A Lightweight Integrity Protection Scheme for Fast Communications in Smart Grid

Alireza Jolfaei and Krishna Kant
Department of Computer Science, Temple University, Philadelphia, U.S.A.

Keywords: GOOSE Message, Integrity Protection, Permutation, Phasor Measurement, Substation Automation System.

Abstract: Due to the mission-critical nature of energy management, smart power grids are prime targets for cyber-attacks. A key security objective in the smart grid is to protect the integrity of synchronized real-time measurements taken by phasor measurement units (PMUs). The current communication protocol in substation automation allows the transmission of PMU data in absence of integrity protection for applications that strictly require low communication latency. This leaves the PMU data vulnerable to man-in-the-middle attacks. In this paper, a lightweight and secure integrity protection algorithm has been proposed to maintain the integrity of PMU data, which fills the missing integrity protection in the IEC 61850-90-5 standard, when the MAC identifier is declared 0. The rigorous security analysis proves the security of the proposed integrity protection method against ciphertext-only attacks and known/chosen plaintext attacks. A comparison with existing integrity protection methods shows that our method is much faster, and is also the only integrity protection scheme that meets the strict timing requirement. Not only the proposed method can be used in power protection applications, but it also can be used in emerging anomaly detection scenarios, where a fast integrity check coupled with low latency communications is used for multiple rounds of message exchanges.

1 INTRODUCTION

The supervisory control and data acquisition (SCADA) systems in emerging smart grids monitor, control and protect power system components by making use of a network of phasor measurement units (PMUs) and phasor data concentrators (PDC). As shown in Figure 1, synchronous PMUs and PDCs are installed in various locations (key substations) of the transmission and distribution lines. PMUs measure voltage magnitude, phase, and line frequency 30 to 60 times a second; and send these measurements along with the global position system (GPS) location of the PMU to PDCs. This is done through a publish-subscribe mechanism, where the PMUs work as publishers to which the PDCs subscribe. A PDC receives data from many (typically 3 to 32) PMUs, and then sorts and aggregates the received data based on the time-tag. The aggregated data is then relayed using a two-way communication system to a number of local control centers (LCCs), which coordinate their actions interacting with a federated control center (FCC). Subsequently, LCCs draw the best overall snapshot solution using all PMU measurements (Weng et al., 2016).

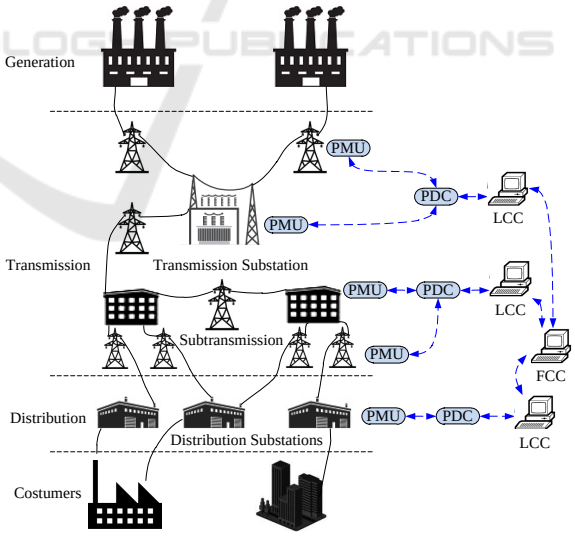

Figure 1: Typical model of a power grid architecture.

Control centers use the IEC 61850-90-5 standard to communicate with smart measurement units (IEC 61850-90-5, 2012). This standard utilizes a generic object oriented substation event (GOOSE) protocol for fast relay-to-relay communications. Since GOOSE messages are time critical, they are only as-

Jolfaei, A. and Kant, K.
A Lightweight Integrity Protection Scheme for Fast Communications in Smart Grid.
DOI: 10.5220/0006394200310042
In *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017) - Volume 4: SECRYPT*, pages 31-42
ISBN: 978-989-758-259-2

Table 1: Size of payload data under various PMU/PDC configurations

Number of PMU voltage channels	PMU data size (bytes)	PDC data size			Applicability
		3 PMUs	6 PMUs	10 PMUs	
1	40	104	200	328	Single phase line in distribution systems
3	56	120	216	344	3-phase power transmission lines
6	80	144	240	368	power transmission lines
8	96	160	256	384	power transmission lines
10	112	176	272	400	power transmission lines

Table 2: Allowed values for MAC value calculation.

Identifier	MAC algorithm	Key size (bits)
0	None	None
1	SHA-256	80
2	SHA-256	128
3	SHA-256	256
4	AES-GMAC	64
5	AES-GMAC	128

sociated with three layers of the open systems interconnection (OSI) model, namely, physical, data-link, and application layers. IEC 61850 utilizes a standardized Ethernet frame with a maximum payload size of 1492 bytes (Fan et al., 2015). The number of required measurements depends on the topology of the power grid as well as the requirements of optimal estimation of the system states (Martin, K., Zwergel, A., and Kapostasy, D., 2015). Generally, in each substation, the principal bus voltages should be included in the measurement set. To this end, several PMUs are normally used to cover a large substation. The size of measurement data depends on the number of voltages that a PMU measures. Table 1 gives the size of payload data in transmission and distribution substations under various PMU/PDC configurations.

Although the utilization of PMUs has facilitated the efficient management and delivery of power in current smart grids, it is vulnerable to integrity attacks. The standard for formatting and delivery of PMU data (IEEE Standard C37.118 (Martin et al., 2008)) includes no end-to-end security mechanisms, and transmits messages in plaintext without any mechanism to protect their integrity. Although, a cyclic redundancy check (CRC) (Sobolewski, 2003) is used in the PMU data frame, the integrity protection of a CRC is null, as it cannot detect intentional tampering. To this end, IEC 61850-90-5 mandates the use of a number of message authentication code (MAC) algorithms (Table 2), which are determined by an identifier in the message header based upon latency requirements. However, the complete deployment of this standard will take a long time (Seyed Reza et al., 2016) because of the compatibility issues of the standard as well as the hardware issues. For example, there are still many PMUs and PDCs in operation that have no cryptographic acceleration (Pappu et al., 2013). In addition, when the latency implied by the MAC implementation cannot be tolerated, IEC 61850-90-5 allows the transmission of messages in the absence of integrity protection, as shown in Table 2. This mainly happens in protection applications where time is critical and therefore a fast message communication is required. Examples of time critical applications are load shed (Adamiak et al., 2014) and synchrophasor-assisted transfer trip (Kundu and Pradhan, 2014) where a trip signal is sent from a substation to another that could be more than 100 miles away. In such applications, the use of MAC algorithm is largely disabled by setting the identifier to zero. This leaves the PMU data vulnerable to man-in-the-middle attacks, which can alter the current phasor of the bus, for instance, to a value greater than the maximum rated current of the line or even to zero amperes. This would trigger protective relays to switch on and off, which could be costly to electrical equipment and perturb the grid, potentially leading to blackouts.

