time for all messages in phase-2 + time at the MS, MSi, AS = 10.7+35.6 = 46.3 ms.

6.2.2 Verification Time
The verification delay in our protocol is evaluated between the MS/MSi and the AS. It is the time estimation between the sent messages and the received response or the protocol completion.

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

**BVPSMS Phase-2** (Time to verify): MS by MSi = 17.8×r ms, and MSi by MS 4.2×r ms.
Total verification delay in phase-2 = 32×r ms.
Therefore, total verification delay in **BVPSMS** = 240.62+736.47×m+32×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 MSi. After detecting the malicious MSi, 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 = 0.00933×3×(m−1−t) + 0.030322 + 0.00933 = 0.028456+0.002799×(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, 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, 10. 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 f3() and f2() are implemented as HMAC functions. The output of HMAC-SHA1 and HMACSHA-256 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. Furthermore, 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×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 f2() and E/D{|}DKi, 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 f2() and E/D{|}DKi in a single authentication (2 operations) as well as batch authentication (2×m operations). The IMSIi and TIDi of 128 bits each also need to be stored in the memory. Furthermore, the storage is also required for HMAC-SHA1, HMACSHA-256, and AES-CTR at the MS/MSi as well as at the AS. For a re-batch verification, O(1) is only the extra cost need to be paid (for 3×m−3×t+2 operations). Therefore, the **BVPSMS** protocol is an efficient, secure, and cost effective protocol that requires less storage.

7 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 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.
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 A queries: (i) Can A successfully recover confidential and useful information from the messages sent over the network?, (ii) Can A successfully compute parameters generated by the MS?, (iii) Can A successfully compute parameters generated by the AS?, (iv) Can A successfully generate DK key of the MS?, and (v) Can A successfully recover secret key of the MS?. Following is the syntax and output observed from the Proverif tool.

