energies

MDPI

Article

S-GoSV: Framework for Generating Secure IEC 61850 GOOSE and Sample Value Messages

Shaik Mullapathi Farooq [1], S.M. Suhail Hussain [2,*] and Taha Selim Ustun [2]

[1] Department of Computer Science and Engineering, YSR Engineering College, Yogi Vemana University, Kadapa 516360, Andhra Pradesh, India
[2] Fukushima Renewable Energy Institute, AIST (FREA), Koriyama 963-0215, Japan
* Correspondence: suhail.hussain@aist.go.jp; Tel.: +81-804-373-5146

Received: 10 June 2019; Accepted: 29 June 2019; Published: 1 July 2019

Abstract: Standardized communication plays an important role in substation automation system (SAS). IEC 61850 is a de-facto standard in SAS. It facilitates smooth communication between different devices located in the substation by achieving interoperability. Generic Object-Oriented Substation Event (GOOSE) and Sample Value (SV) messages developed according to IEC 61850 enable efficient monitoring and operation control of SAS. IEC 61850 is very popular due to its flexible and robust modeling. As the number of critical infrastructures that employed IEC 61850 increases, it is important to study cybersecurity aspects as well. To this end, this paper develops a software framework, S-GoSV (Secure GOOSE and SV), that generates custom GOOSE and Sample Value messages. Furthermore, security features are added to protect them from different security attacks within a substation. IEC 62351-6 specifies digital signatures to achieve node authentication and messages integrity. Therefore, S-GoSV implements RSASSA-PKCS1-v1_5 digital signature algorithm based on RFC 2313. Performance studies show that digital signature algorithms based on RSA signing and verification take long times and do not conform to timing requirements stipulated by IEC 61850 for power system communication. To address this, Message Authentication Code (MAC) based digital signature algorithm, Keyed Hash-Message Authentication Code- Secure Hash Algorithm (HMAC-SHA256), is additionally implemented in S-GoSV framework for securing GOOSE messages.

Keywords: security in Substation communication system; Generic Object-Oriented Substation Event (GOOSE); Sample Values; IEC 62351-6 standard

1. Introduction

Legacy power systems are transforming into smart grids which are designed to operate in a more reliable and resilient fashion. Existing Supervisory Control and Data Acquisition (SCADA) systems do not have the capability to implement these features [1]. Substations play a key role in smart grid revolution. IEC 61850 is a de-facto standard for substation automation [2]. It offers different protocol services such Generic Object-Oriented Substation Event (GOOSE), Sample Value (SV) and Manufacturing Message Service (MMS) [2]. It enables interoperability among substation devices of different vendors with standard modeling. It also offers many services that include seamless communication, defining data sets, defining logical nodes for substation communication equipment [3].

IEC 61850 standard was initially developed for substation automation. Due to its widespread popularity and success, it was extended to other components in power grids such as fault current limiters [4], Distributed Energy Resources (DERs) [5], Electric Vehicles (EVs) [6] and smart meters [7]. In these new implementations, IEC 61850 GOOSE and SV are utilized to exchange different data sets between DERs, EVs and other components. Some researchers have developed open access tools for generating GOOSE and SV messages, which are quite helpful in conducting different studies [8–10].

However, the reported tools only use datasets based on IEC 61850-9-2 LE SV format (only include three-phase current and voltages values) or GOOSE formats. Addressing this knowledge gap; in this paper, a C based framework 'GoSV' [11] is developed for generating custom GOOSE and SV messages which can be used to send customized data sets in GOOSE and SV formats.

Cyber security is a rising concern for substation communication networks [12–14] and phasor measurement units [15] as reported in the literature recently. Authors in [12], proposed Anomaly Detection System (ADS) for the substations to protect simultaneous intrusions into multiple substations which may cause severe cascading events that have catastrophic consequences. Authors in [14] reviewed cyber security challenges and potential threats in IEC 61850-substation network and summarized security measures. They argue that among different IEC 61850 services such as GOOSE, SV and MMS, GOOSE message service is of paramount importance as it carries time critical information related to power system operation. If GOOSE message service is compromised, then it may cause severe loss to power system. To address these concerns, IEC 62351-6 specifies security mechanisms for GOOSE message service [16]. The security requirements such as authentication and integrity are very important in IEC 61850 substation networks and IEC 62351-6 recommends digital signatures to mitigate cyber-attacks. Furthermore, the said standard specifies RSA digital signature algorithm for signing and verifying the GOOSE messages. The timing performance of the specified digital signature is evaluated and reported in [17,18]. Authors in [19] specify that the actual digital signature algorithm explicitly specified by IEC 62351-6 to secure GOOSE is RSA-Probabilistic Signature Scheme based on Signature Scheme with Appendix (RSASSA-PSS) based on RFC 3447 [20], which should be also compatible with RFC 2313 [21].

In order to investigate cybersecurity considerations in power system communication networks, this paper extends the GoSV framework with security implementations as Secure-GoSV (S-GoSV) [22] using openSSL libraries. In S-GoSV framework RSASSA-PKCS1-v1_5 digital signature algorithm is implemented to secure IEC 61850 GOOSE messages. However, investigations show that performance of RSASSA-PKCS1_v1-5 digital signature algorithm is not suitable for time critical GOOSE messages [19]. Therefore, authors have proposed the use of fast computing Message Authentication Code (MAC) based digital signature algorithm, Hash based Message Authentication Code- Secure Hash Algorithm (HMAC-SHA-256) [23]. HMAC-SHA-256 based implementations are also included in the S-GoSV framework for studying their performance.

The rest of the paper is organized as follows, Section 2 discusses GOOSE and SV message structures according to IEC 61850-8-1 and IEC 61850-9-2 standards, respectively. It also shows implementation details of GoSV software framework that generates custom GOOSE and Sample Values. Section 3 briefly discusses S-GoSV software that implements RSASSA-PKCS1-v1_5 digital algorithm specified by IEC 62351-6. Section 4 describes the implementation details of RSASSA-PKCS1-v1_5 digital algorithm and HMAC-SHA256 on GOOSE messages. Section 5 demonstrates the results with Wireshark captures. Section 5 draws the conclusions.

