Received July 21, 2018, accepted August 14, 2018, date of publication August 20, 2018, date of current version November 30, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2866063

AITSteg: An Innovative Text Steganography Technique for Hidden Transmission of Text Message via Social Media

MILAD TALEBY AHVANOOEY[1], QIANMU LI[1], JUN HOU[1,2], HASSAN DANA MAZRAEH[3], AND JING ZHANG[1]

[1]School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China
[2]School of Marxism Studies, Nanjing University of Science and Technology, Nanjing 210094, China
[3]School of Mathematics and Computer Science, Damghan University, Damghan 36715-364, Iran

Corresponding authors: Milad Taleby Ahvanooey (taleby@njust.edu.cn) and Qianmu Li (qianmu@njust.edu.cn)

This work was supported in part by the National Key Research and Development Programming under Grant 2016YFE0108000, in part by the CERNET Next Generation Internet Technology Innovation Project under Grant NGII20160122, in part by The Project of ZTE Cooperation Research under Grant 2016ZTE04-11, in part by the Jiangsu Province Key Research and Development Program: Social Development Project under Grant BE2017739, in part by the Jiangsu Province Key Research and Development Program: Industry Outlook and Common Key Technology Projects under Grant BE2017100, in part by the National Natural Science Foundation of China under Grant 61374186, and in part by the 2018 Smart Jiangsu Project—Jiangsu Province Women+ Smart Cloud Platform Project (Jiangsu Provincial Women's Federation undertakes research), and in part by the Nanjing University of Science and Technology for research works during the academic year 2017/2018.

ABSTRACT With the popularity of smartphones and widespread use of high-speed Internet, social media has become a vital part of people's daily life. Currently, text messages are used in many applications, such as mobile chatting, mobile banking, and mobile commerce. However, when we send a text message via short message service (SMS) or social media, the information contained in the text message transmits as a plain text, which exposes it to attacks. In some cases, this information may be confidential, such as banking credentials, secret missions, and confidential appointments; moreover, it is a major drawback to send such information via SMS or social media, as neither provides security before transmission. In this paper, we propose a novel text steganography technique called AITSteg, which provides end-to-end security during the transmission of text messages via SMS or social media between end users. The AITSteg technique is evaluated by considering a trusted scenario. We then evaluate the efficiency of the proposed technique in terms of embedding capacity, invisibility, robustness, and security. The experiments confirm that the AITSteg is able to prevent various attacks, including man-in-the-middle attacks, message disclosure, and manipulation by readers. Moreover, we compare the experimental results with the existing techniques to show the superiority of the proposed technique. To the best of our knowledge, this is the first text steganography technique that provides end-to-end secure transmission of the text message using symmetric keys via social media.

INDEX TERMS Text steganography, chat hiding, covert communication, text hiding, social media.

I. INTRODUCTION

The growth in Internet and smartphone use has had a major social and commercial impact on daily life. These new technologies benefit people all over the world, and allow information to be stored, processed, and accessed in an inexpensive and widely accessible manner. Since the text message has become a popular and easy form of communication, concerns about data leakage attacks, such as hacking, hijacking, and phishing, have emerged. Users such as detectives, journalists, judges, and election officials rely on short message service (SMS) or Social Media Applications (*SMAPP*) to communicate with each other [1]–[3]. Smartphone users typically send confidential information such as banking credentials (e.g., account details, passwords, and transaction information), secret missions, confidential appointments, and private identities to family members via text messages using SMS or SMAPPs. However, the standard SMS service and social media amazingly do not provide security to this type of digital data being transmitted over networks. In this case, it is required to provide secure communication between smartphone users. Since text messages via SMS and SMAPPs on smartphones are so common, cyber-attacks such as man-in-the-middle (MITM) [3], message disclosure [4], and manipulation by readers (MBR) [2] are a concern. Functionally, text messages are sent as plaintext between smartphone users and service providers (the SMS service center, social media service provider, etc.) using an available network. Text message content is stored by service providers in servers that are easily accessed. Over the last two decades, many techniques have been proposed to improve the efficiency of text steganography in digital texts [6]–[12], but these methods have low embedding capacity and low robustness against distortion attacks. In other words, they are not able to embed a high capacity of secret information through a short message.

The main contributions of this paper are threefold:

- First, we state three types of cyber-attacks that have recently brought a whole new dimension to cyberspace. In addition, we investigate how to address these attacks.
- Second, we propose a novel text steganography technique called AITSteg, which ensures the security of the message during transmission via SMS or SMAPPs. The AITSteg protects confidential information by generating a hidden message (*HM*) containing the confidential information and encoding it using a formulated symmetric key. Afterward, it embeds the generated *HM* in front of a seemingly innocent cover message (*CM*), which is used as a cover.
- Finally, we develop an app based on the AITSteg technique and conduct practical experiments to evaluate it by employing fifteen social media and messenger apps as communication channels. Examples are used to analyze and compare the proposed technique with existing techniques based on common criteria.

The remainder of the paper is organized as follows. Section II introduces some background information and related works on text steganography. Section III presents the proposed technique in detail. Section IV describes the experimental results of the proposed technique conducted on SMAPPs and compares them with existing techniques. Section V presents a formal proof of the AITSteg technique. Finally, Section VI concludes the paper with a summary of research contributions and future work.

II. RELATED STUDIES

This section surveys various studies and related works on text steganography. First, a brief description of cryptography and information hiding, the Unicode standard, and the evaluation criteria is provided, and then text steganography techniques from the literature are described. Table 1 represents the definition of various symbols used in the paper.

A. CRYPTOGRAPHY AND INFORMATION HIDING

Secure communication techniques can be classified into two main branches, cryptography and information hiding.

TABLE 1. The abbreviations.

Abbreviations	Description
SMS	Short message service
SMAPP	Social media app (WeChat, WhatsApp, Imo, etc.)
App	Smartphone application
DS	Database servers of service providers
MITM	Man-In-The-Middle
MD	Message disclosure
MBR	Manipulation by readers
SM	Secret message
ZWC	Zero width character
CM	Cover message
HM	Hidden message
CM_{HM}	Carrier message (HM+CM)
EC	Embedding capacity
LP	Losing probability
DP	Decoding probability
BPL	Bit per locations
EL	Embeddable locations
TWSM	Text watermarking in social media
NELRS	Number of embeddable locations required to embed one secret character
NCRES	Number of cover characters required to embed a secret character

Generally, cryptography scrambles a plain text (or secret text) into cipher text that is reversible without data loss. The goal of cryptography is to prevent unauthorized access to the secret message (*SM*) by scrambling the original form of its content. On the other hand, information hiding is a powerful programming technique that hides an *SM* under cover of an innocent media (e.g., text, image, video and audio or other supports such as network communications) for the purpose of secure communication [6], [9], copyright protection [7], tracking [8], etc. This technique is divided into two major types, steganography and watermarking. Both offer a variety of methods to hide information in the cover media in an invisible or irremovable way, where the hidden information is reversible by the corresponding technique. Steganography is the technique of hiding an *SM* in a cover media to transmit the *SM*; therefore, the main concern is how to conceal it without raising the suspicion of human vision systems. Watermarking is concerned with hiding information in cover media such that the hidden data are robust to alterations and adjustments. Practically, steganography and cryptography are two techniques of secure transmission over the Internet. Cryptography scrambles a message to conceal its contents; steganography conceals the existence of an *SM*. However, cryptography provides optimum security for communication systems but has a fundamental flaw in that it raises the suspicions of attackers. However, when information hiding is used, even if an attacker is suspicious of the transmitted messages, he cannot discover the existence of hidden messages since it is carried out in an invisible way. The main limitation of cryptography is that the third party is always aware of the communication because of the unknown nature of the text. Steganography overcomes this limitation by hiding the existence of the *SM* in the cover media [5]–[10].

TABLE 2. Unicode zero width characters.

Character Name	Hex Code	Written Symbol
Zero-Width-Non-Joiner	0x200C	No symbol and width
POP Directional	0x202C	No symbol and width
Left-To-Right Override	0x202D	No symbol and width
Left-To-Right Mark	0x200E	No symbol and width

B. THE UNICODE STANDARD

In computing systems, the Unicode standard was defined in 1987 to process and display digital texts. Basically, all software systems have to support the Unicode standard for the representation of digital texts. The Unicode standard is a universal character encoding system designed to support the worldwide display, processing, and interchange of texts with different languages and technical disciplines. In addition, it also supports classical and historical characters of many languages. This standard is compatible with the latest version of ISO/IEC 10646-1:2017 and has the same characters and codes as those in ISO/IEC 10646. As of June 2017, the latest version of Unicode is 10.0.0, and it is maintained by the Unicode Consortium. It includes three encoding forms, UTF-8, UTF-16, and UTF-32, for which Unicode allows 17 planes, each with 65,536 possible characters (or 'code points'). Thus, it gives a total of 1,114,112 possible characters in formats such as digits, letters, symbols, emoticons, and a huge number of current characters in various languages. Currently, the most commonly used encoding forms are UTF-8, UTF-16 and the now outdated UCS-2. UTF-8 provides one byte for any ASCII character, which have the same code values in both ASCII and UTF-8 encoding, and up to four bytes for other characters. UCS-2 provides a 16-bit code unit (two 8-bit) for each character but cannot encode every character in the current Unicode standard. UTF-16 extends UCS-2, using one 16-bit unit for the characters that were representable in UCS-2 and two 16-bit units (4 × 8-bit) to process each of the further characters [5].

