IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received May 30, 2019, accepted June 16, 2019, date of publication June 19, 2019, date of current version July 3, 2019.

*Digital Object Identifier 10.1109/ACCESS.2019.2923728*

# Analysis and Implementation of Message Authentication Code (MAC) Algorithms for GOOSE Message Security

**S. M. SUHAIL HUSSAIN[1], (Member, IEEE), SHAIK MULLAPATHI FAROOQ[2], (Member, IEEE), AND TAHA SELIM USTUN[1], (Member, IEEE)**

[1]Fukushima Renewable Energy Institute, AIST (FREA), Koriyama 963-0215, Japan
[2]Department of Computer Science and Engineering, YSR Engineering College, Yogi Vemana University, Andhra Pradesh 516360, India

Corresponding author: Shaik Mullapathi Farooq (smfarooq@ieee.org)

This work was supported in part by the Research and Innovation Fund 2019 and in part by the KEIDANREN (Japan Business Federation) Promotion of Environmental Protection Foundation's Research Grant-2018.

**ABSTRACT** There is growing awareness towards cybersecurity threats in power systems. IEC 61850 standard facilitates communication between different Intelligent Electronic devices (IEDs) and eases interoperable operation with set data and message structures. An unwanted consequence of this standardized communication over ethernet is increased viability to cyber threats. The IEC 62351-6 standard stipulates the use of digital signatures for ensuring integrity in IEC 61850 message exchanges. However, the digital signatures result in higher computational times which makes it very difficult to use for Generic Object-Oriented Substation Events (GOOSE) messages. This short communication article proposes implementation of the Message Authentication Code (MAC) algorithms, such as Hash-based Message Authentication Code (HMAC) and Advanced Encryption Standard-Galois Message Authentication Code (AES-GMAC), for GOOSE message integrity. Lab tests are run to observe their timing performances and feasibility for GOOSE.

**INDEX TERMS** IEC 62351, IEC 61850, generic object-oriented substation Events (GOOSE), cybersecurity, hash-based message authentication code (HMAC), advanced encryption standard-galois message authentication code (AES-GMAC).

## I. INTRODUCTION

IEC 61850 is the de-facto communication standard for Substation Automation Systems (SAS) [1]. IEC 61850's popularity can be attributed to two main factors: ease of connection via ethernet instead of traditional hard-wired systems and standardized message structures which ensures interoperability. An unwanted consequence of these is the increased vulnerability to cyber-attacks. It is easier to access ethernet-based networks and standardized messages allow hackers to know exactly what instructions to give. IEC 62351-6 standard is published to complement IEC 61850 by adding security features [2].

IEC 62351-6 identifies that message authenticity is an important security requirement for IEC 61850 communication in SAS. For achieving this, IEC 62351-6 standard recommends the use of authentication value algorithm with digital signatures (DS). For DS, use of Rivest–Shamir–Adleman (RSA) algorithm as per Request for Comments (RFC) 2313 is stipulated [3]. However, it is found that RSA based DS require computational times on the order of few milli seconds which is not suitable for Generic Object-Oriented Substation Events (GOOSE) message-based applications [4], [5]. GOOSE messages are used to transfer time critical information such as start, stop, trip and close between intelligent electronic devices (IEDs) in a substation. The GOOSE messages have a strict delivery timing requirement of 3 ms. In [6], [7] authors investigated Elliptic Curve Digital Signature Algorithm (ECDSA) based DS which resulted comparatively lower computational times compared to RSA based DS. However, still the computational times for ECDSA DS is much higher than the requirements of GOOSE messages [6]-[8]. Alternatively, it is possible to use Message Authentication Code (MAC) algorithms for GOOSE security as IEC 61850-90-5 [9] already stipulates it for Routable GOOSE (R-GOOSE) and Routable SV (R-SV).

The associate editor coordinating the review of this manuscript and approving it for publication was Filbert Juwono.

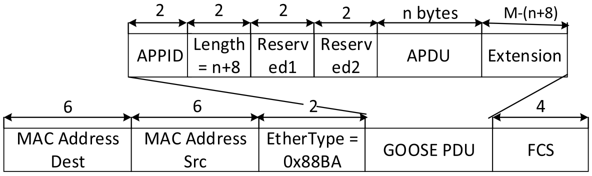


**FIGURE 1.** Extended GOOSE PDU format.

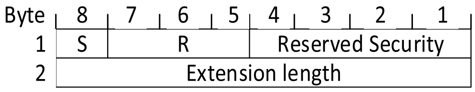


**FIGURE 2.** Structure of Reserved1 field.

Recently, in [10] authors employed hash-based message authentication code (HMAC) algorithm to secure GOOSE messages. However, the details for appending the HMAC value to GOOSE format is not specified. The specification of message format or structure is very crucial to achieve interoperability and standardization. Furthermore, in [10], actual implementation and practical results of computational time required for processing the MAC algorithms in time critical GOOSE messages was not discussed.