2. GoSV Framework

The GoSV framework is developed for generating and publishing customized GOOSE and SV messages. The crucial part of this framework is to accurately construct Ethernet frame according to IEC 61850-9-2 (Sample Values) and IEC 61850-8-1 (GOOSE) formats. Figure 1 shows the ethernet frame and its fields. The Ethernet frame format starts with destination and source MAC address followed by 802.1q and 802.1p (VLAN and priority tag) optional fields. The next field is 'ethertype' which defines the Ethertype Protocol Data unit (PDU) in the frame. The value of the 'ethertype' field for GOOSE and SV messages are 0x88-0xB8 and 0x88-0xBA, respectively. All the fields of 'Ethertype PDU' are encoded according to ASN.1 BER as specified in ISO/IEC 8825-1. All the fields are in triplet format of Tag, Length and Value (TLV). Again, the value can contain a new set of triplets in itself as shown in Figure 2. 'Ethertype PDU' consists of 'APPID', 'Length', 'Reserved1', 'Reserved2' followed by either Sample Values or GOOSE message fields. Ethernet frame ends with Frame Check Sequence (FCS)

field. The 'APPID' field of 2 bytes contains the tag of 'Ethertype PDU'. For 'Ethertype SV PDU' the 'APPID' value can be 0x4000 to 0x7FFF. The length field of 2 bytes contains the value of the total length of the Ethertype SV or GOOSE PDU and it should be less than 1501 bytes. The 'reserved1' field of the Ethertype SV or GOOSE PDU are reserved for security related information when the SV or GOOSE is sent with IEC 62351-6 security considerations. The 'reserved2' field is of 2 bytes to store 16-bit Cyclic Redundancy Check (CRC) value. Figures 3 and 4 gives the description of set of fields that can be interpreted as T-L-V format.

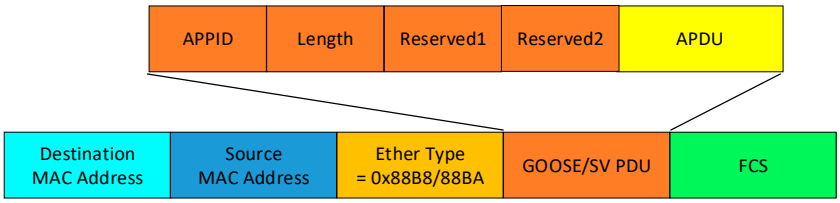

Figure 1. Ethernet frame.

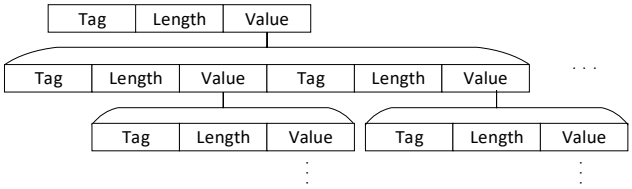

Figure 2. Basic Encoding rule followed by Tag, Length and Value (TLV).

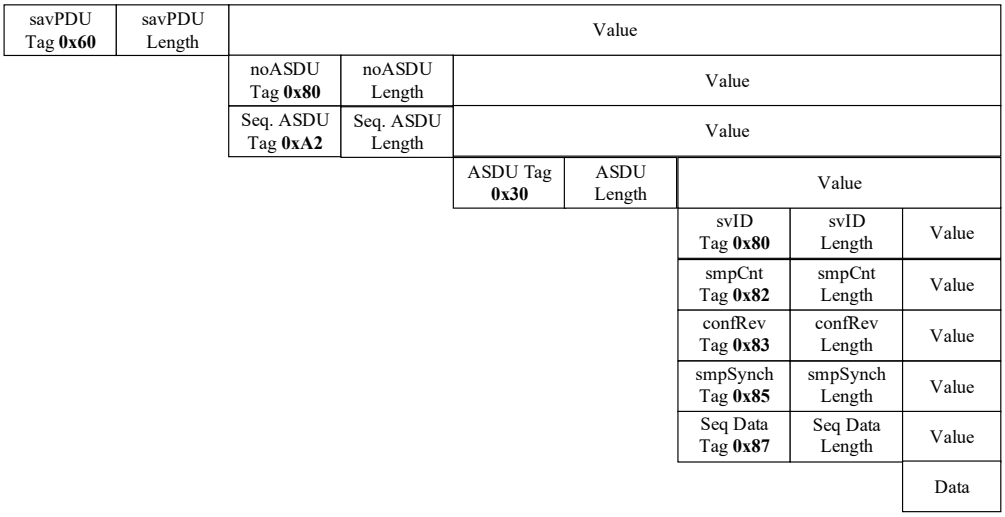

Figure 3. Description of IEC 61850-9-2 Sample Values payload fields.

The SV APDU starts with 'savPdu' field with a tag of 0x60 followed by its length. The 'noASDU' field (number of ASDU) has a tag 0x80 followed by its length and value. The 'Sequence of ASDU' field (Sequence of ASDU) begins with a tag 0xA2 followed by its length and value. The 'ASDU' field (Application Service Data Unit) has a tag 0x30 followed by its length and value. Each ASDU consists of the following fields: 'svID' (sample value ID), 'smp_Cnt' (sampling count), 'confRev' (count of configuration changes), 'smpSynch' (sample synchronization), 'Sequence of Data' (sequence of data) and 'data' (the actual dataset). The 'svID' field begins with a tag 0x80 followed by its length and value. The 'smp_Cnt' field has tag 0x82 followed by its length and value. The 'confRev' field has a tag 0x83 followed by its length and value. The 'smpSynch' field begins with tag of 0x85 followed by its length and value. The 'Sequence of Data' field begins with tag of 0x87 followed by its length and value. The final field is the actual data set. Figure 3 describes the Tag, length, value triplets of every field of SV message.