Delay requirements for power system networks depend on a number of parameters, such as the specific protection equipment used. Most power line equipment can endure faults or short circuits for up to approximately five power cycles before experiencing irreversible damage or affecting other segments in the network. This translates to total fault clearance time of less than 84 ms. However, as a safety precaution, actual operation time of protection systems is limited to approximately 70 percent of this period, including fault recognition time, command transmission time and line breaker switching time (Wetterwald P. and Raymond J., 2015). Some system components, such as large electromechanical switches, require particularly long time to operate and take up the majority of the total clearance time, leaving only a 10 ms window for the communications part of the protection scheme, independent of the distance to travel. Given the sensitivity of the issue, communication networks impose requirements that are even more stringent, for instance IEC 61850 standard limits the transfer time for protection messages to $\frac{1}{4}$ cycle or 4 ms (for 60 Hz lines) for the most critical messages.

To meet the 4 ms latency requirement, we are thus compelled to look for a *more lightweight* and a *non-hash-based* solution. A potential solution is in the use of a fast checksum, which has the advantage of having less computational cost. Checksums are common methods for detecting accidental data corruption, for instance, in TCP (Zander et al., 2007) and ZLIB (Sofia et al., 2015); and compared to MACs, they impose much less computational overheads. However, since checksums can be easily spoofed, we propose a novel integrity protection method, which hides checksum bits inside payload data using a fast bit permutation technique. *Our rigorous security analysis shows no weaknesses in the proposed method, and demonstrates no simple method of recovering the secret key. We also confirm the security of the proposed integrity protection method against ciphertext-only attacks and known/chosen plaintext attacks.* To the best of our knowledge, the proposed method is the first secure technical solution that meets the strict communication latency requirements in protective relaying in transmission and distribution substations.

The remainder of this paper is organized as follows. Section 2 summarizes the related work. Section 3 provides details of the proposed embedding and integrity verification algorithms. The security of the proposed method is evaluated using an adversarial model in Sections 4. Section 5 evaluates the performance of the proposed method with respect to computational complexity and running time, and the results are compared with previous works. Finally, Section 6 concludes that the proposed scheme is secure, efficient and feasible.

2 RELATED WORK

Recent research has mainly been focused on preserving the integrity of PMU data in applications where latency is on the order of seconds (rather than milliseconds), such as static state estimation, power flow analysis and model validation. The integrity problem in such applications can be addressed using conventional hash based techniques, such as a cipher-based MAC (CMAC) (Dworkin, 2007) and a hash-based MAC (HMAC) (Khan et al., 2007). However, these solutions cannot be used in power system protection because of the latency implied by the overheads. Despite the importance of data integrity in the protection of power equipment, only a few technological solutions have been given, (Guo et al., 2007; Zhang et al., 2008; Abuadbba and Khalil, 2015; Abuadbba et al., 2016), which mainly utilized steganographic methods. Compared to crypto primitives, steganographic methods normally require less memory, power and processing capabilities, which can suit the constrained capabilities of smart grid infrastructure.

In (Guo et al., 2007), Guo et al. proposed a fragile watermarking method for protecting the integrity of payload data. In this scheme, payload is firstly split into groups of variable size. The size of the group is determined adaptively as a function of the data itself. A secure hash function, such as MD5, is then applied to each data element in the payload. If the hash value is zero, then the data element marks the end of the group. A watermark is formed by both the current group hash value and the group hash value of the next group. The watermark is stored in the least significant bits of all data elements. However, Guo et al.'s watermarking scheme needs to compute a secure hash function several times. Therefore, it is computationally expensive for the microprocessors used in PMUs.

In (Zhang et al., 2008), Zhang et al. proposed an end-to-end, statistical approach for the integrity protection of sensory measurement data using a direct spread spectrum sequence (DSSS) based watermarking technique. In this scheme, the measurement data, which is sent from the sender nodes, is visualized as an image at different time snapshots, in which every sender node is viewed as a pixel and its value corresponds to the gray level of the pixel. Following this equivalency, a watermark is embedded in this image in a distributed fashion at each node. Given the watermark as a prior knowledge, the receiver is then able to verify the integrity of the measurement data. However, the proposed method is disclosed using known/chosen plaintext attacks, because the size of embedding data is much smaller than the size of payload data, and watermarks are generated under the same key.

In (Abuadbba and Khalil, 2015), Abuadbba and Khalil proposed a steganographic method, which protects the integrity of smart grid readings by hiding a confidential information (a fingerprint) bit-by-bit inside the detailed sub-band coefficients of the discrete wavelet transform (DWT) of the payload data. Although this method is much faster than MAC-based solutions, it cannot protect the integrity of the complete payload data, because the adversary can simply spoof a fradulent message by modifying the approximation coefficients of DWT of the payload data. In other words, only one pair of covertext and stegotext is enough to break the integrity protection algorithm. In addition, the embedding process makes irreversible distortions at the location of hidden bits in the covertext. Therefore, a portion of measurement data will be

lost, which may impact the decision making process of control centers. The transmitted stegotext is highly prone to intentional (interference) and unintentional (noise) attacks rendering the steganography based integrity protection solutions impractical. Indeed, any slight change in the transmitted stegotext will result in the loss of embedded information, and more significantly, loss of faith in the received GOOSE message. To this end, Abuadbba et al. (Abuadbba et al., 2016) combined a 3D DWT based steganographic method with an error detection and correction technique, namely, BCH syndrome codes, to detect and recover any change in the payload data. However, the size of hidden data is much less than the size of payload data, and therefore, similar to (Abuadbba and Khalil, 2015), the location of the embedded data is easily disclosed by making use of known/chosen covertext attacks.