These days, Unicode is required in new Internet protocols (e.g., TCP/IP, FTP, HTTP, and SMTP) and implemented in all modern operating systems (e.g., Android, iOS, Windows Phone and BlackBerry) and programming languages for processing digital texts. In Unicode, there are specific zero-width characters (*ZWC*) that are used to control special entities such as Zero Width Non Joiner (e.g., ZWNJ separates two letters in special languages) and POP directional, which have no written symbol or width in digital text [5]–[9]. In social media, if it utilizes the Unicode standard in order to process digital texts in different languages, then the ZWCs show invisible written symbols; otherwise, they might generate unconventional symbols [10]. We used four ZWCs for hiding the *SM* through the *CM*, which are depicted in Table 2.

C. TEXT STEGANOGRAPHY EVALUATION CRITERIA

There are many considerations that must be taken into account when researchers design a text steganography algorithm. However, the common criteria can be easily found in recently proposed techniques: invisibility, embedding capacity, robustness, and security [5]–[12].

1) INVISIBILITY OR IMPERCEPTIBILITY

The trace of a hidden *SM* in the CM_{HM} must be invisible and avoid raising the suspicions of human vision system. In other words, invisibility refers to how much perceptual alterations are made in the CM_{HM} after embedding an *SM*. Practically, it cannot be measured numerically. The best way of measuring the degree of imperceptibility is to compare the variation of *CM* and CM_{HM} before and after embedding the *HM* [5], [6].

2) EMBEDDING CAPACITY(EC)

The amount of secret bits that can be embedded in a *CM* is termed the embedding capacity. This feature can be measured numerically in units of bit-per-locations (*BPL*). Locations are the number of embeddable locations (*EL*) in the *CM* where the method can insert the *HM* between words, after special characters, etc. However, a steganography algorithm provides a large embedding capacity, it is not useful if it alters the *CM* profoundly [5], [8].

$$EC = BPL \times EL \tag{1}$$

3) ROBUSTNESS

In general, many attacks may occur on the CM_{HM} while it is transmitted on the communication channels where it may be exposed to a hazard that could destroy the *HM* [5]. Moreover, malicious users may try to manipulate the *HM* from the CM_{HM} rather than remove it. Thus, any kind of distortion might occur deliberately or even unintentionally on the CM_{HM}. A robust steganography algorithm makes the *HM* extremely difficult to alter or destroy. It can also be measured numerically based on losing probability (*LP*) [6]–[8]. It is assumed that *LP* is the probability of how much proportion of the embedded secret bits has been lost from the CM_{HM}. The lower losing probability leads to a more robust steganography algorithm.

Let's assume that the number of embeddable locations in the *CM* is *NL*, the length of the *CM* is stand as *TC*, and then the distortion robustness (*DR*) can be calculated as follows.

$$DR = [1 - LP] \tag{2}$$

where $1 < NL < TC, NL \in \mathbb{N}, TC \in \mathbb{N}$.

$$LP = \frac{NL}{TC} \tag{3}$$

4) SECURITY

There is a certain level of safety that prevents attackers from detecting the *HM* visually or from removing it from the CM_{HM} [5]. This measure depends on three other criteria, invisibility, embedding capacity, and robustness. An efficient steganography technique must provide optimum balance among these criteria. If an algorithm provides a large *EC*, the trace of embedding the *SM* is totally invisible, and robustness is high, then the security of the algorithm is equal to the following formula.

$$P_{Security} = [1 - DP] \tag{4}$$

- *Hash Function* is a mathematical method that maps data of arbitrary size to a binary string of a fixed length that is designed to be a one-way function, that is, a function that it is infeasible to invert. It can be employed to protect the secret bits before embedding in the cover media. In practice, it alters the sequence of the secret bits such that they can only be extracted by the corresponding hash function [6]. In other words, this technique makes the secret bits difficult to discover via decoding attacks.
- *Decoding Probability (DP)* is the probability of decoding the *SM* binary string from the CM_{HM} by attackers [9]. It is assumed that, an attacker speculates a message may include an *HM* (e.g., he/she does not have any clue about the approach that is employed to conceal the SM). In addition, the attacker may try to decode the *SM* using conventional approaches or guessing the *SM* binary (using probability distribution analysis) from the invisible symbols or features. Since a hash function is used to map the *SM* binary (or secret bits) based on a key, and NS is the length of the SM binary, the DP can be calculated as follows.

$$DP = \sum_{i=i|k}^{NS} \left(\frac{1}{2^i}\right)^{\frac{NS}{i}}, \; i : \exists_{k \in \mathbb{N} | i \times k = NS}, i \in [1, NS], i \in \mathbb{N} \tag{5}$$

D. RELATED WORKS

Previously, various researchers have introduced several approaches to provide secure transmission of confidential information using data hiding in cover media. Most of the existing literature has focused on data hiding under cover of images, videos, and audios [12]–[17]; however, relatively few techniques have been proposed to hide confidential information in the cover text. Recently, text hiding, where secret information is embedded in the cover text, has drawn considerable interest from security researchers. Since text messages have a limited number of words and symbols, it is difficult to change the text content to hide data through the short cover text. Basically, text messages include handwritten styles such as special phrases, short-hand acronyms, and emoticons [11], [12]. A text-based data hiding technique called UniSpaCh is presented in [7], which generates a binary string of the *SM* and isolates it by 2-bit classification (i.e., "10, 01, 00, and 11"). Moreover, it replaces each 2-bit by a special space (e.g., Thin, Hair, Six-Per-Em, and Punctuation). Finally, it embeds the generated spaces into special locations such as inter-words, inter-sentences, end-of-lines, and inter-paragraphs within the cover text. However, this approach provides high invisibility through the cover text but has low embedding capacity (two bits per spaces) and is not applicable to embedding long secret bits in short cover text. A new algorithm of text steganography called AH4S is introduced in [9], which applies the structure of the omega network to hide an *SM* in a generated *CM*. It selects a letter from the *SM* and uses the omega network to produce two related letters according to a selected letter and, moreover, searches the dictionary for a proper English cover word to conceal the two produced letters and repeats the same process for all letters of the *SM*. For example, to hide "A", it generates a long unknown text as a *CM* according to "A", and then, it embeds two width-spaces through the generated cover text to hide the positions of the two generated letters through the *CM*. In practice, it produces a long unknown text for a short SM and raises suspicions for the readers as well. An efficient text watermarking technique (*TWSM*) for hiding secret data in short Latin based text is proposed in [10], which employs the homoglyph Unicode characters and special spaces in order to embed the secret bits in the Latin based *CM*. Based on the experimental results, it can be concluded that this technique provides optimum embedding capacity, high invisibility, and low robustness against distortion attacks. Sometimes, smartphone users employ emoticons in daily conversations instead of typing their feelings. Several researchers have utilized emoticons to hide an *SM* through the text message. For example, [11], and [12] generate a random text including some words as a *CM*, and in addition, they convert all characters of the *SM* to emoticons based on a predefined pattern (e.g., A="angry", B="sad", C="happy", etc.) and, thus, embed emoticons between words in the *CM*. Although emoticon-based text steganography techniques have high invisibility, they suffer from low robustness against distortion attacks. In the techniques explained above, it is not clear whether the introduced approaches are able to prevent the *SM* against various attacks. Thus, in this paper, we compare the proposed technique with the existing UniSpaCh [7], AH4S [9], TWSM [10], and emoticon-based [12] techniques. More information on traditional text steganography techniques can be found in [5] and [18]–[20].

III. SECURITY GOALS AND PROPOSED TECHNIQUE

This section focuses on the attack model, motivating scenario and detail description of the proposed technique.

A. ATTACK MODEL

An attack model describes various scenarios for the possibilities of different attacks, where an attacker might be able to access/discover the authentic information.

1) MESSAGE DISCLOSURE AND OTA ATTACKS

Since the text message is transmitted as plain text, service-provider operators (SPO) can easily access the message content during the transmission over the network or from the database servers. This leads to message disclosure attacks. Moreover, in the case of the cellular networks, the OTA interface between the base transceiver station (BTS) and mobile station (user) is protected by a weak encryption technique (e.g., A5/1 or A5/2), so an attacker can easily

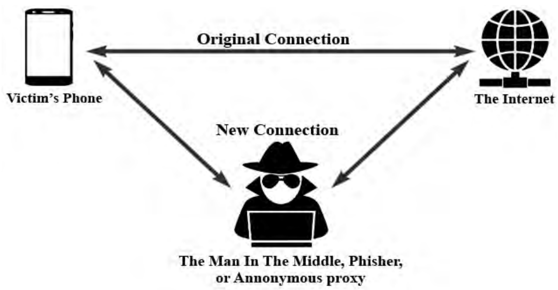

FIGURE 1. An illustration of the MITM attack.

decrypt these techniques to obtain the information contained in the SMS or manipulate message content or forge authentic messages [1], [2].