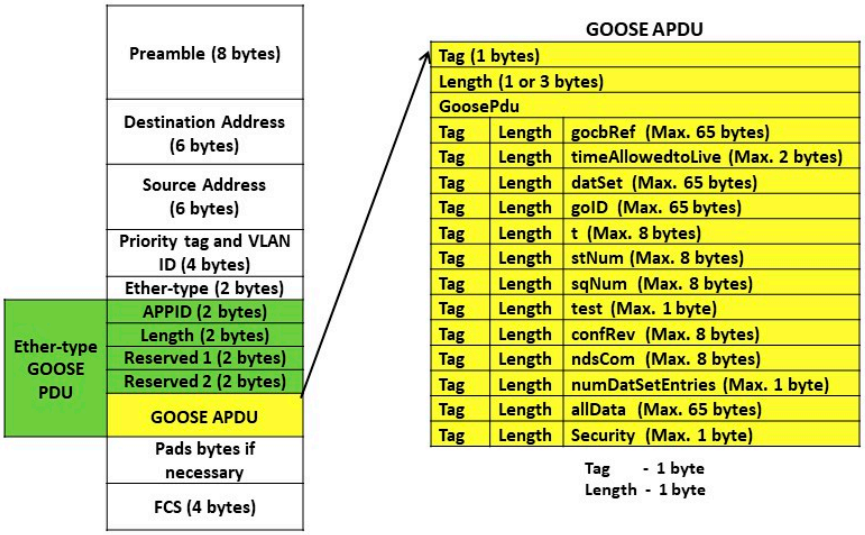

Figure 4. TLV format of pay load fields of GOOSE and Sample Values.

Similarly, PDU of GOOSE consists of following fields as gocbRef, 'timeAllowedtoLive', 'datSet', 'goID', 't', 'stNum', 'sqNum', 'test', 'confRev', 'ndsCom', 'numDatSetEntries', 'allData' as shown in Figure 4. The frame format of GOOSE message along with description of its fields is given in Table 1.

The GoSV software framework is implemented in C programming language using Ethernet libraries available in ether.h, sys/socket.h and sys/ioctl.h header files. The implementation can be accomplished at layer 2 of Open Systems Interconnect (OSI) reference model. Figure 5 describes the implementation details. First it is needed to configure the network device at the sending end of the GOOSE and Sample Values. It includes the following steps: The packet interface is used to create a raw socket on the device. Linux supports ioctls (input-output controls) to configure network devices. For configuring network device, it is needed to set the structure called ifreq (Interface request) with the name and index of the interface using ioctls. Structure ifreq facilitates low level access to Linux network devices. Device independent physical layer structure called sockaddr_ll is used to set the destination MAC address of the packet. A buffer with the max size of 2048 is taken to construct the ethernet frame. Construction of payload field for GOOSE/SV protocol is considered by taking standard fields in the form of cascading tag, length and value. Authors have uploaded the developed version of GoSV [11] in the GitHub repository.

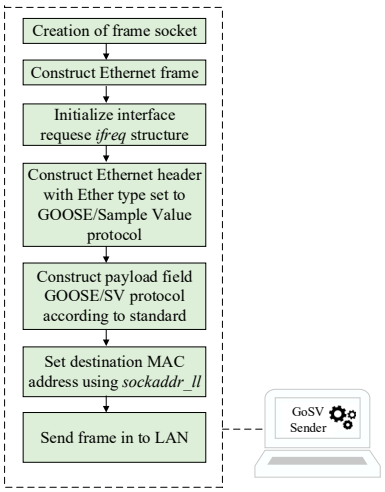

Figure 5. Frame generation process.

Table 1. GOOSE payload fields.

Field Name		Hex Value/Value in Bytes	Size (in Bytes)	Description
goosePDU Tag goosePDU Length		0x6181 145	2 2	GOOSE protocol data unit
goosePDU Value field	gocbRef_Tag gocbRef_length gocbRef_value	0x80 26	1 1 26	Reference to the associated GoCB that is controlling the GOOSE message
	timeAllowedtoLive_Tag timeAllowedtoLive_Length timeAllowedtoLive_value	0x81 3	1 1 3	The time a receiver waits before receiving a re-transmitted
	Dataset_Tag Dataset_length Dataset_value	0x82 24	1 1 24	Name of the Data Set
	goID_tag goID_length goID_value	0x83 11	1 1 11	ID of publishing IED
	Time_tag Time_length Time_value	0x84 8	1 1 8	Time stamp indicating a new GOOSE event
	st_Num_tag st_Num_length st_Num_value	0x85 1	1 1 1	Status Number—counter that increments with every GOOSE event
	sq_Num_tag sq_Num_length sq_Num_value	0x86 1	1 1 1	Sequence Number—counter that increments with every repeated GOOSE message
	Test_tag Test_length Test_value	0x87 1	1 1 1	Specifies if a message is/is not intended for testing
	confRev_tag confRev_length confRev_value	0x88 1	1 1 1	Number of times Data Set has changed
	ndsCom_tag ndsCom_length ndsCom_value	0x89 1	1 1 1	Needs commissioning field
	numDatSetEntries_tag numDatSetEntries_length numDatSetEntries_value	0x8A 1	1 1 1	Number of data elements in 'allData' field
	All_data_tag All_data_length All_data_value	0xAB 32	1 1 32	Actual data being sent (bool, integer, float, etc.)

3. Secure GoSV Framework

Security is a growing concern in power system communication networks. Compromising it may lead to plethora of security attacks such as Data integrity attack, replay attack, Man in The Middle (MITM) attack, Denial of Service (DoS) attack etc. These attacks may cause catastrophic damages to power grid. IEC 62351-6 specifies authentication and integrity as the security requirements and recommends using of digital signatures to protect IEC 61850 GOOSE or SV messages from cyber-attacks. Further, the IEC 62351-6 standard recommends RSASSA-PKCS1-v1_5 to Secure GOOSE and SV. However, the RSA based digital signature algorithms exhibit higher computational delays and do not meet the strict timing requirement of 3 ms for GOOSE messages [19]. The computational times required for generating digital signatures using RSASSA-PKCS1-v1_5 are given in the Table 2. Accordingly, Keyed Hash based Message Authentication Code (HMAC) based digital are proposed in literature for securing GOOSE messages [23].

In this paper, authors developed S-GoSV software framework that implements both RSASSA-PKCS1-v1_5 digital signature and HMAC algorithms. Authors considered securing Sample Value messages as a future work.

Table 2. Computational time for RSASSA-PKCS1-v1_5 digital signature algorithm.

DS Algorithm	Key Size (bits)	DS Signing Time (ms)	DS Verification Time (ms)	Processor	Reference
RSASSA-PKCS1-v1_5	1024 2048	0.942 3.56	0.283 0.75	Intel i5-3210M CPU @2.50GHz	[19]

3.1. RSASSA-PKCS1-v1_5 Digital Signature Algorithm

Development of secure S-GoSV software framework for RSASSA-PKCS1-v1_5 digital signature is done in two steps. In the first step, the digital signature is generated using RSASSA-PKCS1-v1_5 digital signature algorithm. The first step is implemented using python Crypto.PKCS1_v1_5 libraries. In the second step, the generated digital signature is transmitted using GoSV software which is developed using C language by adding extension fields.