Therefore, in this article, GOOSE message structure is modified as per IEC 62351 to secure them with different MAC algorithms (such as HMAC and Advanced Encryption Standard - Galois Message Authentication Code (AES-GMAC)). A software library is developed to implement these secure GOOSE messages in the lab. Finally, lab tests have been run with different MAC algorithms to observe their timing performances. This data is utilized to compare MAC algorithms and analyze their feasibility in securing GOOSE messages in SAS.

## II. MAC ALGORITHMS FOR GOOSE MESSAGE SECURITY

The main objective of GOOSE message is to offer a fast and reliable mechanism to exchange of data among substation IEDs. GOOSE messages are directly mapped on to ethernet layer to avoid the network and transport layer headers reducing the overall size of message, this in turn reduces the propagation and processing delays of the GOOSE messages. As shown in Fig. 1, the GOOSE message consists of destination and source address fields, 6 bytes each, followed by the Ether-type field of 2 bytes which defines the type of data present in the payload field. The value of ether-type for GOOSE message is 88-B8. GOOSE PDU consists of APPID, Length, Reserved1, Reserved2, GOOSE APDU fields followed by padded data and Frame Check Sequence (FCS).

Implementation of MAC algorithms require addition of a MAC value to GOOSE messages. GOOSE message structure is very well-defined by IEC 61850 and the modification required for security extensions is defined in IEC 62351-6. Accordingly, MAC value is appended to the extended GOOSE frame format in "Extension" field as per ASN.1 definitions, shown in Fig. 1. As this increases the length of the GOOSE frame, the difference is reflected on the 2nd byte of 2-byte "Reserved1" field. As shown in Fig. 2, most significant bit in 1st byte of "Reserved1" is used to indicate simulated GOOSE message while the rest is reserved for future implementations. 2nd byte represents the increase in length due to addition of MAC value in "Extension" field. The range of values for the extension length are 0 to 255. If extension length field value is 0, it specifies that no security extension added to the GOOSE PDU. "Reserved2" field is used to specify 16-bit CRC value, which is calculated for first 8 bytes of the "Extension" field.

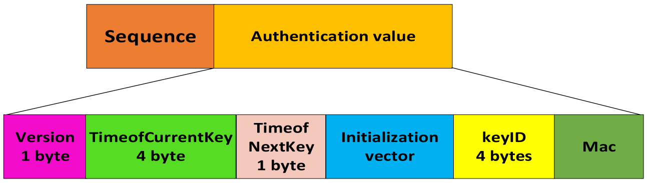


**FIGURE 3.** Structure of the "Extension" field.

**TABLE 1.** MAC variants used in analysis.

| S.No | MAC Algorithm | Hash Function | *MAC* value (Size in bytes) |
|---|---|---|---|
| 1 | HMAC-SHA256-80 | SHA-256 | 10 |
| 2 | HMAC-SHA256-128 | SHA-256 | 16 |
| 3 | HMAC-SHA256-256 | SHA-256 | 32 |
| 4 | AES-GMAC-64 | - | 8 |
| 5 | AES-GMAC-128 | - | 16 |

The structure of the extension field appended to the GOOSE protocol data unit (PDU) is shown in Fig. 3. "SEQUENCE" field is reserved for future security additions other than encryption and message authentication. "Authentication value" consists of "version", "TimeofCurrentKey", "TimeofNextKey", "InitializationVector (IV)", "KeyID" and "MAC" fields. "Version" is by default 1; while "TimeofCurrentKey" and "TimetoNextKey" contain time data pertaining to key used in MAC algorithms. "InitializationVector (IV)" is an optional field which is used when some initialization values are required, e.g. AES-GMAC algorithms requires an initialization value. Its size depends on the MAC algorithm. "KeyID" is the reference to key that is used. "MAC" field contains the MAC value that is generated.