2) A MAN-IN-THE-MIDDLE (MITM)

This is a type of hidden trick where an attacker inserts him/herself into a conversation between two parties, impersonates both parties, and gains access to information by processing and eavesdropping on the data transmission over the network. In the *MITM* attack, a common scenario consists of two victims (endpoints) and an attacker (third party). The attacker has access to the communication channel between two victims, and can manipulate their messages. The MITM attack can be visualized as depicted in Fig. 1. This attack is very effective because of the property of the TCP and the HTTP protocol which are based on the ASCII or Unicode standard [1], [3], [21].

3) MANIPULATION BY READER (MBR)

This is a serious attack, in which a malicious user has access to the victim's device and tries to discover confidential information from the SMAPPs, i.e., the victim's device is compromised by an attacker. The main aim of this attack is to gain sensitive information or manipulate messages in the victim's device on behalf of its owner [2], [3]. For example, a malicious user can login to the account of another user in social media and monitor conversations.

B. MOTIVATING SCENARIO

To overcome the above stated attacks, we present a motivating scenario to demonstrate the research challenge of this paper. As depicted in Fig. 2, we suppose that Alice and Bob are sensitive users working with a messenger (SMS, SMAPPs, etc.) to communicate confidential information during an important mission. The problem is that the MITM attacker is eavesdropping on the data transmission to gain access to sensitive information. In the same trend, the SPO has access to the transmitted messages where they are stored in the database servers of service providers.

Furthermore, it is obvious that a safe conversation is required to guard against these attacks. To address this challenge, we propose a text steganography technique that ensures the security of confidential information while being transmitted via SMS or SMAPPs over the network.

Security measures require extra effort, and this is an additional cost borne by the smartphone user. Therefore, we develop an app based on the proposed technique that can be used by sensitive users to send confidential information via SMS or other SMAPPs. When a user wants to send an *SM* to another user, AITSteg is performed, which makes available a symmetric key between both users, and then the *SM* is hidden using a hash function.

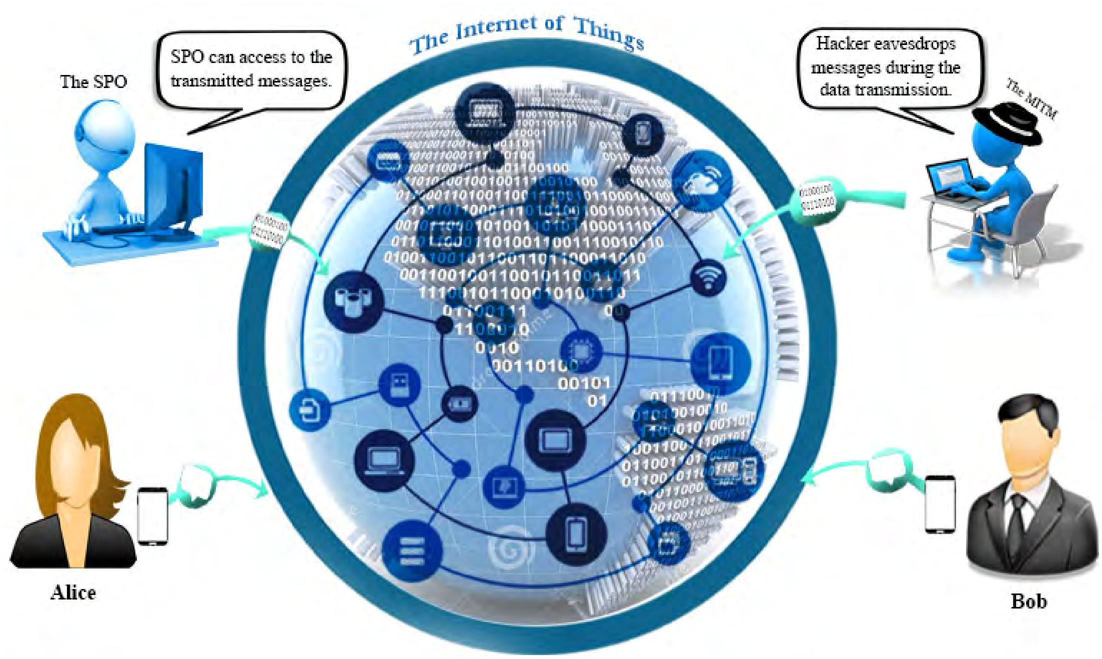

FIGURE 2. An illustration of Motivating Scenario.

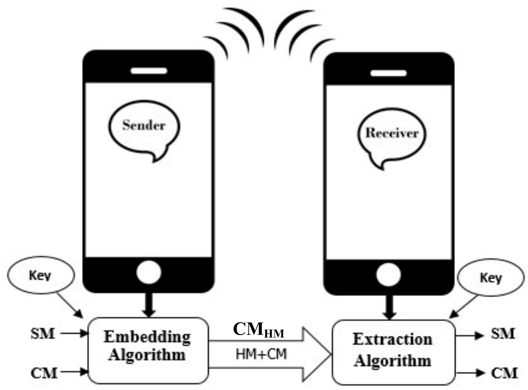

FIGURE 3. An illustration of the proposed mechanism.

C. THE PROPOSED TECHNIQUE: AITSteg

As shown in Fig. 3, the proposed technique has two main phases which are the embedding algorithm and extraction algorithm. Various elements of the proposed technique are explained in the following points.

- *Secret Message (SM):* This can be secret or confidential information.
- *Cover Message (CM):* This is an innocent text that can be any type of text message such as a greeting, a typical question, or a pranks.
- *Key:* This is used as a symmetric key (*SK*) while encoding/decoding the secret bits, which makes the *SM* difficult to discover (e.g., here, a formulated sending/receiving time is considered as the default SK).
- *Embedding Algorithm:* This converts the *SM* into a protected *HM* based on the *SK* and embeds it in front of the *CM*.
- *Carrier Message* (CM_{HM}) : This is the output message which includes the HM and CM.
- *Extraction Algorithm:* This extracts the HM from the CM_{HM}, and decodes/authenticates the *SM* by checking the received *SK*.
- *Message Sender (MS):* This employs the embedding algorithm to send confidential information via SMS or other SMAPPs.
- *Message Receiver (MR):* This utilizes the extraction algorithm to authenticate and extract the confidential information from the CM_{HM}.

Phase-1: The embedding algorithm includes three stages: (1) generating pairs of numbers for all the letters of the *SM*, (2) producing a hashed binary string according to the pairs of numbers and the *SK*, (3) replacing the hashed binary string based on successive 2-bits by the ZWCs (see Table 3) by an *HM* string, and embedding the *HM* in front of the *CM*.

Stage-(1): As the operating systems support the Unicode standard (or ASCII) to process digital texts, the MITM attackers exploit these encoding systems to process the Internet protocols. In the AITSteg, we propose a new encoding technique to change the default structure of the text processing used against MITM attacks. To meet that requirement, we employ an encoding technique on the SM letters, which is called the "Gödel" function (or numbering) [22]. We utilize the Gödel function to generate pairs of numbers for each letter of the *SM* based on its ASCII code. In the Gödel function, if η is any given number, there is a unique solution α, β. Let η be the ASCII code of the letter and α, β be the pairs of numbers that are unique for each letter of the *SM*.

TABLE 3. Unicode ZWCs 2-bit classification pattern.

2-Bit Classification	Hex Code
00	200C
01	202C
10	202D
11	200E

$$\langle \alpha, \beta \rangle = 2^{\alpha}(2\beta + 1) - 1 \tag{6}$$

where $2^{\alpha}(2\beta + 1) \neq 0$

$$\Rightarrow \langle \alpha, \beta \rangle = \eta.$$
$$\Rightarrow \eta = 2^{\alpha}(2\beta + 1) - 1.$$
$$\underset{\alpha \in \mathbb{N}}{Max}\left[\exists_{k \in \mathbb{N}} \left| 2^{\alpha} \times k = \eta + 1 \right.\right] \tag{7}$$

The equation (7) means: α is the largest number such that $2^{\alpha}|(\eta + 1)$, i.e., $\frac{\eta+1}{2^{\alpha}}$, must be odd.

$$\beta = [(\frac{\eta + 1}{2^{\alpha}}) - 1]/2 \tag{8}$$

For example, if η is "z"=122, α, β can be obtained as follows. First, we obtain α by calculating the largest number such that $2^{\alpha}|(122 + 1)$.

$$\underset{\alpha \in \mathbb{N}}{Max}[\exists_{k \in \mathbb{N}}|2^{0} \times 123 = 122 + 1] \Rightarrow \alpha = 0$$

Then, we calculate β by using equation (8) and the obtained α. $\Rightarrow \beta = [(\frac{122+1}{2^{0}}) - 1]/2 = 61$

The embedding algorithm computes $\langle \alpha, \beta \rangle$ for all the letters of the SM by using equation (6) and (7). After producing $\langle \alpha, \beta \rangle$ pairs of numbers, it converts α, β to a 6-bit binary string separately and joins them together to produce a 12-bit binary string for each letter; i.e., we used 12-bit to encode each letter because the largest $\langle \alpha, \beta \rangle$ is equal to $\langle 0, 63 \rangle = 126$, which requires 12-bit for binary representation). This coding system can support ASCII characters ranged from as many as 126 codes that include the letters of the English alphabet and some punctuation symbols. Finally, it generates an *SM* binary string from pairs of numbers.