Digital signatures for GOOSE message are generated in two major steps. First, an EMSA encoded message (EAPDU) generated for the GOOSE message. The EMSA encoding involves generation of hash value using SHA256 and addition of paddings [19]. Second, the generated encoded message is converted to octet string format and then signed by digital signature algorithm RSASSA-PKCS1-v1_5. The complete process of digital signature generation and verification for RSASSA-PKCS1-v1_5 is described in [19]. Figure 6 outlines the generation and verification of digital signature using RSASSA-PKCS1-v1_5 algorithm. GOOSE message along with appended DS is sent to subscriber in the network. At subscriber, a new EMSA encoded message (newEAPDU) is generated with the received GOOSE message and the received DS is decrypted by using the public key of the publisher. This decrypted digital signature value (EAPDU) is compared with the a new EMSA encoded value (newEAPDU) generated at the subscriber. If both are the same, then GOOSE message is valid otherwise it will not be accepted as a legitimate message.

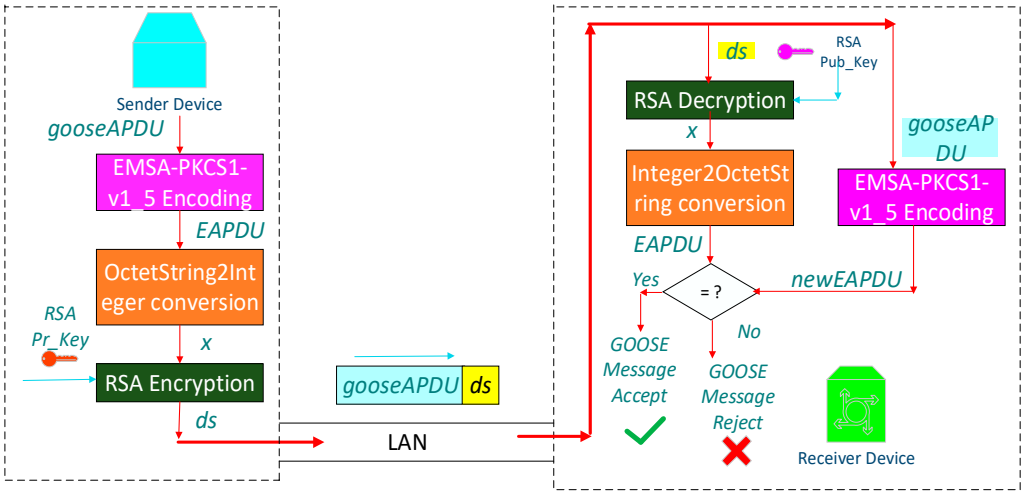

Figure 6. Generation and verification using of digital signature RSASSA-PKCS1-v1_5.

In order to carry this additional Digital Signature (DS) value in the ethernet frame, IEC 62351-6 standard specifies extension to be added at the end of ethernet frame as shown in the Figure 7. Extension field consists of SEQUENCE field and authentication value. SEQUENCE field is reserved for future security considerations other than encryption and message authentication. Authentication value contains digital signature value generated by RSASSA-PKCS1-v1_5.

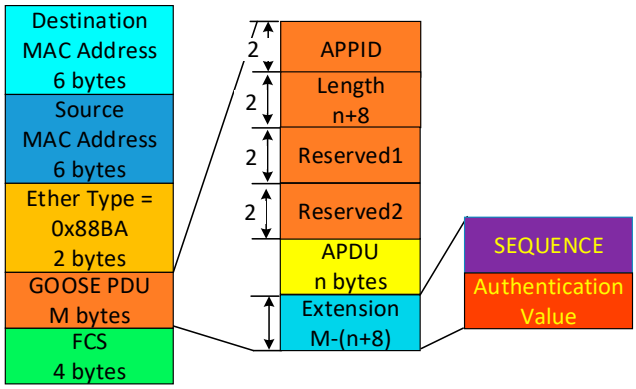

Figure 7. Extension field of GOOSE APDU according to IEC 62351-6.

RSASSA-PKCS1-v1_5 (Public Key Cryptography Standard #1 version 1.5) digital signature algorithm generates digital signature in three steps, as shown in Algorithm 1 below. In the first step, it takes GOOSE APDU as input and performs encoding using EMSA_PKCS1-v1_5 encoding scheme. The output of the encoding scheme is Encoded APDU (EAPDU). In the second step, EAPDU is converted into integer representation. In the final step, integer representation is encrypted with private key of RSA algorithm that gives digital signature (ds).

Algorithm 1. RSASSA_PKCS1-v1_5_Gen_Signature (gooseAPDU)

$(Pub\_Key,\ \mathrm{Pr}\_Key) \leftarrow Gen\_Keys()$
$EAPDU \leftarrow EMSA\_PKCS1\_v1\_5\_Encode(gooseAPDU)$
$x \leftarrow OctetString2Integer(EAPDU)$
$ds \leftarrow RSAEnc_{\mathrm{Pr}\_Key}(x)$
return ds

Algorithm 2. RSASSA_PKCS1-v1_5_Signature_Verify (gooseAPDU, ds)

$(Pub\_Key,\ \mathrm{Pr}\_Key) \leftarrow Gen\_Keys()$
$x \leftarrow RSADec_{\mathrm{Pub}\_Key}(ds)$
$EAPDU \leftarrow Integer2OctetString(x)$
$newEAPDU \leftarrow EMSA\_PKCS1\_v1\_5\_Encode(gooseAPDU)$
if $(EAPDU = newEAPDU)$
then
    received GOOSE message valid
else
    received GOOSE messge is invalid
return False

At the publisher device, Algorithm 2 takes input a GOOSE APDU (*gooseAPDU*) and gives encoded code (*EAPDU*) using EMSA_PKCS1-v1_5 encoding algorithm. The output of encoding scheme is converted into integer representation (x) which is further signed with RSA private key (*Pr_Key*) that generates digital signature (ds). At the subscriber device, RSASSA_PKCS1-v1_5_Signature_verify algorithm verifies the generated signature (*ds*). The received digital signature (*ds*) is decrypted with RSA public key (*Pub_Key*) that gives integer representation x. Further, integer representation (x) value is converted into octet string that gives Encoded APDU (*EAPDU*). A new Encoded APDU (*newEAPDU)* is generated by taking received GOOSE APDU (*gooseAPDU*) as input to EMSA Encoding scheme. Newly generated EAPDU and received EAPDU is compared, if both are same then received GOOSE APDU (*gooseAPDU)* is valid otherwise it is invalid. Figure 6 illustrates the signature generation and verification process based on RSASSA_PKCS1-v1_5 digital signature algorithm.