Different MAC algorithms recommended in IEC 61850-90-5 for R-GOOSE and R-SV are listed in Table 1 and the same are proposed for securing GOOSE messages. Corresponding sizes of generated MAC values are also given. MAC algorithms that are recommended in IEC 61850-90-5 are Hash based Message Authentication Code – Secure Hash Algorithm (HMAC-SHA-256) with 80 and 128 truncations, AES-GMAC-64 and AES-GMAC-128.

## III. IMPLEMENTATION DETAILS AND ANALYSIS OF RESULTS

In this section, proposed MAC algorithms have been implemented and their performances are evaluated to confirm their applicability to GOOSE messages. For performance evaluation tests, the experimental setup of Fig. 4 is used.

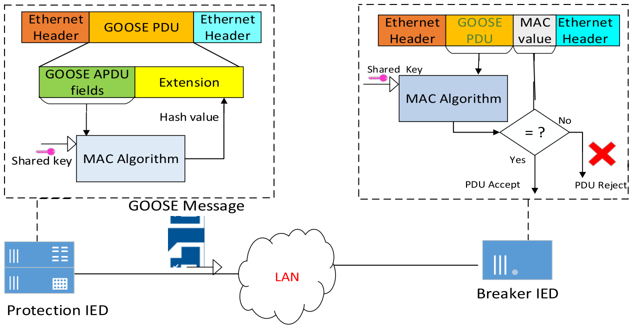
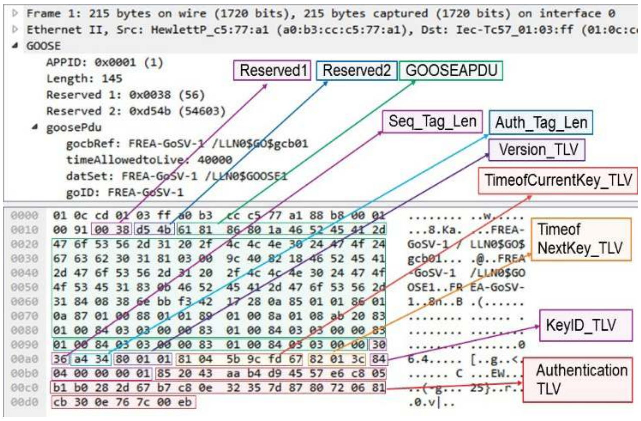


**FIGURE 4.** HMAC integration to GOOSE messages for secure communication.



```
▷ Frame 1: 215 bytes on wire (1720 bits), 215 bytes captured (1720 bits) on interface 0
▷ Ethernet II, Src: HewlettP_c5:77:a1 (a0:b3:cc:c5:77:a1), Dst: Iec-Tc57_01:03:ff (01:0c:c
◢ GOOSE
    APPID: 0x0001 (1)
    Length: 145
    Reserved 1: 0x0038 (56)
    Reserved 2: 0xd54b (54603)
  ◢ goosePdu
      gocbRef: FREA-GoSV-1 /LLN0$GO$gcb01
      timeAllowedtoLive: 40000
      datSet: FREA-GoSV-1 /LLN0$GOOSE1
      goID: FREA-GoSV-1

0000  01 0c cd 01 03 ff a0 b3  cc c5 77 a1 88 b8 00 01   ........ ..w.....
0010  00 91 00 38 d5 4b 61 81  86 80 1a 46 52 45 41 2d   ...8.Ka. ...FREA-
0020  47 6f 53 56 2d 31 20 2f  4c 4c 4e 30 24 47 4f 24   GoSV-1 / LLN0$GO$
0030  67 63 62 30 31 81 03 00  9c 40 82 18 46 52 45 41   gcb01... .@..FREA
0040  2d 47 6f 53 56 2d 31 20  2f 4c 4c 4e 30 24 47 4f   -GoSV-1  /LLN0$GO
0050  4f 53 45 31 83 0b 46 52  45 41 2d 47 6f 53 56 2d   OSE1..FR EA-GoSV-
0060  31 84 08 38 6e bb f3 42  17 28 0a 85 01 01 86 01   1..8n..B .(......
0070  0a 87 01 00 88 01 01 89  01 00 8a 01 08 ab 20 83   ........ ...... .
0080  01 00 84 03 03 00 00 83  01 00 84 03 03 00 00 83   ........ ........
0090  01 00 84 03 03 00 00 83  01 00 84 03 03 00 00 30   ........ .......0
00a0  36 a4 34 80 01 01 81 04  5b 9c fd 67 82 01 3c 84   6.4..... [..g..<.
00b0  04 00 00 00 01 85 20 43  aa b4 d9 45 57 e6 c8 05   ...... C ...EW...
00c0  b1 b0 28 2d 67 b7 c8 0e  32 35 7d 87 80 72 06 81   ..(-g... 25}..r..
00d0  cb 30 0e 76 7c 00 eb                               .0.v|..
```