Stage-(2): In the hash function, we proposed a new formulated symmetric key based on the sending/receiving time (e.g., it can be any symmetric key based algorithm such as AES, DES, RC5, etc.), which generates a dynamic key for the same SM in different times according to the sending/receiving time and the length of *SM*. First, the hash function obtains the sending time as an *MS_SK* (e.g., "12:15"), and then omits the 4th digit of the time and, in addition, it creates a number (e.g., "121"), and thus, converts it to an 8-bit binary string. Further, the hash function repeats the *SK* binary string while the length of the *SK* binary string is greater than or equal to the length of the *SM* binary string.

Let *LS* be the length of the SM_binary string, and *LSK* be the length of the SK binary string; NC is the number of copies required for making the hash position bits.

$$P = \begin{Bmatrix} 0 & Mod(LS, LSK) = 0 \\ 1 & else \end{Bmatrix}$$

$$NC = \frac{LS \times 12}{LSK} + P \tag{9}$$

After generating the SM binary string and the hash position bits, the hash function balances the SM binary string from the left side, and reverses the SM binary (bits) according to the hash position bits to produce the hashed SM binary.

Finally, the embedding algorithm generates a hidden message of SK (*HM_SK*) and an *HM* of the hashed *SM* binary by replacing each 2-bit by one ZWC based on Table 3. Thus, it embeds the *HM* in front of the *CM* and produces the CM_{HM}.

Phase-2: After receiving a CM_{HM}, the extraction algorithm discovers the contractual 2-bit of each ZWC and generates an *MS_SK* binary and a hashed *SM* binary string according to the sequence of *ZWCs* from the *HM*. Therefore, the hash function considers the receiving time as an *MR_SK*, and compares it with the extracted *MS_SK*. If it matches, the extraction algorithm reverses the hashed binary string based on the *MR_SK* binary and length of *HM* and produces an *SM* binary string.

Moreover, the extraction algorithm divides the *SM* binary string into 12-bit groups and converts each 12-bit into two 6-bit groups and calculates α, β. Thus, the hash function obtains each η according to its α, β. Finally, it generates the SM based on the η numbers.

For example, if the calculated $\langle \alpha, \beta \rangle$ is $\langle 0, 61 \rangle$, then $\eta = 2^0(2 \times 61 + 1) - 1 = 122$. After generating η for all α, β pairs of numbers, the extraction algorithm converts each η to its letter based on the ASCII standard (e.g., $\eta = 122$ ➔ "z").

Algorithm 1 Pseudocode of Embedding Algorithm

Input: a cover message (CM), a secret message (SM), and a symmetric key (*MS_SK*).
Output: a carrier message (CM_{HM}) which includes of *HM* and *CM*.

SM 🡐 *Secret Message;*
CM 🡐 *Cover Message;*
MS_SK 🡐 *Sending Time;*
for each $l_i \in SM = \{l_1, l_2, \ldots, l_n\}$ do
  η 🡐 Obtain ASCII Code of SM[l_i];
  α🡐 Calculate [$2^\alpha \times k = \eta + 1$];
  β🡐 Calculate [$[(\frac{\eta+1}{2^\alpha}) - 1]/2$];
  α_binary 🡐 *Convert (α to 6-bit);*
  β_binary 🡐 *Convert (β to 6-bit);*
  SM_binary 🡐 SM_binary +(α_binary +β_binary);
end for
LSK 🡐 *Length (MS_SK_binary);*
if (*Mod(length(SM_binary), LSK)==0*) **then** *P* 🡐 0;
  else *P*🡐*1;*
end if
NC 🡐 *[Length(SM_binary)/LSK]+P;*
Hash_position_bits 🡐 *NC times copy of MS_SK_binary*
Hashed_SM_binary 🡐 *XOR (SM_binary string based on Hash_positions_bits);*
HM_SK 🡐 Replace (each 2-bit of SK_binary (8-bit) by one ZWC);
HM 🡐 *HM_SK* + *Replace (each 2-bit of Hashed_SM_binary by one ZWC based on Table 3 classification pattern);*
Return CM_{HM} 🡐 HM+CM;

IV. EXPERIMENTAL RESULTS AND COMPARISONS

In this section, we evaluate the efficiency of the proposed technique in terms of the evaluation criteria. We have implemented the AITSteg in Java programming (e.g., Eclipse ADT 23.02X) and executed the experiments on various smartphones with Android OS. Moreover, we compare the experimental results with the existing techniques with respect to the evaluation criteria.

A. EVALUATION CRITERIA ANALYSIS

In this subsection, we evaluate the experimental results based on the evaluation criteria.

1) EMBEDDING CAPACITY (EC)

Since the embedding algorithm converts each letter of *SM* to six ZWCs to generate the CM_{HM}, if the SMS is used as a form of communication, it provides the *EC* by the max number of characters in SMS (e.g., 2048 '8-bit' characters for English alphabets, and 1024 '16-bit' for UTF-8 *ZWCs* characters). In case of other SMAPPs, where an app is used as the means of transmission of the CM_{HM}, the *HM* can be embedded by the max number of characters in the specific SMAPP. We have tested practically all the maximum text limits of SMAPPs, which are listed in Table 4.

2) INVISIBILITY

As shown in fig. 4, the AITSteg does not change the written symbols of CM_{HM} after embedding the *HM* because the ZWCs are used to hide the *SM* within the *CM*. To analyze this criterion, we have tested the CM_{HM} by sending it through the SMAPPS on mobile platforms such as Android, iOS, and Windows.

The experimental results showed that readers cannot detect the CM_{HM}, and no one can infer the existence of the *HM* using human vision systems. Moreover, we evaluated the invisibility of the *HM* by sending the CM_{HM} through the conventional *SMAPPs* listed in Table 5. Based on the results conducted on the SMAPPs, thirteen SMAPPs and messengers (except for 'Telegram' and 'Twitter') supported the Unicode *ZWCs* and

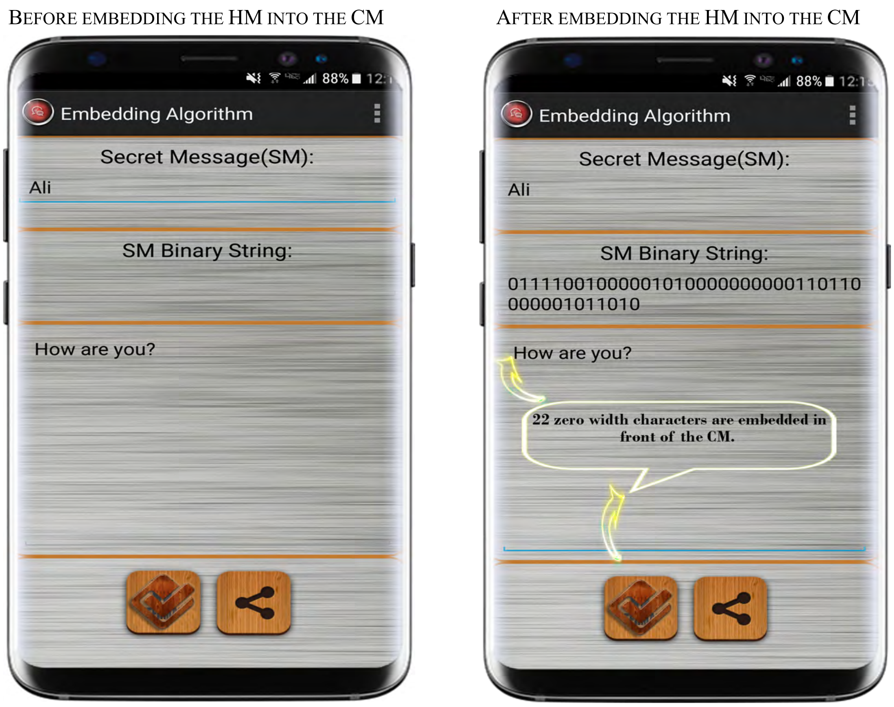

FIGURE 4. An illustration of embedding an SM in front of CM.

TABLE 4. Text message character limitation and EC of AITSteg.