It follows that steganography by itself cannot sufficiently address the integrity problem, and it needs to be combined with another method, such as hashing or encryption, to make the stegotext secure from attacks. To this end, we propose a novel lightweight solution for the integrity problem, which is secure from well-known attacks.

3 PROPOSED INTEGRITY PROTECTION SCHEME

The building blocks of the proposed scheme are explained in following subsections. The notations used in the explanation are listed in Table 3.

3.1 16-bit Fletcher Checksum

The 16-bit Fletcher checksum has two variants: a one's complement and a two's complement version. In this paper, we used the former, because it provides better error detection than the latter (Nakassis, 1988). The 16-bit Fletcher checksum is calculated iteratively over a sequence of 8 bit blocks, namely, $P_0, P_1, \ldots, P_{N-1}$, by maintaining two unsigned one's-complement 16-bit accumulators $\boldsymbol{R}$ and $\boldsymbol{S}$, whose contents are initially zero. The pseudo-code of the 16-bit Fletcher checksum is given in Algorithm 1. It could be shown that at the end of the Fletcher's loop, $\boldsymbol{R}$ will contain the 8-bit one's complement sum of all 8 bit blocks in the payload data, and $\boldsymbol{S}$ will contain $N \cdot P_0 + (N-1) \cdot P_1 + \cdots + P_{N-1}$. One advantage of the Fletcher algorithm is that it detects the transposition of octets/words of any size within the data stream. The error detection properties of the Fletcher checksum is comparable to CRCs (Sobolewski, 2003) with significantly reduced computational cost.

Table 3: Notations.

Notation	Description
N	The number of bytes in a plaintext
$\boldsymbol{P}$	A plaintext represented by an $8N$-bit array
$\boldsymbol{X}$	A checksum represented by a 16-bit array
$\boldsymbol{E}$	An expanded text represented by a $(8N+15)$-bit array
$\boldsymbol{C}$	A ciphertext represented by a $(8N+15)$-bit array
$p(i)$	A binary value at the position i $(0 \leq i \leq 8N-1)$ of the plaintext
$x(d)$	A binary value at the position d $(0 \leq d \leq 15)$ of the checksum
$e(w)$	A binary value at the position w $(0 \leq w \leq 8N+15)$ of the expanded text
$c(j)$	A binary value at the position j $(0 \leq j \leq 8N+15)$ of the ciphertext
K	The set of secret key
P	The set of N bytes in a plaintext $\boldsymbol{P}$, that is, $P = \{P_0, P_1, \ldots, P_{N-1}\}$
C	The set of $N+2$ bytes in a ciphertext $\boldsymbol{C}$, that is, $C = \{C_0, C_1, \ldots, C_{N+1}\}$
L	The set of byte locations for a plaintext $\boldsymbol{P}$ with N bytes, that is, $L = \{l \mid l = 0, 1, \ldots, N-1\}$
$Pr(\boldsymbol{P_t}, \boldsymbol{C_t})$	The probability that the resulting sequence of the t-th query and response is $(\boldsymbol{P_t}, \boldsymbol{C_t})$
$Pr(\boldsymbol{C}_{t+1} \mid (\boldsymbol{P_t}, \boldsymbol{C_t}))$	The probability that a ciphertext $\boldsymbol{C}_{t+1}$ is transmitted by the sender as the next message, given the sequence of current query and response pairs $(\boldsymbol{P_t}, \boldsymbol{C_t})$

3.2 Permutation-only Encryption Scheme

The permutation-only encryption scheme shuffles the bit locations within an expanded text $\boldsymbol{E}$, which is constructed by appending the checksum $\boldsymbol{X}$ to the payload data $\boldsymbol{P}$. The bit permutation process dissipates the statistical structure of the expanded text into long range statistics. To permute the bit locations, a sequence of pseudo-random numbers is constructed by a concatenation of three pseudo-random sequences generated from a linear congruential pseudo-random number generator, defined by the following recurrence relation:

$$y_{j+1} = (h \cdot y_j + q) \mod 2^{32}, \quad (1)$$

where $0 \leq j \leq \lceil \frac{8N+16}{3} \rceil$, $h \bmod 4 = 1$, $q \bmod 2 = 1$, and y_0 is arbitrary. If h and q are selected appropriately, for example, $h \in \{2891336453, 29943829, 32310901\}$ and $q \in \{3, 5, 7\}$, the generated sequences pass formal

Algorithm 1: 16-bit Fletcher checksum.

procedure FLETCHER($\boldsymbol{P}$)
  {Fletcher computes the checksum value $\boldsymbol{X}$, given a payload $\boldsymbol{P}$}
  $\boldsymbol{R} \leftarrow 0,\ \boldsymbol{S} \leftarrow 0$
  for $i \leftarrow 0, N-1$ **do**
    $\boldsymbol{R} \leftarrow (\boldsymbol{R} + \boldsymbol{P_i}) \mod 255$
    $\boldsymbol{S} \leftarrow (\boldsymbol{S} + \boldsymbol{R}) \mod 255$
  end for
  $\boldsymbol{X} \leftarrow (\boldsymbol{S} \ll 8) + \boldsymbol{R}$
  ▷ $\boldsymbol{S} \ll 8$ denotes a logical left shift of $\boldsymbol{S}$ by 8 bits
end procedure

Algorithm 2: Permutation-only encryption.