**FIGURE 5.** Packet capture of secure GOOSE message with HMAC extension.

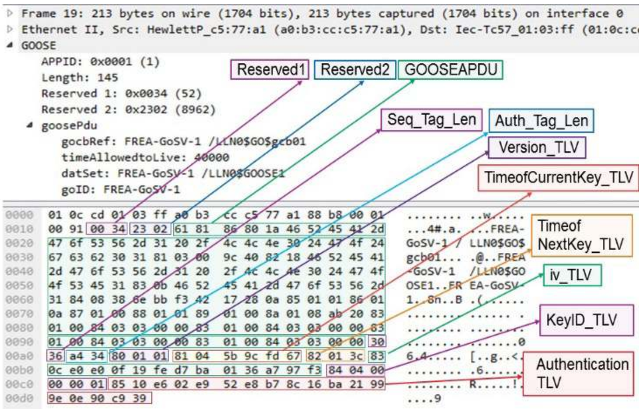
## A. SECURE GOOSE GENERATION

Authors have developed a GOOSE and SV generator library GoSV [11] to run tests on custom messages. By integrating MAC algorithms proposed above, a new library S-GoSV is developed. It secures GOOSE message communication by employing key verification and MAC algorithms as shown in Fig. 4. It is written in C language utilizing linux structures ifreq and sockaddr_ll libraries. S-GoSV framework generates a MAC value using a symmetric key that is shared beforehand. Then, it is stored in the "Extension" field of GOOSE PDU. The secure GOOSE message generation algorithm, Gen_MAC(), is given below:

**Algorithm 1** Algorithm Gen_MAC( )

```
goosePDU ← $GoSV()$
$InputData$ ← $goosePDU.APDU$
$k$ ← $PreSharedKey()$
$h$ ← $MAC_k(InputData)$
$goosePDU.Extension$ ← $h$
```

Protection IED generates a secure GOOSE message by appending "Extension" field to the original GOOSE PDU. Figure 5 shows the Wireshark capture of Secure GOOSE messages generated by S-GoSV framework for HMAC algorithm. It shows all the relative fields of GOOSE PDU along with extension fields such as "version", "TimeofCurrentKey", "TimeofNextKey", "keyID" and actual MAC value in ASN.1 format (i.e. tag, value and length - TLV). "Reserved1" field shows the size of the "Extension" field generated in the GOOSE PDU. "Reserved2" field shows the CRC value of first 8 bytes of the "Extension" field. Similarly, using S-GoSV library, secure GOOSE messages generated for AES-GMAC algorithms is shown in Fig. 6.



```
▷ Frame 19: 213 bytes on wire (1704 bits), 213 bytes captured (1704 bits) on interface 0
▷ Ethernet II, Src: HewlettP_c5:77:a1 (a0:b3:cc:c5:77:a1), Dst: Iec-Tc57_01:03:ff (01:0c:c
◢ GOOSE
    APPID: 0x0001 (1)
    Length: 145
    Reserved 1: 0x0034 (52)
    Reserved 2: 0x2302 (8962)
  ◢ goosePdu
      gocbRef: FREA-GoSV-1 /LLN0$GO$gcb01
      timeAllowedtoLive: 40000
      datSet: FREA-GoSV-1 /LLN0$GOOSE1
      goID: FREA-GoSV-1

0000  01 0c cd 01 03 ff a0 b3  cc c5 77 a1 88 b8 00 01   ........ ..w.....
0010  00 91 00 34 23 02 61 81  86 80 1a 46 52 45 41 2d   ...4#.a. ...FREA-
0020  47 6f 53 56 2d 31 20 2f  4c 4c 4e 30 24 47 4f 24   GoSV-1 / LLN0$GO$
0030  67 63 62 30 31 81 03 00  9c 40 82 18 46 52 45 41   gcb01... .@..FREA
0040  2d 47 6f 53 56 2d 31 20  2f 4c 4c 4e 30 24 47 4f   -GoSV-1  /LLN0$GO
0050  4f 53 45 31 83 0b 46 52  45 41 2d 47 6f 53 56 2d   OSE1..FR EA-GoSV-
0060  31 84 08 38 6e bb f3 42  17 28 0a 85 01 01 86 01   1..8n..B .(......
0070  0a 87 01 00 88 01 01 89  01 00 8a 01 08 ab 20 83   ........ ...... .
0080  01 00 84 03 03 00 00 83  01 00 84 03 03 00 00 83   ........ ........
0090  01 00 84 03 03 00 00 83  01 00 84 03 03 00 00 30   ........ .......0
00a0  36 a4 34 80 01 01 81 04  5b 9c fd 67 82 01 3c 83   6.4..... [..g..<.
00b0  0c e0 e0 0f 19 fe d7 ba  01 36 a7 97 f3 84 04 00   ........ .6......
00c0  00 00 01 85 10 e6 02 e9  52 e8 b7 8c 16 ba 21 99   ........ R.....!.
00d0  9e 0e 90 c9 39                                     ....9
```