Social Media	Text Limits (Number of Characters)	Maximize EC (HM Characters)
SMS	1024 (UTF-8)	170
Facebook	640 (UTF-8)	106
WhatsApp	30,000 (UTF-8)	5,000
WeChat	16207 (UTF-8)	2701
Skype	400 (UTF-8)	66
Yahoo Messenger	2048 (UTF-8)	341
Gmail	35,000,000 (UTF-8)	5,833,333
Imo	Virtually Unlimited (UTF-8)	Unlimited
QQ	16,207 (UTF-8)	2701
Viber	7000 (UTF-8)	1166
Hangouts	Virtually Unlimited (UTF-8)	Unlimited
Line	10,000 (UTF-8)	1666
Tango	520 (UTF-8)	86
Telegram	4096 (Exclusive encoding)	0
Twitter	140 (Exclusive encoding)	0

TABLE 5. Invisibility analysis of cm through SMAPPs.

APP Name	Invisible: Yes (✓) or Not (×)
SMS	✓
Facebook	✓
WhatsApp	✓
WeChat	✓
Skype	✓
Yahoo Messenger	✓
Gmail	✓
Imo	✓
QQ	✓
Viber	✓
Hangouts	✓
Line	✓
Tango	✓
Telegram	×
Twitter	×

allowed transmission of the CM_{HM} with high invisibility, so that viewers were only able to see the *CM*.

3) ROBUSTNESS

Basically, one of the most important criteria in steganography is the reversibility of the *HM* from the CM_{HM}. To meet this measure, we considered the embedding location of *HM* in front of the *CM* due to modification of the CM_{HM} content by attackers, which may lead to a lower probability of the *HM* being destroyed. Let *LCM* be the length of the *CM*; then, the LP of the *HM* can be obtained by using the equation (3): $LP = \frac{1}{LCM+1}$.

For example, if the *SM* is "Ali" and the *CM* is "How are you?", the *LP* can be obtained as follows: $LP = \frac{1}{13} \cong 0.0769$.

$$DR = [1 - LP] = [1 - 0.0769] = 0.9231 \times 100 = 92.31\%$$

Thus, if a malicious user removes or alters a part of the CM_{HM}, the *HM* will remain in front of the CM_{HM} with more probability and can be extracted by the AITSteg technique. During data transmission, the *MITM* attackers can monitor *ZWCs* through CM_{HM} and may try to discover/decode the confidential information. Because an encoding function and a dynamic symmetric key is applied for each *SM*, it is very

Algorithm 2 Pseudocode of Extraction Algorithm

Input: a carrier message (CM_{HM}), and a symmetric key (MR_SK).
Output: a secret message (SM).

```
CM_HM ← Carrier Message;
MR_SK ← Receiving Time;
  foreach l_i ∈ CM_HM = {l_1, l_2, ..., l_n} do
    switch (CM_HM[l_i]) {
      case 'u200C':
        Hashed_SM_binary ← Hashed_SM_binary
        +"00"; break;
      case 'u202C':
        Hashed_SM_binary ← Hashed_SM_binary
        +"01"; break;
      case 'u202D':
        Hashed_SM_binary ← Hashed_SM_binary
        +"10";    break;
      case 'u200E':
        Hashed_SM_binary ← Hashed_SM_binary
        +"11";    break;  }
   end for
      MS_SK ← Hashed_SM_binary.Substring(0,8);
   if (MR_SK==MS_SK) then
      Hashed_SM_binary ← Hashed_SM_binary.
      Substring(8);
      LSK ← Length (MR_SK_binary);
    if (Mod(length(Hashed_SM_binary), LSK)==0)
       then P ← 0;
     else P ← 1;
     end if
     NC ← [Length(Hashed_SM_binary)/LT]+P;
     Hash_position_bits ← NC times copy of
     MR_SK_binary;
     SM_binary ← XOR (Hashed_SM_binary based on
     Hash_position_bits);
  While (Length(SM_binary) >=12)
      AlfaBeta ← SM_binary.Substring(0,12);
      SM_binary ← SM_binary.Substring(12);
      Alfa ← AlfaBeta.Substring(0,6);
      Beta ← AlfaBeta.Substring(6,6);
      α ← Calculate the decimal number of (Alfa);
      α ← Calculate the decimal number of (Beta);
      η ← Compute [2^α(2β + 1) − 1];
      SM ← SM + (Convert η to its ASCII letter);
  end while
end if
Return SM
```

difficult to decode the *SM* from the *ZWCs* for knowledgeable attackers as well.

4) SECURITY

Since the AITSteg provides optimum balance between three other criteria - higher EC, high imperceptibility, and high robustness - it is able to perfectly secure the SM. Therefore, the eavesdropper (attacker) cannot decode or even discover the hidden information from the CM_{HM}, but he/she may try to decode the ZWCs from the CM_{HM}. It is assumed that the algorithm is not available to attackers, and he/she only has access to CM_{HM}. For a situation where an attacker compromises the victim's device since we defined a tricky feature that hides the AITSteg app in the Android OS, and the user must only run it by calling a number (e.g., "*110110"). Hence, the attacker cannot access to the AITSteg app.

Let NS be the length of the *SM* binary string, *NZ* (e.g., *NS/2* is the number of ZWCs that are used to generate the *HM*; thus, DP is the decoding probability of the correct guess of the Hashed SM binary that can be obtained by equation (4).

For example: as depicted in Table 6, the AITSteg converts SM="Ali" to an *SM* binary string that containing $NS = (3 \times 12) + 8 = 44$ and generates the hash position bits according to the *SK* as follows: if (LS=length (SM)=3, SK="12:13" = 121, SK_binary = "1111001", LSK = Length (SK_binary)=7, then, $NC = \frac{3\times 12}{7} + 1 = 6$.

An NC times copy of the *SK* binary is used to generate the hash position bits. Thus, the hash function reverses the *SM* binary string according to the hash position bits. Finally, the embedding algorithm replaces each 2-bit in the hashed *SM* binary by one ZWC based on Table 3. Finally, it generates an *HM* which includes "22" ZWCs, and inserts the *HM* in front of the *CM*.

$$P_{Security} = [1 - DP] = [1 - 3.4106e - 13] \cong 0.99$$

$$DP = \sum_{i=i|k}^{NS=44} (\frac{1}{2^i})^{\frac{44}{i}} = 3.4106e - 13, \; i \in [1, 44], \; i \in \mathbb{N}$$

Herein, the *DP* is the probability of decoding the hashed binary string; still, there are two encoding techniques that generate dynamic ZWCs even for the same *SM* in different sending/receiving times. In practice, it is impossible to decode the original SM without having the hash function and formulated symmetric keys. The probability distribution of *DP* regarding the number of characters of *SM* and *CM* is illustrated in Fig. 5.

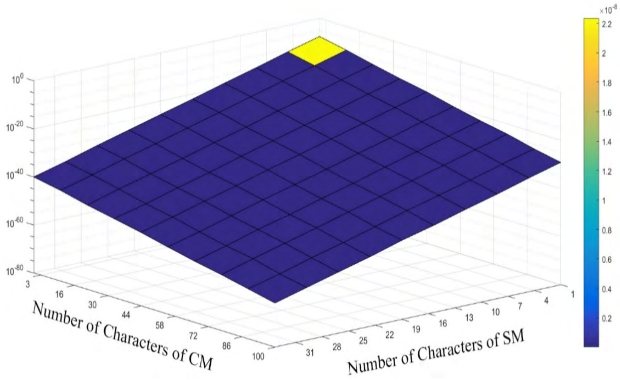

FIGURE 5. The probability distribution of DP.

TABLE 6. An example of embedding process in detail.

Elements	Values
Secret Message (SM)	Ali
Cover Message(CM)	How are you?
SM's pairs of numbers	[“A”=65=<1, 16>], [“l”=108=<0, 54>], [“i”=105= <1, 26>]
SM binary string	“000001010000” +“000000110011”+ “000001011010”
Symmetric Key (MS_SK)	“12:13” => *MS*_SK=121, *MS*_SK binary =”1111001”, NC=6
Hash positions bits (NC times copy of MS_SK binary)	”01111001”+”1111001”+”1111001”+”1111001”+”1111’001”+”1111001”
MS_SK_binary + Hashed SM binary string	“01,11,10,01” + “11,11,01,10,11,10, 01,11,00,00,11,01,01,11,10,01,01,01”
Unicode ZW characters of the MS_SK+HM	“202C+200E+202D+202C”+“200E+200E+202C+202D+200E+202D”+“202C+200E+ 200C+ 200C+200E+202C” + “202C+200E+202D+202C+202C+202C”
Hidden Message (HM) String	“”
Carrier Message (CM_{HM}=HM+CM)	“”+ How are you?
Output Message	How are you?

TABLE 7. Embedding algorithm analysis of AITSteg and existing techniques.