\[\text{let } \text{dect1ms1:bitstring} = \text{sdecrypt(enct1ms1,DKms1)} \text{ in }\]
\[\text{encnewsimcode1:bitstring, enct1ms1:bitstring); }\]
\[\text{in(pubChannel, } (\text{MSG6, enctm1ms1as:nonce, timsi, mac2mias2:mac, enctexpmsi:nonce, mac2msias2:mac, enct1ms1:nonce, tims1, mac1msias2:mac, enctm1ms1:nonce, tims1, mac1ms1as:mac); }\]
\[\text{let } \text{dectexpmsi:nonce} = \text{sdecrypttime(enctexpmsi,DKmsi)} \text{ in }\]
\[\text{new newrno:bitstring; new q:bitstring; new i:bitstring; }\]
\[\text{let } \text{timsi:iden} = \text{f2(imsimsi,timsi,DKmsi)} \text{ in }\]
\[\text{let } \text{DKms1:sskey} = \text{f1(tims1,skms1)} \text{ in }\]
\[\text{new imsimsi:ident; new skms1:skey; get keys(=MSi, skms1) in }\]
\[\text{in(PubChannel, } (\text{xA1: host, xB1: host}); }\]
\[\text{let } \text{processMSi} = \text{in(PubChannel, } (\text{xA2: host, xB2: host}); }\]
\[\text{if } xA2 = \text{MS } \text{and } xB2 = \text{MS then }\]
\[\text{in(pubChannel, } (\text{MSG1, tims1:nonce, rnomsi:bitstring, tidsms1:iden, s-acctcodei:bitstring, mac1ms1as:mac); }\]
\[\text{new tims1:nonce; new simcodei:bitstring; new ki:bitstring; }\]
\[\text{new insmimsi:iden; new skmsi:skey; get keys(=MSi, skmsi) in }\]
\[\text{let DKmsi:sskey = f1(timsi,skmsi) in }\]
\[\text{new timsi:nonce; new simcodei:bitstring, mac1ms1:mac); }\]
\[\text{let } \text{processMS} = \text{in(PubChannel, } (\text{xA1: host, xB1: host}); }\]
\[\text{if } xA1 = \text{MS } xB1 = \text{MS then }\]
\[\text{new insmims1:iden; new skms1:skey; get keys(=MS, skms1, skms1 in }\]
\[\text{let DKms1:sskey = f1(tims1,skms1) in }\]
\[\text{new tims1:nonce; new simcodei:bitstring, mac1ms1:mac); }\]
\[\text{let } \text{processMS} = \text{in(PubChannel, } (\text{xA1: host, xB1: host}); }\]
\[\text{if } xA1 = \text{MS } xB1 = \text{MS then }\]
\[\text{new insmims1:iden; new skms1:skey; get keys(=MS, skms1, skms1 in }\]
\[\text{let DKms1:sskey = f1(tims1,skms1) in }\]
\[\text{new tims1:nonce; new simcodei:bitstring, mac1ms1:mac); }\]
\[\text{let } \text{processMS} = \text{in(PubChannel, } (\text{xA1: host, xB1: host}); }\]
\[\text{if } xA1 = \text{MS } xB1 = \text{MS then }\]
\[\text{new insmims1:iden; new skms1:skey; get keys(=MS, skms1, skms1 in }\]
\[\text{let DKms1:sskey = f1(tims1,skms1) in }\]
\[\text{new tims1:nonce; new simcodei:bitstring, mac1ms1:mac); }\]
\[\text{let } \text{processMS} = \text{in(PubChannel, } (\text{xA1: host, xB1: host}); }\]
\[\text{if } xA1 = \text{MS } xB1 = \text{MS then }\]
\[\text{new insmims1:iden; new skms1:skey; get keys(=MS, skms1, skms1 in }\]
\[\text{let DKms1:sskey = f1(tims1,skms1) in }\]
\[\text{new tims1:nonce; new simcodei:bitstring, mac1ms1:mac); }\]
\[\text{let } \text{processMS} = \text{in(PubChannel, } (\text{xA1: host, xB1: host}); }\]
\[\text{if } xA1 = \text{MS } xB1 = \text{MS then }\]
\[\text{new insmims1:iden; new skms1:skey; get keys(=MS, skms1, skms1 in }\]
\[\text{let DKms1:sskey = f1(tims1,skms1) in }\]
\[\text{new tims1:nonce; new simcodei:bitstring, mac1ms1:mac); }\]
\[\text{let } \text{processMS} = \text{in(PubChannel, } (\text{xA1: host, xB1: host}); }\]
\[\text{if } xA1 = \text{MS } xB1 = \text{MS then }\]
\[\text{new insmims1:iden; new skms1:skey; get keys(=MS, skms1, skms1 in }\]
\[\text{let DKms1:sskey = f1(tims1,skms1) in }\]
\[\text{new tims1:nonce; new simcodei:bitstring, mac1ms1:mac); }\]
\[\text{let } \text{processMS} = \text{in(PubChannel, } (\text{xA1: host, xB1: host}); }\]
\[\text{if } xA1 = \text{MS } xB1 = \text{MS then }\]
\[\text{new insmims1:iden; new skms1:skey; get keys(=MS, skms1, skms1 in }\]
\[\text{let DKms1:sskey = f1(tims1,skms1) in }\]
\[\text{new tims1:nonce; new simcodei:bitstring, mac1ms1:mac); }\]
\[\text{let } \text{processMS} = \text{in(PubChannel, } (\text{xA1: host, xB1: host}); }\]
\[\text{if } xA1 = \text{MS } xB1 = \text{MS then }\]
\[\text{new insmims1:iden; new skms1:skey; get keys(=MS, skms1, skms1 in }\]
\[\text{let DKms1:sskey = f1(tims1,skms1) in }\]
\[\text{new tims1:nonce; new simcodei:bitstring, mac1ms1:mac); }\]
\[\text{let } \text{processMS} = \text{in(PubChannel, } (\text{xA1: host, xB1: host}); }\]
\[\text{if } xA1 = \text{MS } xB1 = \text{MS then }\]
\[\text{new insmims1:iden; new skms1:skey; get keys(=MS, skms1, skms1 in }\]
\[\text{let DKms1:sskey = f1(tims1,skms1) in }\]
\[\text{new tims1:nonce; new simcodei:bitstring, mac1ms1:mac); }\]
\[\text{let } \text{processMS} = \text{in(PubChannel, } (\text{xA1: host, xB1: host}); }\]
\[\text{if } xA1 = \text{MS } xB1 = \text{MS then }\]
sencrypt(newsvimodel,DKmsia) in
let endMSfull(AS, MSi, enctmsi); out(pubChannel,MSG6,encnewrni,encnewsimcode1, enctmsi); event endMSi(MSi, AS);
end event(endMSi(MSi, AS, enctmsi)); (* Key registration *)
let processK = inj-event(beginMSfull(x1,x2)) is true
if h < MS && h < AS then insert keys(h,k).
(* Start process *)

new skms1: skkey; new skms2: skkey;
insert keys(MS, skms1); insert keys(MS, skms2);
((processMS) | (processMS) | (processAS) | (processK))

Output: — Query attacker(s) ==> event(endMS)
Completing...ok, secrecy assumption verified: fact unreachable

RESULT attacker(s) ==> event(endMS) is true.
— Query inj-event(endMS(x1,x2)) ==> event(begMS(x1,x2))
Completing...ok, secrecy assumption verified: fact unreachable

RESULT inj-event(endMS(x1,x2)) is true.
— Query inj-event(beginMSfull(x1,x2)) ==> event(endMSfull(x1,x2))
Completing...ok, secrecy assumption verified: fact unreachable

RESULT inj-event(beginMSfull(x1,x2)) is true.
— Query inj-event(beginMS(x1,x2)) ==> event(endMS(x1,x2))
Completing...ok, secrecy assumption verified: fact unreachable

RESULT inj-event(beginMS(x1,x2)) is true.
— Query inj-event(begMS(x1,x2)) ==> event(endMS(x1,x2))
Completing...ok, secrecy assumption verified: fact unreachable

RESULT inj-event(begMS(x1,x2)) is true.

ACKNOWLEDGMENTS
This work was supported by TCS, India and Australian Research Council Discovery Projects funding #DP150103871. K.-K. R. Choo is supported by the Cloud Technology Endowed Professorship.

REFERENCES