3.2. HMAC

Keyed Hash based Message Authentication Code (HMAC) is used for data origin authentication and integrity verification mechanisms in the network communications. It can be combined with either of the three different Secure Hash Algorithms: SHA256, SHA384 and SHA512. Secure Hash algorithms are cryptographic hash functions that generate hash values. HMAC combined with either of the above Secure Hash algorithms can be employed to generate digital signature instead of asymmetric RSASSA-PKCS1-v1_5 specified by IEC 62351-6. In this paper, among the above variants we have chosen HMAC with SHA256 algorithm for digital signature calculation. The main goal of these variants is to ensure the authenticity of data and it is not modified during its transmission in the network. HMAC implementation requires a pre-shared key shared by both publisher and subscriber. The security of the HMAC is ensured by the secure distribution and unpredictability of the associated secret symmetric key. The size of input data is arbitrary and produces a fixed size output. For HMAC-SHA256 where key size is 256 bits, it produces 32 bytes of digital signature.

In S-GoSV framework the HMAC algorithm is implemented using openSSL/hmac C libraries. The openSSL/hmac libraries are used to generate the MAC value (i.e., digital signature), using the pre-shared key, which is appended to GOOSE message and sent to subscriber. At the subscriber, again the MAC value is calculated for the received GOOSE message using the pre-shared key. The newly generated MAC value at the subscriber is compared with the received MAC value. If they do not match the GOOSE message is rejected. Figure 8 shows the process of securing GOOSE messages with HMAC algorithm.

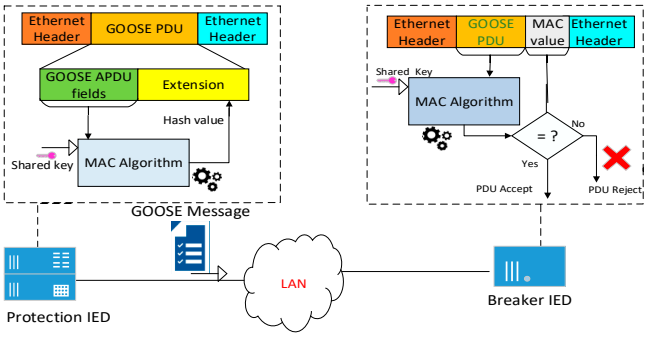

Figure 8. HMAC integration to GOOSE messages for secure communication.

4. Implementation Results

4.1. GoSV

Initially, GoSV software framework is implemented that generates custom GOOSE and Sample Values using C library. A testbed consists of three laptop computers connected in a Local Area Network (LAN) environment as shown in the Figure 9. GoSV software program running in a laptop computer constructs custom GOOSE and Sample Values according to TLV formats described in the Figures 3 and 4 and send them in to LAN. Figures 10 and 11 shows the screenshots of GoSV software program sending custom GOOSE and Sample Value frames respectively. In order to test the validity of messages generated by GoSV, Infotech Avenue SAV receiver [24] software is run at the destination node which runs Windows OS. The stream was successfully detected as shown in the Figure 12 where the input stream from GoSV is listed as a legitimate SV message. Further to test the legitimacy of the generated GOOSE and SV messages, Libiec61850's SV subscriber [10] module is run at the receiving node. Libiec61850 SV subscriber run in Linux platform. The packets are successfully received as SV messages and unpacked as defined in IEC 61850 standard. Figure 12 shows how libiec61850 SV subscriber receives SV messages generated by GoSV software. In addition to this, Wireshark network

sniffer software tool identifies GoSV frames as IEC SV and IEC GOOSE messages and all the required fields are successfully decoded as shown in the Figures 13 and 14.

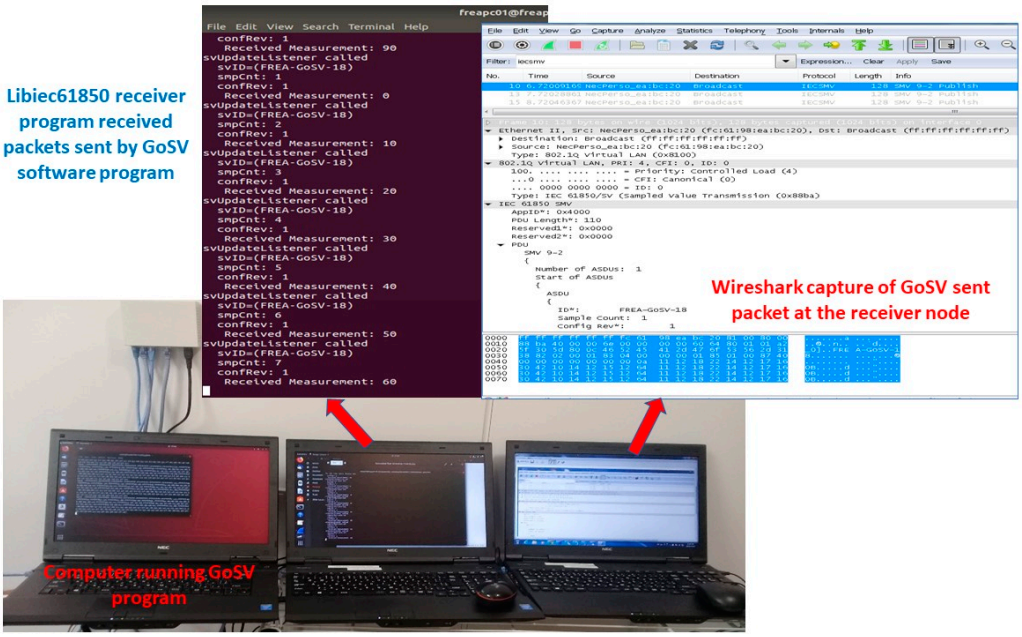

Figure 9. LAN set up to test the developed software 'GoSV'.