Algorithm	Type of Embedding	SM	CM_{HM} (HM+CM)	NELRS	NCRES (Approximate)	Summary of embedding algorithm
The AITSteg	Bit-level	Ali	How are you?	1	3	It embeds a hidden string of *SM* in front of *CM*, which does not depend on *CM*.
UniSpaCh [7]	Bit-level	Ali	How are you?	4	20	This technique embeds the secret bits by adding a special space beside of normal space into the *CM*, which each space refers to a 2-bit of SM binary (e.g., “00,10,01,11”).
TWSM [11]	Bit-level	Ali	How are you?	3	15	It utilizes the homoglyph letters and special spaces in order to embed the secret bits such that, some letters are replaced by similar letters with different codes for embedding 1-bit, and one special space is inserted between words for hiding 3-bit of secret bits.
AH4S [9]	Character-level	A	aard aard aard aard aard aard aard aard aard aard aard aard 2	2	33	This method employs the structure of omega network to embed the *SM* through a specific *CM*, which the *CM* is made according to the letters of *SM*. Moreover, it makes a *CM* contained 33 characters for each latter of SM.
Emoticons Based [12]	Character-level	Ali	How☺are☹you?☺	1	3	It embeds an emoticon for hiding each character of SM between words in the CM.

B. COMPARISON RESULTS

In what follows, we compare the AITSteg with the existing UniSpaCh [7], AH4S [9], TWSM [10], and emoticon-based [12] techniques with respect to the evaluation criteria. These techniques are chosen for comparison because they are the only techniques that apply Unicode characters to hide confidential information in digital text.

For fair comparison, first, we implemented the existing techniques on the highlighted examples shown in Table 7 and Table 3. We then rated the evaluation results for each technique with respect to the EC, invisibility, and robustness: for example, low and high scale for the EC; imperceptible and visible for invisibility; and low, modest, and high for robustness. In addition, we have outlined the security highlights and limitations of the evaluated techniques according to three evaluation criteria. Let us assume that we want to hide an *SM* = “Ali” using a *CM* = “How are you?”; and if we utilize the existing techniques to hide the *SM* in the *CM*, the evaluated results conducted on the CM_{HM} of each technique are as shown in Table 7. We have to observe that the comparison is carried out by considering a minimum word length of three characters. We highlighted special spaces to show the text trace in the CM_{HM}, but in practice, they have a transparent text trace (no color).

Table 7 shows the embedding features of the AITSteg and existing techniques, and Table 8 summarizes the *EC* and *DR* results produced by the AITSteg that are obtained from the highlighted messages (e.g., *SM* and *CM*). For fair

TABLE 8. The EC and DR results offered by the AITSteg and existing techniques on highlight examples.

Msg. Tests	SM	CM	Length of SM	Length of CM	Number of Embeddable Characters (approximate)				DR (%)			
					AITSteg	UniSpaCh	TWSM	Emoticons	AITSteg	UniSpaCh	TWSM	Emoticons
Msg.1	Hi	Where are you thinking about?	2	29	2	1	2	2	96	86	82	93
Msg.2	Hello	Do you want to change your life?	5	32	5	1.5	2.8	5	96	81	68	84
Msg.3	Golden	I choose a lazy person to do a hard job.	6	40	6	2.2	4	6	97	77	65	85
Msg.4	Bill Gates	I choose a lazy person to do a hard job. Because a lazy person will find an easy way to do it.	10	94	10	5.25	9.75	10	98	78	63	89
Msg.5	Bill Gates Golden Words	Don't Compare yourself with anyone in this world. If you do so, you are insulting yourself.	23	90	23	3.5	7	14	98	84	68	84
Msg.6	William Henry Gates III Bill Gates	I can understand wanting to have millions of dollars, there's a certain freedom, meaningful freedom, that comes with that. But once you get much beyond, that I have to tell you, it's the same hamburger.	34	202	34	8.5	16.3	34	99	83	68	83

comparison, we normalized the EC results of each method based on the number of embeddable characters per message by considering an 8-bit binary for each character of *SM*. Moreover, because the AH4S [9] generates an unknown *CM* for embedding the *SM*, we omitted it during the evaluation process.

To rate the criteria, we evaluated the EC of each technique based on the *EL* and the *BPL* (e.g., the number of embeddable bits or character per locations) using equation (1). As illustrated in the Fig. 6, the AITSteg exhibits a higher EC compared to those of the others on the same cover messages. For invisibility, we analyzed each technique by considering the embedding trace through the *CM*. Moreover, we obtained the *DR* of each technique using equation (2).

From Table 7 and 8, it can be observed that the AITSteg requires the lowest number of cover characters to embed a secret character for all of the *HM*.

Moreover, when compared with other bit-level embedding techniques, the AITSteg stands best by inserting a secret character in only one location, while the other techniques require a minimum of four locations. In addition, when we compare the AITSteg with character-level techniques, it is obvious that the proposed technique achieves a higher EC, high invisibility, and high robustness. This means that the AITSteg provides greater efficiency by considering optimum trade-offs between criteria. To evaluate the proposed technique, we compare some of the benefits and limitations of the AITSteg with those of the existing techniques, as shown in Table 9.

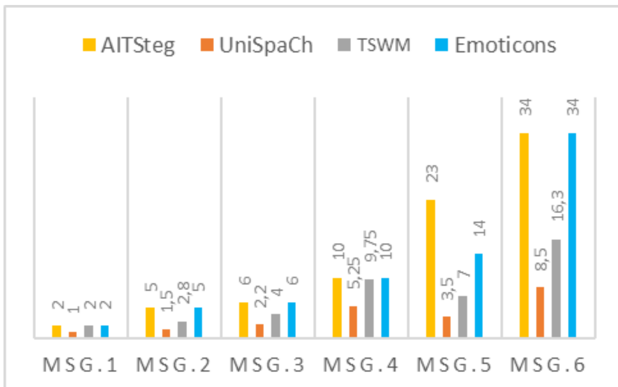

FIGURE 6. EC results of AITSteg vs existing techniques.

C. ANALYSIS AND DISCUSSIONS

This subsection discusses the AITSteg in various aspects such as mutual verification, key management, and prevention against attacks. Is the *SK* Safely stored? Since the attacker does not know the detailed structure of the encoding functions such as the Gödel function, hash function and symmetric key algorithm, he/she can discover neither the correct pairs of numbers nor the original *SM*. Moreover, because the *SK* is hidden by the CM_{HM} on the database server, it is almost impossible to extract the *SK* for the attackers. Is there any alternative to decoding the *HM*? Since an attacker only knows the CM_{HM} (using conventional approaches, although encoding functions and the *SK* are still unknown), the security of the

TABLE 9. Comparison of the AITSteg with existing techniques.

Algorithm	EC	Invisibility	Robustness	Highlights & Limitations
The AITSteg	High	Imperceptible	High DR>97%	➢ High EC (e.g., embedding total ZWCs of SM in front of CM) ➢ High invisibility ➢ High robustness against the MBR attacks ➢ High Security level against decoding attacks
UniSpaCh [7]	Low	Imperceptible	Modest DR>81%	➢ Low EC (e.g., 2-bit per inter-word spaces) ➢ High invisibility for readers ➢ Modest robustness against the MBR attacks ➢ Is not applicable for hiding an SM through a short CM
TWSM [11]	Low	Imperceptible	Modest DR>68%	➢ Low EC (e.g., 3- bit per inter-word spaces and 1-bit per homoglyph letters) ➢ High invisibility ➢ High robustness against the MBR attacks ➢ Is not applicable for hiding an SM through a short CM
AH4S [9]	Low	Visible	High DR>93%	➢ Low EC (e.g., 60 characters for each letter of the SM) ➢ Visible embedding trace in CM ➢ High robustness against the MBR attacks ➢ Raising suspicions for readers ➢ Is not applicable for hiding an SM through a short CM
Emoticons Based [12]	High	Visible	Modest DR>86%	➢ High EC ➢ Visible embedding trace in the CM ➢ Modest robustness against MBR attacks ➢ Raising suspicions for readers

proposed technique cannot be broken. Therefore, the AITSteg is perfectly secure.

1) MUTUAL AUTHENTICATION BETWEEN MS AND MR

In the scenario of the AITSteg technique, the *MR* verifies the *SM* integrity by using the *SK* and checks the identity of the *MS* from the CM_{HM}. When the *MR* receives a CM_{HM}, it extracts the SK_{MS} (e.g., the *SK* for *MS*) from the ZWCs and calculates the formulated SK_{MR} from the receiving time and compares it with the SK_{MS}. If it matches, then the authentication of the *SM* is performed by the *MR*, and moreover, it extracts the original *SM*; otherwise, the extraction algorithm will display the following message: ''you cannot access the hidden information!'' This certifies the mutual authentication between the *MS* and the *MR* through SMAPPs (or network).

2) KEY MANAGEMENT

The AITSteg technique provides efficient control of the key management issue on both sides of the algorithm (*MS* and *MR*), where the SK_{MS} is securely communicated by the *SMAPPs* to the *MR*. Therefore, this technique successfully hides the confidential information before its transmission over the *SMAPPs*. Because low-processing smartphones are in use, we employed a symmetric key algorithm to optimize the efficiency of the proposed technique, as it is 1000 times faster than the asymmetric algorithms [1].