procedure PERMUTATION($\boldsymbol{E}, K$)
  {Permutation rearranges the bit locations, given an expanded text $\boldsymbol{E}$ and a key $K = (y'_0, y''_0, y'''_0)$}
  for $j \leftarrow 0, \lceil \frac{8N+16}{3} \rceil$ **do**
    $y'_{j+1} \leftarrow \left(h_1 \cdot y'_j + q_1\right) \mod 2^{32}$
    $y''_{j+1} \leftarrow \left(h_2 \cdot y''_j + q_2\right) \mod 2^{32}$
    $y'''_{j+1} \leftarrow \left(h_3 \cdot y'''_j + q_3\right) \mod 2^{32}$
    $y_{3j} \leftarrow y'_{j+1},\ y_{3j+1} \leftarrow y''_{j+1},\ y_{3j+2} \leftarrow y'''_{j+1}$
  end for
  $X \leftarrow sort\ \{y_w\}_{w=0}^{8N+15}$
  for $j \leftarrow 0, 8N+15$ **do**
    $c(j) \leftarrow e(x_j)$
  end for
end procedure

statistical tests (Bellare et al., 1997). To avoid repetition, differing seeds are used to initiate linear congruential generators. The pseudo-random sequence is then sorted in an ascending order, and therefore, a unique index order number is obtained. To complete the permutation, each bit of the expanded text $\boldsymbol{E}$ is relocated according to its corresponding index order. The permuted array forms the ciphertext $\boldsymbol{C}$. More precisely, $\boldsymbol{C} = \{c(j) \mid c(j) = e(x_j),\ \text{for } 0 \leq j \leq 8N+15\}$. The permutation-only encryption scheme is described in Algorithm 2.

3.2.1 Cryptographic Properties

The recurrence relations used in Algorithm 2 are fast, and require minimal memory (32 bits) to preserve the state. This allows the simulation of multiple independent streams. Moreover, there is no repetition in the permutation sequence, because the period of each linear congruential generator 2^{32} is by design much larger than $8N+16$. To further study the statistical properties, the NIST SP800-22 tests (L'Ecuyer and Simard, 2007) were applied to a sequences of 10 million bits generated using the linear congruential generators. The test results are reported in Table 4 using

Algorithm 3: Checksum embedding.

procedure EMBEDDING($\boldsymbol{P}, K$)
  {Embedding embeds a Fletcher checksum and generates a ciphertext $\boldsymbol{C}$, given a plaintext $\boldsymbol{P}$ and a key K}
  $\boldsymbol{X} \leftarrow$ FLETCHER($\boldsymbol{P}$)
  $\boldsymbol{E} \leftarrow [\boldsymbol{P}, \boldsymbol{X}]$
    ▷ $\boldsymbol{P}$ and $\boldsymbol{X}$ are concatenated to form $\boldsymbol{E}$
  $\boldsymbol{C} \leftarrow$ PERMUTATION ($\boldsymbol{E}, K$)
end procedure

Algorithm 4: Integrity verification.

procedure VERIFICATION($\boldsymbol{C}, K$)
  {Verification checks the integrity of a transmitted payload, given a ciphertext $\boldsymbol{C}$ and a key K}
  $\boldsymbol{E} \leftarrow$ PERMUTATION ($\boldsymbol{C}, K$)
  $\boldsymbol{P} \leftarrow [e(0), e(1), \ldots, e(8N-1)]$
  $\boldsymbol{X} \leftarrow [e(8N), e(8N+1), \ldots, e(8N+15)]$
  $\boldsymbol{X'} \leftarrow$ FLETCHER($\boldsymbol{P}$)
  if $\boldsymbol{X} \neq \boldsymbol{X'}$ **then**
    Integrity verification is failed
  end if
end procedure

P-value, which summarizes the strength of the evidence against the randomness (null) hypothesis. For these tests, each P-value is the probability that a perfect random number generator would have produced a sequence less random than the sequence that was tested, given the kind of non-randomness assessed by the test. A P-value less than 0.01 is an indication that the randomness hypothesis is false with a confidence level of 0.99. As shown in Table 4, the sequence passed the test.

The pseudo-random numbers generated by linear congruential generators are known to be predictable in their simplest form (Plumstead, 1983). Even if the parameters y_0, h, q, and b are unknown (used as the secret key), the sequence of numbers produced by a linear congruential generator is still predictable given some of the y_j for j $(0 \leq j \leq 8N+15)$ (Plumstead, 1983). However, this predictability does not imply that a cryptographic algorithm using a linear congruential generator is breakable, because the random numbers used by the permutation algorithm are intermediate values, which are not made public.

3.2.2 Salting Procedure

To avoid using the same permutation mapping, the key is randomized with a salt value. Since GOOSE is a connectionless protocol, it would require out-of-band synchronization for employing key randomization. Such out-of-band synchronization is difficult to achieve in the constrained substation environment. Therefore, to avoid out-of-band synchronization, the

Table 4: NIST SP 800-22 test results.

Tests	*P*-value
Frequency test	0.8263
Frequency test within a block	0.0959
Runs test	0.4726
Test for the longest run of ones in a block	0.5797
Binary matrix rank test	0.7882
Spectral test	0.2366
Non-overlapping template matching test	0.4256
Overlapping template matching test	0.5724
Maurer's "Universal statistical" test	0.8801
Linear complexity test	0.8063
Serial test	0.0802
Approximate entropy test	0.7637
Cusums test	0.6700
Random excursion test	0.2172
Random excursion variant test	0.4014

key is salted with the status number (a 32-bit counter) in the GOOSE message to generate differing permutations in every $\lceil \log_2(8N+16) \rceil - 1$ PMU transmissions, where N is he number of plaintext bytes. This helps to resist certain types of cryptanalysis, such as plaintext attacks and precomputed rainbow table attacks (Oechslin, 2003). If the limit of the status number (that is, 2^{32}) is reached, the GOOSE protocol re-establishes the number (namely roll-over), and the secret key is re-synced. For example, for $N = 128$ bytes, the key is salted every 10 transmissions using a counter, which is 32 bits in most microprocessors. Assuming PMU sampling rate of 60 per seconds, re-keying is needed only every $\frac{2^{32}}{6}$ seconds or 22 years, which is much longer than re-syncing periods chosen for other reasons.