**FIGURE 6.** Packet capture of secure GOOSE message with AES-GMAC extension.

**Algorithm 2** Algorithm verify_MAC(goosePDU )

```
$h$ ← $goosePDU.extension$
$ReceivedData$ ← $goosePDU.APDU$
$k$ ← $PreSharedKey()$
$h1$ ← $MAC_k(ReceivedData)$
if $h = h1$ then
    return "Accept GOOSE packet"
else
    return "Reject GOOSE packet"
end if
```

The breaker IED receives the secure GOOSE message and reads the APDU values into ReceivedData buffer. Algorithm verify_MAC is utilized to check the integrity of GOOSE message. Firstly, a new MAC value "h1" is computed for the received GOOSE PDU using the symmetric PreSharedKey k. "h1" is compared with the received MAC value "h". If they match, then the received GOOSE PDU is accepted as a legitimate packet; otherwise, it is discarded.

Table 2 shows the average computational time required and size of resultant secure GOOSE message for different MAC algorithms. The computational times are calculated for generating MAC at publisher, for generating MAC at subscriber and then for comparing both the MACs at subscriber. The programs were executed on a system with Intel® Celeron(R) processor with 4 GB RAM. This relatively old and slow system is intentionally selected. If this system can meet timing requirements, then current IEDs should not face any difficulties as they have much higher computing power (Intel® Core i7-3555LE with 8 GB RAM) [12]. It can be observed

**TABLE 2.** E2E delay of different MAC algorithms.

| S.No | Algorithm | | Total Size (bytes) | Average Computational time (ms) | | | Comm. Delays (ms) | | E2E delay (ms) | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Publisher MAC generation | Subscriber | | Avg | Max | Avg | Max |
| | | | | | MAC generation | Comparison | | | | |
| 1 | No security | | 159 | 0 | 0 | 0 | 0.0664 | 0.0685 | 0.0664 | 0.0685 |
| 2 | HMAC-SHA256 | 80 | 193 | 0.0127 | 0.0127 | 0.0014 | 0.0709 | 0.0730 | 0.0977 | 0.0998 |
| | | 128 | 199 | 0.0127 | 0.0127 | 0.0015 | 0.0722 | 0.0744 | 0.0991 | 0.1013 |
| | | 256 | 215 | 0.0127 | 0.0127 | 0.0016 | 0.0757 | 0.0780 | 0.1027 | 0.1050 |
| 3 | AES-GMAC-64 | | 205 | 0.0054 | 0.0054 | 0.0012 | 0.0730 | 0.0753 | 0.0850 | 0.0873 |
| 4 | AES-GMAC-128 | | 213 | 0.0055 | 0.0055 | 0.0014 | 0.0749 | 0.0771 | 0.0873 | 0.0895 |