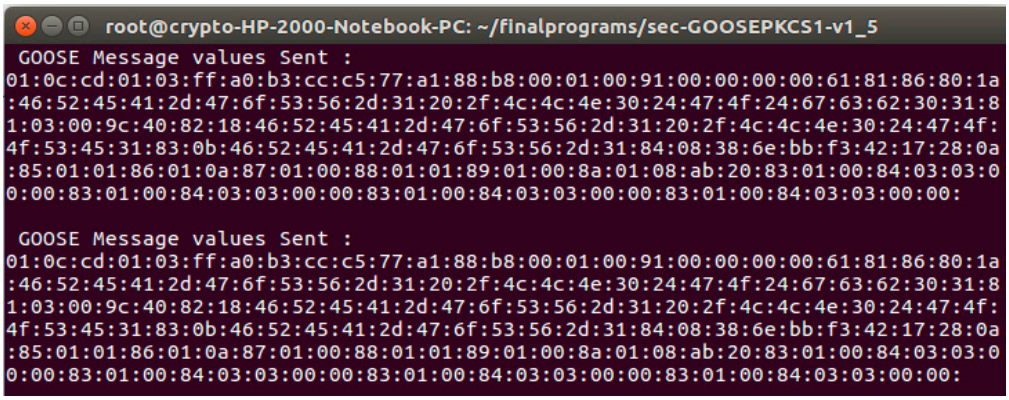

Figure 10. GoSV software program sending custom GOOSE frames.

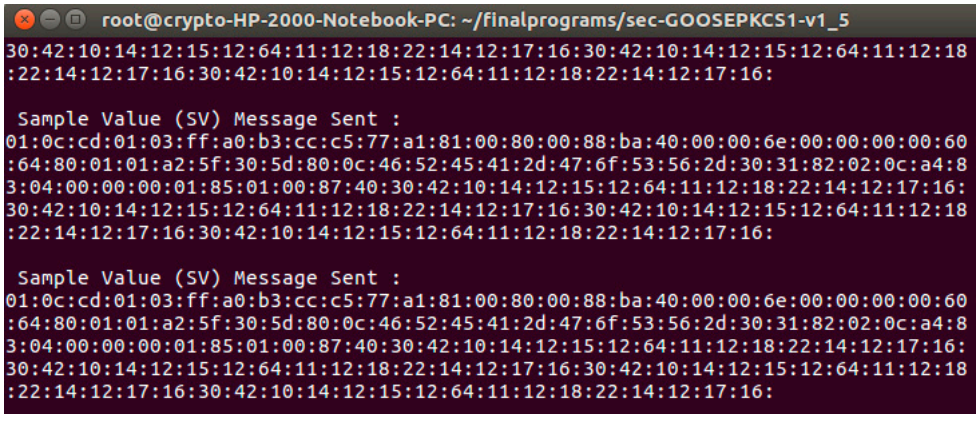

Figure 11. GoSV software program sending custom SV frames.

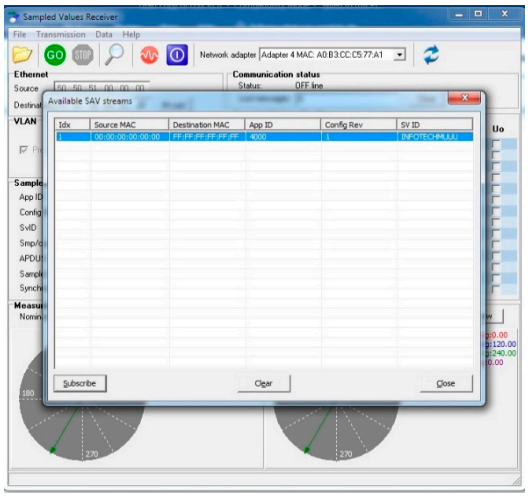

Figure 12. Infotech Avenue Receiver detecting the stream of GoSV software framework.

```
▷ Frame 1: 159 bytes on wire (1272 bits), 159 bytes captured (1272 bits) on
▷ Ethernet II, Src: HewlettP_c5:77:a1 (a0:b3:cc:c5:77:a1), Dst: Iec-Tc57_01
◢ GOOSE
    APPID: 0x0001 (1)
    Length: 145
    Reserved 1: 0x0000 (0)
    Reserved 2: 0x0000 (0)
  ◢ goosePdu
      gocbRef: FREA-GoSV-1 /LLN0$GO$gcb01
      timeAllowedtoLive: 40000
      datSet: FREA-GoSV-1 /LLN0$GOOSE1
      goID: FREA-GoSV-1
      t: Jan  2, 2000 02:46:11.258165836 UTC
      stNum: 1
      sqNum: 10
      test: False
      confRev: 1
      ndsCom: False
      numDatSetEntries: 8
```

```
0000  01 0c cd 01 03 ff a0 b3  cc c5 77 a1 88 b8 00 01   ........ ..w.....
0010  00 91 00 00 00 00 61 81  86 80 1a 46 52 45 41 2d   ......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      ........ .......
```

Figure 13. Wireshark capture of GOOSE frame values.

```
▷ Frame 1: 128 bytes on wire (1024 bits), 128 bytes captured (1024 bits) on
▷ Ethernet II, Src: HewlettP_c5:77:a1 (a0:b3:cc:c5:77:a1), Dst: Iec-Tc57_01
▷ 802.1Q Virtual LAN, PRI: 4, DEI: 0, ID: 0
◢ IEC61850 Sampled Values
    APPID: 0x4000
    Length: 110
    Reserved 1: 0x0000 (0)
    Reserved 2: 0x0000 (0)
  ◢ savPdu
      noASDU: 1
    ◢ seqASDU: 1 item
      ◢ ASDU
          svID: FREA-GoSV-01
          smpCnt: 3236
          confRef: 1
          smpSynch: none (0)
          seqData: 3042101412151264111218221412171630421014121512 64...
```

```
0000  01 0c cd 01 03 ff a0 b3  cc c5 77 a1 81 00 80 00   ........ ..w.....
0010  88 ba 40 00 00 6e 00 00  00 00 60 64 80 01 01 a2   ..@..n.. ..`d....
0020  5f 30 5d 80 0c 46 52 45  41 2d 47 6f 53 56 2d 30   _0]..FRE A-GoSV-0
0030  31 82 02 0c a4 83 04 00  00 00 01 85 01 00 87 40   1....... .......@
0040  30 42 10 14 12 15 12 64  11 12 18 22 14 12 17 16   0B.....d ..."....
0050  30 42 10 14 12 15 12 64  11 12 18 22 14 12 17 16   0B.....d ..."....
0060  30 42 10 14 12 15 12 64  11 12 18 22 14 12 17 16   0B.....d ..."....
0070  30 42 10 14 12 15 12 64  11 12 18 22 14 12 17 16   0B.....d ..."....
```

Figure 14. Wireshark capture of SV frame values.