3.3 Embedding and Verification Procedures

The Embedding and Verification algorithms are defined over three sets, namely the plaintext P, the ciphertext C, and the key K, respectively. P is not an arbitrary set, and it follows IEEE C37.118 (Martin et al., 2008). For instance, the data communicated by PMU can never be all 0's or 1's, as the frequency value would never become zero or the time-stamp would never become a negative value. Each key determines a mapping from a set of plaintext into a set of ciphertext, and vice versa. The embedding procedure computes the checksum $\boldsymbol{X}$ of the plaintext $\boldsymbol{P}$; it appends the checksum to the payload data $\boldsymbol{P}$, and then, permutes the expanded data using a fast keyed bit shuffler. More precisely, $\boldsymbol{C} = \Pi_k[\boldsymbol{P}, \boldsymbol{X}]$, where $\boldsymbol{X} = \text{FLETCHER}(\boldsymbol{P})$, and $[\boldsymbol{P}, \boldsymbol{X}]$ denotes the concatenation of $\boldsymbol{P}$ and $\boldsymbol{X}$. The embedding process is described in Algorithm 3.

The content verification procedure is the inverse procedure of checksum embedding. To verify the integrity of transmitted messages, the receiver first decrypts the received message $\boldsymbol{C}$ using a shared key K; recomputes the checksum value $\boldsymbol{X'}$, and then, compares it with the embedded checksum $\boldsymbol{X}$. If the computed and received checksum mismatch, then this shows that there was a message modification. This procedure is detailed in Algorithm 4.

4 SECURITY ANALYSIS

Mathematically, the adversary could be considered as an oracle machine which has access to the sender's embedding function without knowing the key. The adversary asks n number of queries from the sender's embedding function to obtain a set of n plaintext and ciphertext pairs, that is, $\Delta = \{(\boldsymbol{P_t}, \boldsymbol{C_t}) \mid t = 1, 2, \ldots, n\}$. If the adversary successfully breaks the integrity protection algorithm, the location of the checksum is disclosed, and therefore, the adversary can create a forgery that cannot be detected by the verification algorithm of the receiver. The spoofing query is successful if the adversary receives a positive verification response $(\boldsymbol{C}_{t+1}, 1)$ from the receiver. This type of spoofing represents a rather strong adversary, but is realistic in smart grid settings, since the messages could be intercepted and potentially forged. The above discussion suggests that defining security against such a strong adversary is usually not a simple task.

Before we analyze the security of our proposed method, we point out why simpler variants of the same idea are insecure. Suppose that we encrypt the payload data using pseudo-random permutations, that is, $\boldsymbol{C} = \Pi_k(\boldsymbol{P})$. Permutation dissipates the statistical structure of the payload into long range statistics. Choosing a permutation of large length size can exponentially increase the number of possible permutations of payload, that is, $\#(\pi) = (8N)!$. This exponential search space can make the statistical attacks cumbersome by increasing the size of a payload data. However, the encryption of payload bits can only maintain data confidentiality, rather than integrity. As explained in (Krawczyk, 2001), an adversary can simply spoof a new message by modifying the bits in transit.

Another variant is to hide a checksum inside the payload data. This could be achieved by using a steganographic method, for instance, an LSB method (Cogranne and Retraint, 2013). However, this method is not secure from chosen-covertext attacks and chosen-stegotext attacks (Cohen and Lapidoth,

2002). Only one pair of chosen covertext and stegotext is enough to disclose the location of hidden checksum, which is sufficient to bypass the integrity protection algorithm. This problem could be solved by encrypting or hashing the embedded payload (that is, expanded text). However, this solution is not appropriate because of the high latency induced by cryptographic primitives.

Now we analyze the security of the proposed integrity protection method, that is, $\boldsymbol{C} = \Pi_k[\boldsymbol{P}, \boldsymbol{X}]$.

4.1 Ciphertext-only Attack

Theorem 1. *The proposed integrity protection scheme is secure from ciphertext-only attacks.*

Proof. In a ciphertext-only attack, the adversary is able to observe the transmitted ciphertext (that is, the embedded GOOSE message), and then, recalculate the checksum value of the first N bytes in every possible permutation, that is, $(8N+16)!$ checksum recalculations (in worst case). If the value of the recalculated checksum corresponds to the last two bytes of a permutation, then the checksum bytes and their locations will be disclosed. However, this is infeasible for a large N. As discussed in Section 1, the minimum size of a PMU data is 40 bytes, which means the adversary needs to recalculate at least 2.11×10^{664} checksums. This number of computations is extremely large, and thus makes the ciphertext-only attack impractical. A less computationally intensive approach is to simply try to guess the key. However, brute-force attacks are not feasible, because the proposed algorithm employs three 32-bit secret seeds (recall Algorithm 2), which makes the key size more than 80 bits, that is, the minimum key space recommended by NIST (Barker et al., 2016). This requires checking 10^{28} possibilities, which too is infeasible. □

4.2 Known-Plaintext Attack

In the following, we derive an information theoretic bound on the success probability of the query adversary, who spoofs after making t oracle queries, under the assumption that the key is not changed during the queries. To this end, we use the following lemmas for the proof of Theorem 2.

Lemma 1. *Let $(\boldsymbol{P_t}, \boldsymbol{C_t})$ be a pair of query and response with $Pr(\boldsymbol{P_t}, \boldsymbol{C_t}) \neq 0$, let $\boldsymbol{C}_{t+1}$ denote a ciphertext with $Pr(\boldsymbol{C}_{t+1} \mid (\boldsymbol{P_t}, \boldsymbol{C_t})) \neq 0$, and let $\mathcal{K} = K((\boldsymbol{P_t}, \boldsymbol{C_t}), (\boldsymbol{C}_{t+1}, 1))$. Then, $Pr(\boldsymbol{C}_{t+1} | (\boldsymbol{P_t}, \boldsymbol{C_t})) \cdot \log_2(Pr(\boldsymbol{C}_{t+1} | (\boldsymbol{P_t}, \boldsymbol{C_t}))) \leq \sum_{k \in \mathcal{K}} Pr(k, \boldsymbol{C}_{t+1} \mid (\boldsymbol{P_t}, \boldsymbol{C_t})) \cdot \log_2(Pr((\boldsymbol{P_t}, \boldsymbol{C_t}), (\boldsymbol{C}_{t+1}, 1))) \cdot Pr(\boldsymbol{C}_{t+1} | k, (\boldsymbol{P_t}, \boldsymbol{C_t}))$.*