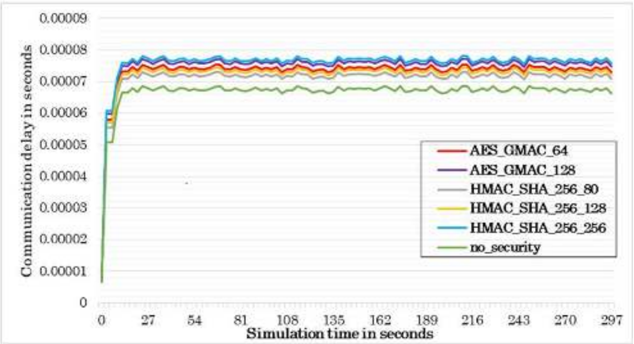


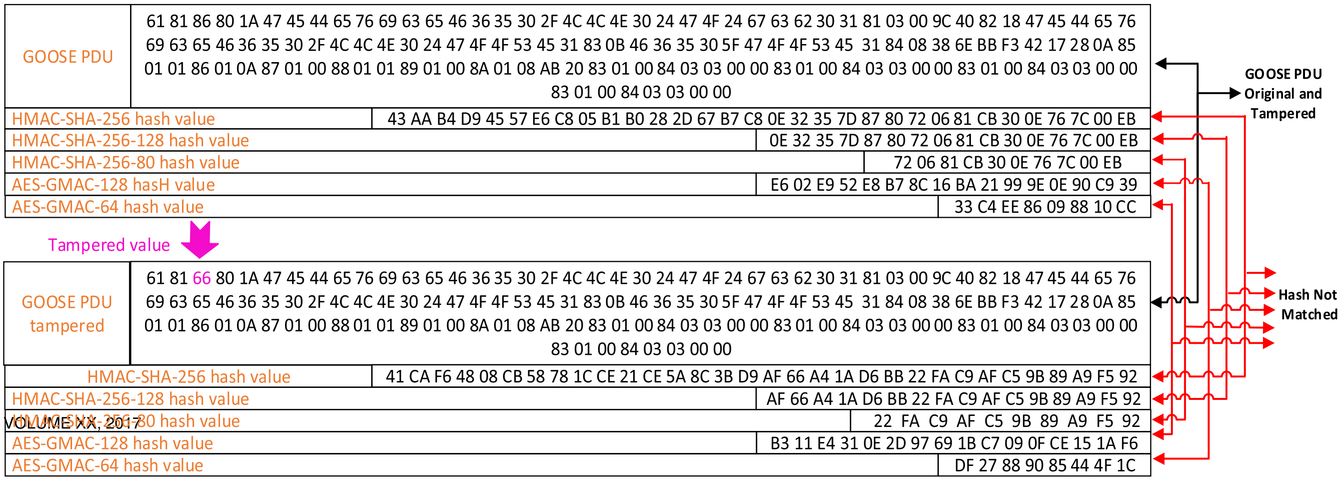
**FIGURE 7.** Communication delay for GOOSE messages with different MAC algorithms.

that AES-GMAC-64 algorithm has the best timing performance, while truncated HMAC-SHA256-80 has the lowest message size. From Table 2, it is quite evident that the performance in terms of computational times of MAC algorithms is comparatively much better than the RSA or ECDSA based digital signatures (which has computational times 1- 4 milli seconds) [3], [4].

Based on the secure GOOSE message size obtained from S-GoSV implementation, substation communication network has been simulated in Riverbed Modeler simulator tool [13] to find out communication delays. A bay of typical type D2-1 substation communication network consisting a Protection and Control (P&C) IED and Breaker IED was simulated in Riverbed Modeler. The communication delays for sending the secure GOOSE message with different MAC algorithms from P&C IED to Breaker IED are shown in Fig. 7. The total end to end (E2E) delays for exchanging secure GOOSE messages includes the computational times for running MAC algorithms and the communication delays. In Table 2, the last column gives the total E2E delay for exchanging the secure GOOSE messages. From the riverbed modeler simulations, the average and maximum communication delays for GOOSE messages are obtained. The maximum communication delays give the worst-case performance indicator. It is clearly evident from the results that the E2E delays for GOOSE messages secured with MAC algorithms even in worst-case are well within the stipulated requirements, i.e. 3 ms.

## B. TAMPER DTECTION

Any change in the GOOSE PDU during transmission reflects a discrepancy between the appended MAC value, h, and the



| | |
|---|---|
| GOOSE PDU | 61 81 86 80 1A 47 45 44 65 76 69 63 65 46 36 35 30 2F 4C 4C 4E 30 24 47 4F 24 67 63 62 30 31 81 03 00 9C 40 82 18 47 45 44 65 76<br>69 63 65 46 36 35 30 2F 4C 4C 4E 30 24 47 4F 4F 53 45 31 83 0B 46 36 35 30 5F 47 4F 4F 53 45 31 84 08 38 6E BB F3 42 17 28 0A 85<br>01 01 86 01 0A 87 01 00 88 01 01 89 01 00 8A 01 08 AB 20 83 01 00 84 03 03 00 00 83 01 00 84 03 03 00 00 83 01 00 84 03 03 00 00<br>83 01 00 84 03 03 00 00 |
| HMAC-SHA-256 hash value | 43 AA B4 D9 45 57 E6 C8 05 B1 B0 28 2D 67 B7 C8 0E 32 35 7D 87 80 72 06 81 CB 30 0E 76 7C 00 EB |
| HMAC-SHA-256-128 hash value | 0E 32 35 7D 87 80 72 06 81 CB 30 0E 76 7C 00 EB |
| HMAC-SHA-256-80 hash value | 72 06 81 CB 30 0E 76 7C 00 EB |
| AES-GMAC-128 hasH value | E6 02 E9 52 E8 B7 8C 16 BA 21 99 9E 0E 90 C9 39 |
| AES-GMAC-64 hash value | 33 C4 EE 86 09 88 10 CC |