3) PREVENTION AGAINST ATTACKS

In this subsection, we show that the AITSteg technique is able to prevent the transmitted *HM* from various attacks over the SMAPPs or even the network. Let us assume that the encoding functions are not available and are securely protected. Obtaining any secret key *SK* is not feasible because it has been transmitted in an imperceptible way through the CM_{HM} and is always compared with the formulated receiving time when required.

- *Message Disclosure and OTA Attacks:* The AITSteg provides end-to-end security for confidential information from the *MS* to the *MR* consisting of an *OTA* interface with a combination of encoding and text hiding algorithms. The proposed technique does not depend on the default encryption algorithm (e.g., A5/1, A5/2), which exists between the mobile station and BTS in cellular networks. Moreover, it is utilized to provide end-to-end confidentiality to the transmitted secret message in the SMAPPs. Therefore, the encoding system and hidden trace of the *HM* protect the transmitted CM_{HM} from message disclosure attack.
- *Man-in-the-middle Attack:* the AITSteg employs a dynamic symmetric key algorithm to create a hash function for encoding/decoding the hidden information transmission between the *MS* and the *MR;* i.e., it generates various hashed *SM* binaries or ZWCs for the same secret information at different times. The *HM* is securely encoded/decoded with an *SK* for every subsequent verification and since the attacker does not have sufficient information to reproduce the *SK*, it prevents the transmission from the MITM attack over the network.
- *Manipulation by Readers (MBR):* If the CM_{HM} is transmitted through the SMAPPs, the AITSteg still provides end-to-end security to the *MR/MS*; where it prevents the *HM* from being discovered by the SPOs as well as attackers who are monitoring conversations on the *SMAPPs*.

V. FORMAL PROOF OF PROPOSED TECHNIQUE

To clear the security analysis statement, we utilize the BAN Logic symbols to formally prove the authentication process of the AITSteg. (1) $P| \equiv X$:P believes X, (2) $P \triangleleft X$: P sees X. The MS has sent a CM_{HM} (HM + CM) including X to P, who can discover/read/repeat X (possibly after doing decoding), (3) $P|\sim X$:P once said X. P at some time sent a CM_{HM} containing the statement X, (4)$P|\Rightarrow X$:P has jurisdiction over X. P is an authority on X and should be trusted on this matter, (5) $\#(X)$: the formula X is fresh, which is, X has not been sent in a CM_{HM} at any time before the current run of the technique, (6) $P \overset{K}{\leftrightarrow} Q$: P and Q may use the shared key K to communicate, (7) $\overset{K}{\Leftrightarrow} Q$: The formula X is a secret known only to P and Q, (8) $\{X\}_K$: The formula X is encrypted under the key K [23].

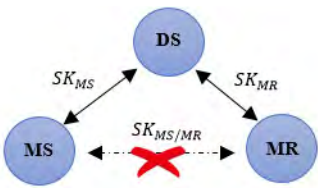

FIGURE 7. **AITSteg technique architecture.**

As illustrated in Fig. 7, there are three principals: *MS* and *MR*, two principals desiring mutual communication, and the *DS*, a trusted server. Let us assume that the *MS* and the *MR* already have a secure symmetric communication with the *DS* using keys SK_{MS} and SK_{MR}, respectively.

In practice, all the messages of SMAPPs are stored in the *DS*. Since the AITSteg uses SMAPPs or massagers as communication channels, there is no direct communication between *MS* and *MR*.

A. THE FORMAL MESSAGES IN THE AITSteg TECHNIQUE

(a): $MS \rightarrow DS : MS, \{T_{MS}, SK_{MS}\}_{SK_{MS}}, MR.$
(b): $DS \rightarrow MR : \{T_{MS}, SK_{MS}\}_{SK_{MS}}, MR.$
(c): $MR \rightarrow DS : MR, \{T_{MR}, SK_{MR}\}_{SK_{MR}}, MS.$
(d): $DS \rightarrow MS : \{T_{MR}, SK_{MR}\}_{SK_{MR}}, MS.$

B. SECURITY ASSUMPTIONS

Let us assume that *SK* is a symmetric key that is shared between *MS* and *MR*. (a) *MS* has a SK_{MS} key and $MS| \equiv MS \overset{SK}{\leftrightarrow} MR$. (b) *MR* has a SK_{MR} key and $MR| \equiv MR \overset{SK}{\leftrightarrow} MS$. It is assumed that the hidden conversations between the *MS* and *MR* is performed with a hidden *SK*, which is shared between the pair of them, i.e., $MS| \equiv (DS \Rightarrow MS \overset{SK}{\leftrightarrow} MR)$; $MS| \equiv (DS \Rightarrow MR \overset{SK}{\leftrightarrow} MS)$; $MS| \equiv DS \Rightarrow MS| \equiv DS \Rightarrow \#(MS \overset{SK}{\leftrightarrow} MR)$;

C. SECURITY ANALYSIS

(a): $MS \rightarrow MR : MS| \equiv \#(T_{MS}) \wedge DS| \equiv \#(T_{MS})$; $MR \triangleleft \{T_{MS}, SK_{MS}\}_{SK_{MS}}$; $MS \overset{SK_{MS}}{\leftrightarrow} MR$;
(b): After receiving a CM_{HM} from the *DS*, the *MR* extracts the SK_{MS}, and calculates SK_{MR}; if it matches with the extracted SK_{MS}, the *MR* decodes the original *SM* from CM_{HM}.
(c): $MR \rightarrow MS : MR| \equiv \#(T_{MR}) \wedge DS| \equiv \#(T_{MR})$; $MS \triangleleft \{T_{MR}, SK_{MR}\}_{SK_{MR}}$; $MR \overset{SK_{MR}}{\leftrightarrow} MS$

D. MESSAGE MEANING RULE

(a): $\dfrac{MS|\equiv(MS \overset{SK_{MS}}{\leftrightarrow} DS)\wedge(DS \overset{SK_{MS}}{\leftrightarrow} MR), MR\triangleleft\{T_{MS}, SK_{MS}\}_{SK_{MS}}}{MS|\equiv MR|\sim\{T_{MS}, SK_{MS}\}_{SK_{MS}}}$

(b): $\dfrac{MR|\equiv(MR \overset{SK_{MR}}{\leftrightarrow} DS)\wedge(DS \overset{SK_{MR}}{\leftrightarrow} MS), MS\triangleleft\{T_{MR}, SK_{MR}\}_{SK_{MR}}}{MR|\equiv MS|\sim\{T_{MR}, SK_{MR}\}_{SK_{MR}}}$

E. NONCE/TIMESTAMP VERIFICATION RULE

(a): $\dfrac{MS|\equiv\#(T_{MS})\wedge MR|\equiv\#(T_{MR}), MS|\equiv DS|\sim\{T_{MS}, SK_{MS}\}_{SK_{MS}}}{MS|\equiv DS|\equiv\{T_{MS}, SK_{MS}\}_{SK_{MS}}}$

(b): $\dfrac{MR|\equiv\#(T_{MR})\wedge MS|\equiv\#(T_{MS}), MR|\equiv DS|\sim\{T_{MR}, SK_{MR}\}_{SK_{MR}}}{MR|\equiv DS|\equiv\{T_{MR}, SK_{MR}\}_{SK_{MR}}}$

F. JURISDICTION RULE

(a): $\dfrac{MS|\equiv DS\Rightarrow(MS \overset{SK_{MS}}{\leftrightarrow} MR), MS|\equiv DS|\equiv(MS \overset{SK_{MS}}{\leftrightarrow} MR)}{MS|\equiv(MS \overset{SK_{MS}}{\leftrightarrow} MR)}$

(b): $\dfrac{MR|\equiv DS\Rightarrow(MR \overset{SK_{MR}}{\leftrightarrow} MS), MR|\equiv DS|\equiv(MR \overset{SK_{MR}}{\leftrightarrow} MS)}{MR|\equiv(MR \overset{SK_{MR}}{\leftrightarrow} MS)}$

G. GOALS

(1) Mutual Authentication between the MS and the *MR* : $MS| \equiv DS \wedge MR| \equiv DS \rightarrow MR| \equiv DS \wedge MS| \equiv DS$, therefore, the mutual authentication is retained. (2) Efficient key management between the *MS* and *MR*: SK_{MS}/SK_{MR} is used to provide an agreement.

(3) Confidentiality between the *MS* and MR via the DS:

$$\frac{MS| \equiv (MS \overset{SK_{MS}}{\leftrightarrow} MR),\ MR \triangleleft \{HM\}_{SK_{MS}}}{MS| \equiv MR \sim |HM} \wedge \frac{MR| \equiv (MR \overset{SK_{MR}}{\leftrightarrow} MS),\ MS \triangleleft \{HM\}_{SK_{MR}}}{MR| \equiv MS \sim |HM}$$

(4) Resistance Message Disclosure and OTA attacks: Since the existence of the SM and the SK_{MS} or SK_{MR} is hidden, and dynamic for each *SM*, the AITSteg protects the *SM* from these attacks.