Proof. Let γ denote a function of k. If $k \in \mathcal{K}$, then, $\gamma = 1$; otherwise, $\gamma = 0$. From $Pr(\boldsymbol{C}_{t+1} \mid (\boldsymbol{P_t}, \boldsymbol{C_t})) \neq 0$, it follows that $\mathcal{K} \neq \varnothing$ and $Pr((\boldsymbol{P_t}, \boldsymbol{C_t}), (\boldsymbol{C}_{t+1}, 1)) \neq 0$. Accordingly, we can define a probability distribution $\Psi_{((\boldsymbol{P_t}, \boldsymbol{C_t}), \boldsymbol{C}_{t+1})}$ on k as

$$\Psi_{((\boldsymbol{P_t}, \boldsymbol{C_t}), \boldsymbol{C}_{t+1})}(k) = \frac{Pr(\boldsymbol{C}_{t+1} \mid (\boldsymbol{P_t}, \boldsymbol{C_t})) \cdot \gamma(k)}{Pr((\boldsymbol{P_t}, \boldsymbol{C_t}), (\boldsymbol{C}_{t+1}, 1))}. \tag{2}$$

Since $Pr((\boldsymbol{P_t}, \boldsymbol{C_t}), (\boldsymbol{C}_{t+1}, 1)) = \sum_{k \in \mathcal{K}} Pr(k \mid (\boldsymbol{P_t}, \boldsymbol{C_t})) \cdot \gamma(k)$, then, $\sum_{k \in \mathcal{K}} \Psi_{((\boldsymbol{P_t}, \boldsymbol{C_t}), \boldsymbol{C}_{t+1})}(k) = 1$. Therefore, $\Psi_{((\boldsymbol{P_t}, \boldsymbol{C_t}), \boldsymbol{C}_{t+1})}$ is a probability distribution. If $Pr(\boldsymbol{C}_{t+1} \mid k, (\boldsymbol{P_t}, \boldsymbol{C_t})) \neq 0$, then, $\gamma(k, \boldsymbol{C}_{t+1}, (\boldsymbol{P_t}, \boldsymbol{C_t})) = 1$; thus, we can rewrite the conditional entropy as

$$Pr(\boldsymbol{C}_{t+1} \mid (\boldsymbol{P_t}, \boldsymbol{C_t})) = \sum_{k \in \mathcal{K}} Pr(k | (\boldsymbol{P_t}, \boldsymbol{C_t})) \cdot Pr(\boldsymbol{C}_{t+1} | k, (\boldsymbol{P_t}, \boldsymbol{C_t})) \cdot \gamma(k). \tag{3}$$

By the definition of $\Psi_{((\boldsymbol{P_t}, \boldsymbol{C_t}), \boldsymbol{C}_{t+1})}$, we get

$$Pr(\boldsymbol{C}_{t+1} | (\boldsymbol{P_t}, \boldsymbol{C_t})) = \sum_{k \in \mathcal{K}} \Psi_{((\boldsymbol{P_t}, \boldsymbol{C_t}), \boldsymbol{C}_{t+1})}(k) \cdot Pr((\boldsymbol{P_t}, \boldsymbol{C_t}), (\boldsymbol{C}_{t+1}, 1)) \cdot Pr(\boldsymbol{C}_{t+1} \mid k, (\boldsymbol{P_t}, \boldsymbol{C_t})). \tag{4}$$

Using Jensen's Inequality (Briat, 2011) and Equation 2, we obtain

$$Pr(\boldsymbol{C}_{t+1} \mid (\boldsymbol{P_t}, \boldsymbol{C_t})) \cdot \log_2(Pr(\boldsymbol{C}_{t+1} \mid (\boldsymbol{P_t}, \boldsymbol{C_t}))) \leq \sum_{k \in \mathcal{K}} \Psi_{((\boldsymbol{P_t}, \boldsymbol{C_t}), \boldsymbol{C}_{t+1})}(k) \cdot Pr((\boldsymbol{P_t}, \boldsymbol{C_t}), (\boldsymbol{C}_{t+1}, 1)) \cdot Pr(\boldsymbol{C}_{t+1} \mid k, (\boldsymbol{P_t}, \boldsymbol{C_t})) \cdot \log_2(Pr((\boldsymbol{P_t}, \boldsymbol{C_t}), (\boldsymbol{C}_{t+1}, 1)) \cdot Pr(\boldsymbol{C}_{t+1} \mid k, (\boldsymbol{P_t}, \boldsymbol{C_t}))) = \sum_{k \in \mathcal{K}} Pr(k, \boldsymbol{C}_{t+1} \mid (\boldsymbol{P_t}, \boldsymbol{C_t})) \cdot \log_2(((\boldsymbol{P_t}, \boldsymbol{C_t}), (\boldsymbol{C}_{t+1}, 1)) \cdot Pr(\boldsymbol{C}_{t+1} \mid k, (\boldsymbol{P_t}, \boldsymbol{C_t}))). \tag{5}$$

This proves the lemma. □

Lemma 2. *For $(\boldsymbol{P_t}, \boldsymbol{C_t}) \in \Delta$ with $Pr(\boldsymbol{P_t}, \boldsymbol{C_t}) \neq 0$, $H(\boldsymbol{C}_{t+1} | (\boldsymbol{P_t}, \boldsymbol{C_t})) \geq -\log_2(Pr(\boldsymbol{P_t}, \boldsymbol{C_t})) + H(\boldsymbol{C}_{t+1} \mid K, (\boldsymbol{P_t}, \boldsymbol{C_t}))$.*