| | |
|---|---|
| GOOSE PDU tampered | 61 81 66 80 1A 47 45 44 65 76 69 63 65 46 36 35 30 2F 4C 4C 4E 30 24 47 4F 24 67 63 62 30 31 81 03 00 9C 40 82 18 47 45 44 65 76<br>69 63 65 46 36 35 30 2F 4C 4C 4E 30 24 47 4F 4F 53 45 31 83 0B 46 36 35 30 5F 47 4F 4F 53 45 31 84 08 38 6E BB F3 42 17 28 0A 85<br>01 01 86 01 0A 87 01 00 88 01 01 89 01 00 8A 01 08 AB 20 83 01 00 84 03 03 00 00 83 01 00 84 03 03 00 00 83 01 00 84 03 03 00 00<br>83 01 00 84 03 03 00 00 |
| HMAC-SHA-256 hash value | 41 CA F6 48 08 CB 58 78 1C CE 21 CE 5A 8C 3B D9 AF 66 A4 1A D6 BB 22 FA C9 AF C5 9B 89 A9 F5 92 |
| HMAC-SHA-256-128 hash value | AF 66 A4 1A D6 BB 22 FA C9 AF C5 9B 89 A9 F5 92 |
| VOLUME XX, 2017 HMAC-SHA-256-80 hash value | 22 FA C9 AF C5 9B 89 A9 F5 92 |
| AES-GMAC-128 hash value | B3 11 E4 31 0E 2D 97 69 1B C7 09 0F CE 15 1A F6 |
| AES-GMAC-64 hash value | DF 27 88 90 85 44 4F 1C |

**FIGURE 8.** Hash values for tampered GOOSE PDUs for different HMAC algorithms.

one recomputed at the receiver, h1. Fig. 5 shows the results generated by the S-GoSV framework developed in this letter. As an example, GOOSE PDU value is tampered third byte is changed from 86 to 66, as shown in Fig. 8, and sent to the receiver. When hash values are recomputed, the tampering is successfully detected. Fig. 8 shows that all the MAC algorithms yield different MAC values when the GOOSE PDU is tampered. As these algorithms exhibit avalanche effect which makes them even more effective, i.e. a small change in input results in a large-scale variation in the MAC values, Also, due to this property, the truncated versions of HMAC would be effective as they potentially offer the same security benefits with lower size of MAC values. This considerably reduces the overhead of overall GOOSE packet size and, thus, in turn reduces the transmission delays of GOOSE message packets.

## IV. CONCLUSIONS

IEC 61850 is gaining more ground as the de facto communication standard in smart grids. However, it does not have necessary tools to mitigate or detect cyber-attacks. The existing security standard for providing security to IEC 61850 message exchanges i.e. IEC 62351-6 recommends use of digital signatures such as RSA to generate hash values. The DS algorithms have higher computational times and are very difficult to implement in GOOSE messages. In this article, a software library has been developed to generate secure GOOSE messages with HMAC and GMAC based MAC algorithms. Their packet sizes and timing performances are evaluated for their suitability to ensure cybersecurity for GOOSE messages. Lab tests show that all algorithms can effectively sustain secure communication. Furthermore, the exhibited avalanche effect means that truncated versions with smaller PDU sizes are the most optimum solution for security and timing requirements.

## REFERENCES

[1] *Communication Networks and Systems for Power Utility Automation*, 2nd ed., Standard IEC 61850, International Electrotechnical Commission, 2019.

[2] *Power Systems Management and Associated Information Exchange—Data and Communication Security—Part 6: Security for IEC 61850*, Standard IEC/TS 62351-6:2007(E), 2007.

[3] S. M. Farooq, S. M. S. Hussain, and T. S. Ustun, “Performance evaluation and analysis of IEC 62351-6 probabilistic signature scheme for securing GOOSE messages,” *IEEE Access*, vol. 7, pp. 32343–32351, 2019.