4.2. Secure-GoSV

As security of GOOSE messages is of prime importance which transmits time critical information, compromising it causes devastating consequences in the grid. Authors of this paper further extended the implementation of GoSV software framework to protect GOOSE messages using RSASSA-PKCS1-v1_5

digital signature algorithm as specified by IEC 62351-6 using openSSL library. Protecting Sample Value messages can be further implemented in the future works.

Figure 15 shows the Wireshark capture of implementation of RSASSA-PKCS1-v1_5 digital signature algorithm. Reserved1 and Reserved2 fields of GOOSE APDU fields are used when security mechanism is applied. In order to carry additional value of digital signature, an extension field which consists of SEQUENCE, Authentication value in tag-length-value (TLV) format is constructed in the program. Figure 15 shows tag value for SEQUENCE field is 0x30 and length is 0x24. Authentication field tag value is 0xA4 and length is 0x22. Finally, digital signature tag value is 0x85, length value is 0x80 followed by 128 bytes of generated digital signature value by RSASSA-PKCS1-v1_5 digital signature algorithm. Similarly, Figure 16 shows the Wireshark capture of implementation of HMAC algorithm for securing GOOSE message.

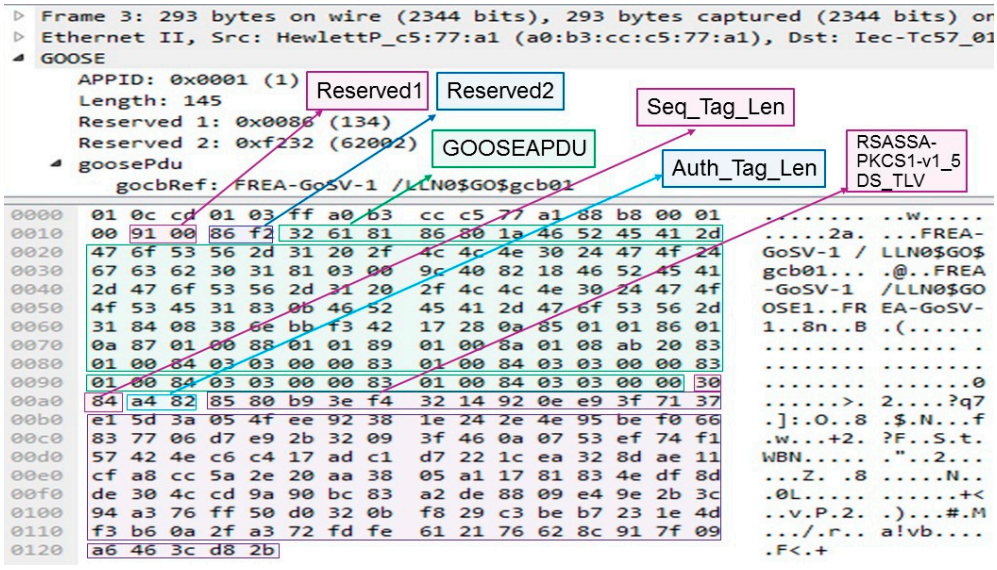

Figure 15. Wireshark capture of GOOSE frame with the extension of RSASSA-PKCS1-v1_5 digital signature.

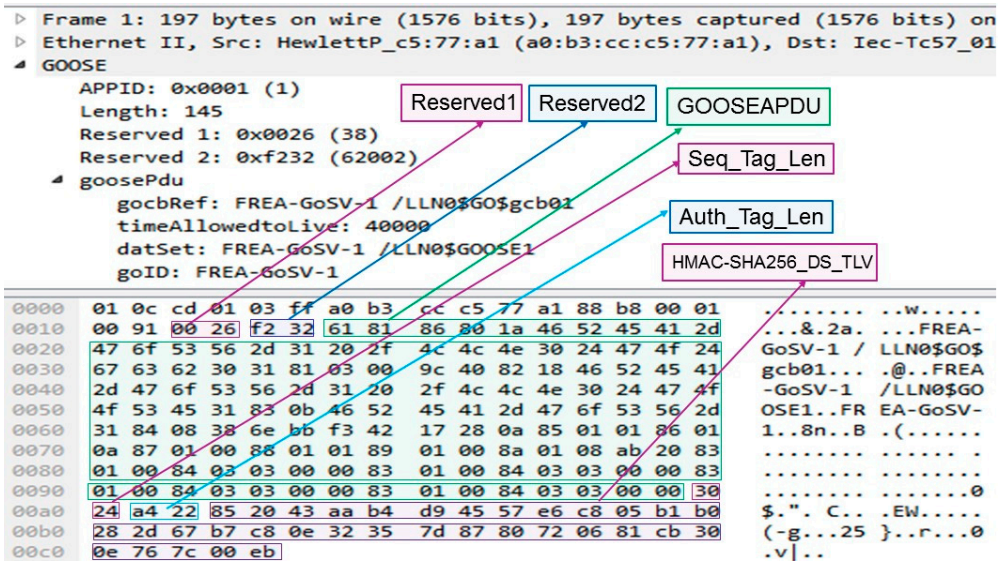

Figure 16. Wireshark capture of GOOSE frame with the extension of HMAC-SHA256 digital signature.