Proof. From the definition of $H(\boldsymbol{C}_{t+1} \mid (\boldsymbol{P_t}, \boldsymbol{C_t}))$ and the use of Lemma 1,

$$H(\boldsymbol{C}_{t+1} | (\boldsymbol{P_t}, \boldsymbol{C_t})) \geq -\sum_{\boldsymbol{C}_{t+1} \in \Delta} Pr(\boldsymbol{C}_{t+1} | (\boldsymbol{P_t}, \boldsymbol{C_t})) \cdot \log_2(Pr((\boldsymbol{P_t}, \boldsymbol{C_t}), (\boldsymbol{C}_{t+1}, 1)) - \sum_{\boldsymbol{C}_{t+1} \in \Delta} \sum_{k \in K((\boldsymbol{P_t}, \boldsymbol{C_t}), (\boldsymbol{C}_{t+1}, 1))} Pr(k \mid (\boldsymbol{P_t}, \boldsymbol{C_t})) \cdot Pr(\boldsymbol{C}_{t+1} \mid k, (\boldsymbol{P_t}, \boldsymbol{C_t})) \cdot \log_2(Pr(\boldsymbol{C}_{t+1} \mid k, (\boldsymbol{P_t}, \boldsymbol{C_t}))). \tag{6}$$

This relation is expanded by the definition of $H(\boldsymbol{C}_{t+1}|K,(\boldsymbol{P_t},\boldsymbol{C_t}))$ as

$$H(\boldsymbol{C}_{t+1}|(\boldsymbol{P_t},\boldsymbol{C_t})) \geq -\sum_{\boldsymbol{C}_{t+1}\in\Delta} Pr(\boldsymbol{C}_{t+1}|(\boldsymbol{P_t},\boldsymbol{C_t}))\cdot\log_2(Pr((\boldsymbol{P_t},\boldsymbol{C_t}),(\boldsymbol{C}_{t+1},1)) + H(\boldsymbol{C}_{t+1}\mid K,(\boldsymbol{P_t},\boldsymbol{C_t})). \quad (7)$$

Since $Pr(\boldsymbol{P_t},\boldsymbol{C_t}) \geq Pr((\boldsymbol{P_t},\boldsymbol{C_t}),(\boldsymbol{C}_{t+1},1))$, we have

$$H(\boldsymbol{C}_{t+1}\mid(\boldsymbol{P_t},\boldsymbol{C_t})) \geq -\log_2(Pr_t(\boldsymbol{P_t},\boldsymbol{C_t})) + H(\boldsymbol{C}_{t+1}\mid K,(\boldsymbol{P_t},\boldsymbol{C_t})). \quad (8)$$

This completes the proof. □

We are now in position to prove the following lower bound on the success probability of the query adversary.

Theorem 2. *Given the proposed integrity protection scheme, let Pr_t denote the probability of success of an adversary, who spoofs after making t oracle queries. Then, $Pr_t \geq |K|^{\frac{-1}{t+1}}$.*

Proof. This proof is a direct application of the proof used in (Rosenbaum, 1993) to the situation in which the adversary makes oracle queries rather than observing messages. From the definition of the conditional mutual information, $I(K;\boldsymbol{C}_{t+1}\mid(\boldsymbol{P_t},\boldsymbol{C_t})) = H(K\mid(\boldsymbol{P_t},\boldsymbol{C_t})) - H(K\mid\boldsymbol{C}_{t+1},(\boldsymbol{P_t},\boldsymbol{C_t})) = H(\boldsymbol{C}_{t+1}\mid(\boldsymbol{P_t},\boldsymbol{C_t})) - H(\boldsymbol{C}_{t+1}\mid K,(\boldsymbol{P_t},\boldsymbol{C_t}))$. Hence, it is sufficient to show that $\log_2(Pr_t) \geq H(\boldsymbol{C}_{t+1}\mid K,(\boldsymbol{P_t},\boldsymbol{C_t}) - H(\boldsymbol{C}_{t+1}\mid(\boldsymbol{P_t},\boldsymbol{C_t}))$. By the definition of Pr_t and using Jensen's Inequality (Briat, 2011), we obtain

$$\log_2(Pr_t) = \log_2\left(\sum_{(\boldsymbol{P_t},\boldsymbol{C_t})\in\Delta} Pr(\boldsymbol{P_t},\boldsymbol{C_t})\cdot Pr_t(\boldsymbol{P_t},\boldsymbol{C_t})\right) \geq \sum_{(\boldsymbol{P_t},\boldsymbol{C_t})\in\Delta} Pr(\boldsymbol{P_t},\boldsymbol{C_t})\cdot\log_2(Pr_t(\boldsymbol{P_t},\boldsymbol{C_t})). \quad (9)$$

The lower bound of $\log_2(Pr_t(\boldsymbol{P_t},\boldsymbol{C_t}))$ proved in Lemma 2 yields

$$\sum_{\boldsymbol{C}_{t+1}\in\Delta} Pr(\boldsymbol{P_t},\boldsymbol{C_t})\cdot\log_2(Pr_t(\boldsymbol{P_t},\boldsymbol{C_t})) \geq \sum_{\boldsymbol{C}_{t+1}\in\Delta} Pr(\boldsymbol{P_t},\boldsymbol{C_t})\cdot(H(\boldsymbol{C}_{t+1}|K,(\boldsymbol{P_t},\boldsymbol{C_t})) - H(\boldsymbol{C}_{t+1}|(\boldsymbol{P_t},\boldsymbol{C_t}))) = H(\boldsymbol{C}_{t+1}\mid K,(\boldsymbol{P_t},\boldsymbol{C_t})) - H(\boldsymbol{C}_{t+1}\mid(\boldsymbol{P_t},\boldsymbol{C_t})). \quad (10)$$

Using Inequalities 9 and 10, we obtain

$$Pr_t \geq 2^{H(K|\boldsymbol{C}_{t+1},(\boldsymbol{P_t},\boldsymbol{C_t}))-H(K|(\boldsymbol{P_t},\boldsymbol{C_t}))}. \quad (11)$$

Accordingly, $-\log_2(Pr_{t+1}) \leq -\log_2(Pr_0 \times Pr_1 \times \cdots \times Pr_t) \leq (H(K) - H(K|(\boldsymbol{P_1},\boldsymbol{C_1}))) + (H(K\mid(\boldsymbol{P_1},\boldsymbol{C_1})) - H(K|(\boldsymbol{P_2},\boldsymbol{C_2}))) + \cdots + (H(K|(\boldsymbol{P_t},\boldsymbol{C_t})) - H(K|(\boldsymbol{P}_{t+1},\boldsymbol{C}_{t+1}))) = H(K) - H(K|(\boldsymbol{P}_{t+1},\boldsymbol{C}_{t+1})) \leq H(K)$. Therefore, $\log_2(Pr_t) \geq -\frac{1}{t+1}H(K)$. Since keys are equally likely to be used, and also as the probability of success after the observation of t pairs of plaintext and ciphertext is equal, we have $H(K) = \log_2(|K|)$. Therefore, $\log_2(Pr_t) \geq -\frac{1}{t+1}\log_2(|K|)$. This proves the theorem. □

In the following, we explain a strategy that a query adversary may undertake for known plaintext attacks.