[4] F. Hohlbaum, M. Braendle, and A. Fernando, “Cyber Security Practical considerations for implementing IEC 62351,” in *Proc. PAC World Conf.*, Dublin, U.K., Jun. 2010, pp. 1–8.

[5] D. Ishchenko and R. Nuqui, “Secure communication of intelligent electronic devices in digital substations,” in *Proc. IEEE/PES Transmiss. Distrib. Conf. Expo. (T&D)*, Denver, CO, USA, Apr. 2018, pp. 1–5.

[6] T. T. Tesfay and J.-Y. Le Boudec, “Experimental comparison of multicast authentication for wide area monitoring systems,” *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 4394–4404, Sep. 2018.

[7] B. J. Matt, “The cost of protection measures in tactical networks,” in *Proc. 24th Army Sci. Conf.*, 2005, pp. 1–9.

[8] S. M. Farooq, S. M. S. Hussain, S. Kiran, and T. S. Ustun, “Certificate based security mechanisms in vehicular ad-hoc networks based on IEC 61850 and IEEE WAVE standards,” *Electronics*, vol. 8, no. 1, p. 96, 2019.

[9] *Communication Networks and Systems for Power Utility Automation—Part 90-5: Use of IEC 61850 to Transmit Synchrophasor Information According to IEEE C37.118*, Standard IEC TR 61850-90-5:2012, 2012.

[10] J. Zhang, J. Li, X. Chen, M. Ni, T. Wang, and J. Luo, “A security scheme for intelligent substation communications considering real-time performance,” *J. Mod. Power Syst. Clean Energy*, to be published. [Online]. Available: https://link.springer.com/article/10.1007/s40565-019-0498-5#citeas

[11] *GoSv*. Accessed: Mar. 25, 2019. [Online]. Available: https://github.com/61850security/GoSV

[12] *Data Sheet-SEL 3555 Real Time Automation Controller (RTAC)*. Accessed: Jun. 24, 2019. [Online]. Available: https://goo.gl/jjnfnV

[13] *Riverbed Modeler-(Formerly OPNET)*. Accessed: Jun. 24, 2019. [Online]. Available: http:// goo.gl/72SgAM

**S. M. SUHAIL HUSSAIN** received the Ph.D. degree in electrical engineering from Jamia Millia Islamia (a Central University), New Delhi, India, in 2018. He is currently a Postdoctoral Researcher with the Fukushima Renewable Energy Institute, AIST (FREA), Koriyama, Japan. His research interests include power system communication, cybersecurity in power systems, substation automation systems, IEC 61850 standards, electric vehicle integration, and smart grid.

He is a Guest Editor of the IEEE Transactions on Industrial Informatics. He was a recipient of IEEE Standards Education Grant approved by the IEEE Standards Education Committee for implementing project and submitting a student application paper, in 2014–2015.

**SHAIK MULLAPATHI FAROOQ** received the B.Tech. and M.Tech. degrees in computer science engineering from Jawaharlal Nehru Technological University, Hyderabad, India. He is pursuing the Ph.D. degree in computer science engineering from Yogi Vemana University, Kadapa, India. He was a Visiting Researcher with Fukushima Renewable Energy Institute, AIST (FREA), Japan, from September 2018 to December 2018. His research interests include cryptography, cyber physical systems, cybersecurity in vehicular networks, and power systems.

**TAHA SELIM USTUN** received the Ph.D. degree in electrical engineering from Victoria University, Melbourne, VIC, Australia. He is currently a Researcher with Fukushima Renewable Energy Institute, AIST (FREA) and leads Smart Grid Cybersecurity Lab. Prior to that he was an Assistant Professor in electrical engineering with the School of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA. His research interests include power systems protection, communication in power networks, distributed generation, microgrids, electric vehicle integration and cybersecurity in smartgrids.

He is an Associate Editor of IEEE Access and Guest Editor of the IEEE Transactions on Industrial Informatics. He is a member of the IEEE 2004, IEEE 2800 Working Groups, and IEC Renewable Energy Management Working Group 8. He has edited several books and special issues with international publishing houses. He is a Reviewer in reputable journals and has taken active roles in organizing international conferences and chairing sessions. He has been invited to run specialist courses in Africa, India, and China. He delivered talks for Qatar Foundation, World Energy Council, Waterloo Global Science Initiative, and European Union Energy Initiative (EUEI).