(5) Resistance to *MITM* Attack: Since the *MITM* attacker knows neither the SK_{MS}/SK_{MR} nor the hash function, it prevents the confidential information from being discovered.

(6) Resistance to *MBR* Attack: As we have already proved in section (IV), if an MBR manipulates the CM_{HM}, the *HM* will remain at the front of the CM_{HM} with high probability. Thus, it prevents the *HM* from being visually detected.

VI. CONCLUSION

This study proposes a novel text steganography technique called AITSteg for hidden transmission of text message via SMS or social media between smartphone users. The proposed technique is able to hide large capacity secret information inside a short cover message so that the embedding trace is totally invisible to viewers. In addition, a combination of mathematical encoding and symmetric key algorithms is introduced that generates various secret bits even for the same confidential information in different times and keeps it perfectly secure against attacks. The analysis of the AITSteg technique confirms that it is able to prevent various attacks. Moreover, it provides higher EC, invisibility, and robustness compared to the existing techniques. In the best case scenario, the AITSteg provides optimum trade-offs between evaluation criteria, offering a certain level of safety for the hidden conversation via conventional social media or messengers.

Future work will investigate the use of ZWCs as a security tool in version control systems (VCS) to protect the open source programs against reverse engineering attacks.

ACKNOWLEDGMENT
The authors would like to thank Professor John Boyland from University of Wisconsin-Milwaukee, the USA, who provided insight and expertise that greatly assisted this research.

REFERENCES
[1] N. Saxena and N. S. Chaudhari, ''EasySMS: A protocol for end-to-end secure transmission of SMS,'' *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 7, pp. 1157–1168, Jul. 2014.
[2] A. Das and H. U. Khan, ''Security behaviors of smartphone users,'' *Inf. Comput. Secur.*, vol. 24, no. 1, pp. 116–134, 2016.
[3] M. T. Ahvanooey, Q. Li, M. Rabbani, and A. R. Rajput, ''A survey on smartphones security: Software vulnerabilities, malware, and attacks,'' *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 10, pp. 30–45, 2017.
[4] K. Park, G. I. Ma, J. H. Yi, Y. Cho, S. Cho, and S. Park, ''Smartphone remote lock and wipe system with integrity checking of SMS notification,'' in *Proc. IEEE ICCE*, Jan. 2011, pp. 263–264.
[5] M. T. Ahvanooey, Q. Li, H. J. Shim, and Y. Huang, ''A comparative analysis of information hiding techniques for copyright protection of text documents,'' *Secur. Commun. Netw.*, vol. 2018, Apr. 2018, Art. no. 5325040, doi: 10.1155/2018/5325040.
[6] K. F. Rafat and M. Sher, ''Secure digital steganography for ASCII text documents,'' *Arabian J. Sci. Eng.*, vol. 38, no. 8, pp. 2079–2094, 2013.
[7] L. Y. Por, K. Wong, and K. O. Chee, ''UniSpaCh: A text-based data hiding method using Unicode space characters,'' *J. Syst. Softw.*, vol. 85, no. 5, pp. 1075–1082, 2012.
[8] M. T. Ahvanooey, H. D. Mazraeh, and S. H. Tabasi, ''An innovative technique for Web text watermarking (AITW),'' *Inf. Secur. J., Global Perspective*, vol. 25, no. 6, pp. 191–196, 2016.
[9] A. M. Hamdan and A. Hamarsheh, ''AH4S: An algorithm of text in text steganography using the structure of omega network,'' *Secur. Commun. Netw.*, vol. 9, no. 18, pp. 6004–6016, Dec. 2016, doi: 10.1002/sec.1752.
[10] S. G. Rizzo, F. Bertini, D. Montesi, and C. Stomeo, ''Text watermarking in social media,'' in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining (ASONAM)*, Jul. 2017, pp. 208–211.
[11] T. P. Nagarhalli, ''A new approach to SMS text steganography using emoticons,'' *Int. J. Comput. Appl.*, pp. 1–3, 2014.
[12] S. A. L. Patiburn, V. Iranmanesh, and P. L. Teh, ''Text steganography using daily emotions monitoring,'' *Int. J. Educ. Manage. Eng.*, vol. 7, no. 3, pp. 1–14, 2017.
[13] M. Khodaei, B. S. Bigham, and K. Faez, ''Adaptive data hiding, using pixel-value-differencing and LSB substitution,'' *Cybern. Syst.*, vol. 47, no. 8, pp. 617–628, 2016, doi: 10.1080/01969722.2016.1214459.
[14] Y.-T. Lin, C.-M. Wang, W.-S. Chen, F.-P. Lin, and W. Lin, ''A novel data hiding algorithm for high dynamic range images,'' *IEEE Trans. Multimedia*, vol. 19, no. 1, pp. 196–210, Jan. 2017.
[15] D. Bucerzan, C. Ratiu, and M. J. Manolescu, ''SmartSteg: A new Android based steganography application,'' *Int. J. Comput. Commun. Control*, vol. 8, no. 5, pp. 681–688, 2013, doi: 10.15837/ijccc.2013.5.642.
[16] M. Habibi, R. Karimi, and M. Nosrati, ''Using SFLA and LSB for text message steganography in 24-bit RGB color images,'' *Int. J. Eng. Sci.*, vol. 2, no. 3, pp. 68–75, 2013.
[17] S. Gupta and R. Jain, ''An innovative method of text steganography,'' in *Proc. 3rd Int. Conf. Image Inf. Process.*, Dec. 2015, pp. 60–64.
[18] M. Agarwal, ''Text steganographic approaches: A comparison,'' *Int. J. Netw. Secur. Appl.*, vol. 5, no. 1, pp. 91–106, 2013.
[19] Z. Jalil and A. M. Mirza, ''A robust zero-watermarking algorithm for copyright protection of text documents,'' *J. Chin. Inst. Eng.*, vol. 36, no. 2, pp. 180–189, 2013.
[20] N. S. Kamaruddin, A. Kamsin, L. Por, and H. Rahman, ''A review of text watermarking: Theory, methods, and applications,'' *IEEE Access*, vol. 6, pp. 8011–8028, 2018, doi: 10.1109/ACCESS.2018.2796585.
[21] M. Conti, N. Dragoni, and V. Lesyk, ''A survey of man in the middle attacks,'' *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2027–2051, 3rd Quart., 2016.
[22] M. Davis and E. J. Weyuker, *Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science*. New York, NY, USA: Academic, 1993, pp. 47–60.
[23] M. Burrows, M. Abadi, and R. Needham, ''A logic of authentication,'' *ACM Trans. Comput. Syst.*, vol. 8, no. 1, pp. 18–36, 1990.

MILAD TALEBY AHVANOOEY received the B.Sc. degree in software engineering from UAST, Semnan, Iran, in 2012, and the M.Sc. degree in computer engineering from IAU Science and Research, Tehran, Iran, in 2014. He is currently pursuing the Ph.D. degree in computer science and technology with the Nanjing University of Science and Technology, Nanjing, China. From 2014 to 2016, he was a Lecturer with the School of Mathematics and Computer Sciences, Damghan University, Iran. His research interests include information hiding (text watermarking and text steganography), mobile security, IoT security, and genetic programming. He is also an External Reviewer of various international journals, including the *Computers in Human Behavior* and the *KSII Transactions on Internet and Information Systems*.

QIANMU LI received the B.Sc. and Ph.D. degrees from the Nanjing University of Science and Technology, China, in 2001 and 2005, respectively. He is currently a Full Professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, China. His research interests include information security, computing system management, and data mining. He received the China Network and Information Security Outstanding Talent Award in 2016 and multiple education ministry science and technology awards in 2012, 2014, and 2017.

JUN HOU received the M.Sc. degree in computer engineering from the Nanjing University of Science and Technology, Nanjing, China, in 2007, where she is currently pursuing the Ph.D. degree. Since 2005, she has been a Lecturer with the College of Zijin, Nanjing University of Science and Technology. Her research interests include social evaluation and data mining. She received the Jiangsu Province Science and Technology Award in 2011 and 2013, respectively.

HASSAN DANA MAZRAEH received the B.Sc. degree in computer science from the University of Tabriz, Iran, in 2010, and the M.Sc. degree in computer science from Damghan University, Semnan, Iran, in 2012. Since 2014, he has been a Faculty Member with the School of Mathematics and Computer Sciences, Damghan University, where he has been teaching computer science courses. His research interests include scientific computing, steganography, and evolutionary algorithms.

JING ZHANG received the Ph.D. degree in computer science from the Hefei University of Technology, China, in 2015. He is currently an Associate Professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, China. His research interests include data mining, machine learning, distributed systems, and information security. He has published dozens of articles in prestigious journals, such as TKDE, TCYB, TNNLS, JMLR, and top-tier conferences, such as AAAI, SIGIR, and CIKM. He serves a PC member for several international conferences and external reviewer for many journals.