5. Conclusions

IEC 61850 standard is receiving more attention as it is poised to become the power system communication standard of the future. Much of the research focuses on customizing IEC 61850 information models as well as message structures to accommodate new smart grid equipment. Current software available for GOOSE and SV messages do not have the capability to customize their contents and properties, such as contained data and sampling rate. However, performance studies necessitate the existence of a tool that can construct and publish custom GOOSE and SV messages. GoSV software is developed to fill this gap. Lab tests show that messages published by GoSV strictly conform to IEC 61850 format as they are successfully identified by several other software such as Infotech Avenue receiver software which runs on Windows OS platform and Libiec61850 Subscriber module which runs on Linux. Furthermore, Wireshark network sniffer software tool also successfully decodes all the fields of generated custom GoSV frames. In addition to publishing messages, security requirements for GOOSE messages is implemented based on IEC 62351-6 which specifies digital signatures to achieve message authentication and integrity. GoSV software is extended as S-GoSV to implement RSASSA-PKCS1-v1_5 digital signature based on IEC 62351-6 recommendation. After running performance studies with S-GoSV, it is found that RSA based digital signatures do not meet the timing requirements of power system communication and an alternative is required. MAC based digital signature algorithm HMAC-SHA256 is implemented for GOOSE messages. For future work, it is possible to secure Sample Value messages in the substation communication network.

Author Contributions: All authors contributed equally.

Funding: This work was supported by Research and Innovation Fund 2018.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ali, I.; Hussain, S.M.S.; Tak, A.; Ustun, T.S. Communication modeling for differential protection in IEC-61850-based substations. *IEEE Trans. Ind. Appl.* **2018**, *54*, 135–142. [CrossRef]
2. *Communication Networks and Systems for Power Utility Automation, 2.0.*; Standard IEC 61850; IEC: Geneva, Switzerland, 2013.
3. Aftab, M.A.; Roostaee, S.; Hussain, S.M.S.; Ali, I.; Thomas, M.; Mehfuz, S. Performance Evaluation of IEC 61850 GOOSE based inter substation communication for accelerated distance protection scheme. *IET Gener. Transm. Distrib.* **2018**, *12*, 4089–4098. [CrossRef]
4. Ustun, T.S.; Ozansoy, C.; Zayegh, A. A central microgrid protection system for networks with fault current limiters. In Proceedings of the 2011 10th International Conference on Environment and Electrical Engineering, Rome, Italy, 8–10 May 2011; pp. 1–4.
5. Ali, I.; Thomas, M.S.; Gupta, S.; Hussain, S.M.S. Information modeling for Distributed Energy Resource integration in IEC 61850 based substations. In Proceedings of the 2015 Annual IEEE India Conference (INDICON), New Delhi, India, 17–19 December 2015; pp. 1–6.
6. Hussain, S.M.S.; Ustun, T.S.; Nsonga, P.; Ali, I. IEEE 1609 WAVE and IEC 61850 Standard Communication Based Integrated EV Charging Management in Smart Grids. *IEEE Trans. Veh. Technol.* **2018**, *67*, 7690–7697. [CrossRef]
7. Hussain, S.M.S.; Tak, A.; Ustun, T.S.; Ali, I. Communication Modeling of Solar Home System and Smart Meter in Smart Grids. *IEEE Access* **2018**, *6*, 16985–16996. [CrossRef]
8. Hegazi, O.; Hammad, E.; Farraj, A.; Kundur, D. IEC-61850 GOOSE traffic modeling and generation. In Proceedings of the 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP), Montreal, QC, Canada, 14–16 November 2017; pp. 1100–1104.
9. Lopes, Y.; Muchaluat-Saade, D.C.; Fernandes, N.C.; Fortes, M.Z. Geese: A traffic generator for performance and security evaluation of IEC 61850 networks. In Proceedings of the 2015 IEEE 24th International Symposium on Industrial Electronics (ISIE), Buzios, Brazil, 3–5 June 2015; pp. 687–692.
10. LibIEC61850. Available online: http://libiec61850.com/libiec61850/ (accessed on 1 July 2019).
11. GoSv. Available online: https://github.com/61850security/GoSV (accessed on 1 July 2019).

12. Hong, J.H.; Liu, C.-C.; Govindarasu, M. Integrated anomaly detection for cyber security of the substations. *IEEE Trans. Smart Grid.* **2014**, *5*, 1643–1653. [CrossRef]
13. Cai, J.; Zheng, Y.; Zhou, Z. Review of cyber-security challenges and measures in smart substation. In Proceedings of the 2016 International Conference on Smart Grid and Clean Energy Technologies (ICSGCE), Chengdu, China, 19–22 October 2016; pp. 65–69.
14. Chattopadhyay, A.; Ukil, A.; Jap, D.; Bhasin, S. Toward threat of implementation attacks on substation security: Case study on fault detection and isolation. *IEEE Trans. Ind. Inform.* **2018**, *14*, 2442–2451. [CrossRef]
15. Farooq, S.M.; Hussain, S.M.S.; Kiran, S.; Ustun, T.S. Certificate based authentication mechanism for PMU communication networks based on IEC 61850-90-5. *Electronics* **2018**, *7*, 370. [CrossRef]
16. *Power Systems Management and Associated Information Exchange—Data and Communications Security, Part 6: Security for IEC 61850, IEC/TS 62351-6:2007(E)*; IEC: Geneva, Switzerland, 2007.
17. Hohlbaum, F.; Braendle, M.; Fernando, A. Cyber security practical considerations for implementing IEC 62351. In Proceedings of the PAC World Conference, Dublin, Ireland, 28–90 June 2010; pp. 21–24.
18. Ishchenko, D.; Nuqui, R. Secure communication of intelligent electronic devices in digital substations. In Proceedings of the IEEE PES Transmission and Distribution Conference & Exposition, Denver, CO, USA, 16–19 April 2018; p. 115.
19. Farooq, S.M.; Hussain, S.M.S.; Ustun, T.S. Performance Evaluation and Analysis of IEC 62351-6 Probabilistic Signature Scheme for Securing GOOSE Messages. *IEEE Access* **2019**, *7*, 32343–32351. [CrossRef]
20. Jonsson, J.; Kaliski, B. *Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1*; Document RFC 3447; IETF: Fremont, CA, USA, 2003.
21. Jonsson, J.; Kaliski, B. *Public-Key Cryptography Standards (PKCS) #1: RSA Encryption Version 1.5*; Document RFC 2313; IETF: Fremont, CA, USA, 1998.
22. S-GoSV. Available online: https://github.com/61850security/S-GoSV-part-1 (accessed on 1 July 2019).
23. Hussain, S.M.S.; Farooq, S.M.; Ustun, T.S. Analysis and Implementation of Message Authentication Code (MAC) Algorithms for GOOSE Message Security. *IEEE Access* **2019**, in press. [CrossRef]
24. SAV Sender and Receiver—INFO TECH. Available online: https://goo.gl/8yLw9A (accessed on 26 June 2019).