Lemma 3. *Given a permutation-only encryption primitive, which operates on plaintexts with $8N+16$ number of binary entries, the number of required known plaintexts n to perform a successful known-plaintext attack is $O(\lceil\log_2(8N+16)\rceil)$.*

Proof. In a known-plaintext attack, to disclose the permutation mapping, which works on arrays of $8N+16$ bits, it is sufficient to input a plaintext with distinct entries. However, from the practical point of view, constructing a plaintext with distinct entries may not be feasible, because each entry of the plaintext array is from a finite set $\{0,1\}$, and the number of entry locations, that is $8N+16$, exceeds the number of entry values. Therefore, a collection of plaintexts, all of which have repeated entry values, is required to uniquely determine the underlying permutation. This problem is equivalent to splitting an array with distinct entries into a number of arrays whose entry values are equal or less than the maximum number of entry values. To split the array, the adversary needs to expand the entries using n digit expansions in radix 2, where n digits clearly produce 2^n different values. This implies the following relationship for the number $8N+16$ of entry locations:

$$2^n < 8N+16 \leq 2^{n+1}. \quad (12)$$

The inequalities above indicate that the source entries can be expanded by $O(\lceil\log_2(8N+16)\rceil)$ digits, and therefore, the source array can split into $O(\lceil\log_2(8N+16)\rceil)$ plaintexts. In other words, $O(\lceil\log_2(8N+16)\rceil)$ plaintexts construct a source array with distinct entries. □

4.3 Chosen-plaintext Attack

In a chosen-plaintext attack, which is a stronger notion of security compared to a known-plaintext attack, the aim is to find a procedure with a reduced number of required plaintexts.

Lemma 4. *Given a permutation-only encryption primitive, which operates on plaintexts with $8N+16$ number of binary entries, the number of required chosen plaintexts n to perform a successful chosen-plaintext attack is $n = \lceil\log_2(8N+16)\rceil$.*

Proof. Theoretically, the permutation mapping can be easily deduced using a source array of size $8N+16$, whose entries are sequentially labeled with distinct values $0, 1, \ldots, 8N+15$. However, this is not practical, because the encryption/decryption primitives only operate on binary values, which are less than the number of entries. Therefore, to make the attack feasible, the entries are firstly expanded by $\lceil \log_2(8N+16) \rceil$ digits with radix 2. This matrix is then separated into $\lceil \log_2(8N+16) \rceil$ numbers of plaintexts based on the digit positions in radix 2. Once permutation is applied to the plaintexts, it produces $\lceil \log_2(8N+16) \rceil$ ciphertexts with entries in radix 2. A combination of ciphertexts using the positional digits reveals the mapped locations. □

We are now in position to prove the following theorem.

Theorem 3. *The proposed integrity verification scheme is secure from known/chosen plaintext attacks.*

Proof. To falsify the data, the adversary needs to disclose the permutation mapping. To this end, following Lemmas 3 and 4, the adversary asks at least $\lceil \log_2(8N+16) \rceil$ queries from the oracle machine to determine the permutation mapping. These queries are made under the assumption that the same key is used for generating plaintext and ciphertext pairs in each and every queries. However, the keys are salted to a frequency less than the required number of pairs for a successful attack, and are all equally likely to be used. Nevertheless, the adversary tries to impersonate the sender by using only one query. Following Theorem 2, the minimum probability of success for such attacks is $\frac{1}{\sqrt{|K|}}$, which is negligible. In addition, the adversary may try to find the permutation period, or may try to exhaust all possible permutation keys by asking numerous queries. Such an analysis could provide the adversary with a number of plaintext and ciphertext pairs, which had been generated using the same key. However, the permutation domain grows exponentially by increasing the size of payload, and this makes the attack computationally infeasible. For instance, given a payload data with 100 bytes, 9.33×10^{157} queries need to be asked to exhaust all keys. This is computationally infeasible, and makes the proposed scheme secure from known/chosen plaintext attacks. □

4.4 Salt Attack

In a salt attack scenario, the adversary attempts to partially disclose the key by comparing different pairs of plaintext and ciphertext, which were made under small modifications of salt value. Since the status number has linear properties, the proposed scheme seems to be vulnerable to off-path attacks (Gilad, Y. and Herzberg, A., 2014). The adversary can observe the initial messages, and hence, determine the status number. The adversary can then perform a man-in-the-middle attack, and fabricate a new status number for the GOOSE message. To this end, the adversary may increase the status number, because a message with a lower status number than that of the previously received message will not be processed. However, the receiver can easily recognize the fake GOOSE messages through the verification process. Any change in the status number would result in a different permutation mapping, which makes the message verification fail at the receiver side. In addition, knowing the salt value does not help in finding the secret key, because permutation indices are constructed by the numerical order of expanded key stream; and thus, it is not easy to obtain the exact expanded key from permutation indices. Furthermore, the permutation domain is long, and this makes the adversarial analysis difficult to find pairs which were encrypted under the same key.

Another attack strategy is to take control over the status number, before the embedding process, on the receiver's side. To this end, the adversary may give the same status (order) to GOOSE messages. This may provide the adversary with a number of plaintext and ciphertext pairs that are generated under the same encryption key. Under such attack scenarios, the security margin of the proposed scheme would be $\lceil \log_2(8N+16) \rceil$ chosen queries (as demonstrated in Figure 2). However, such physical attacks may be impractical, because the access points of PMUs are physically secured upon proper configuration, which prevents tampering and reprogramming. Nonetheless, the adversary makes further effort to break the scheme by making use of less number of chosen queries (even one). Following Theorem 2, since the permutation domain depends on the size of the plaintext, the minimum success probability of an adversary, who uses only one query for an attack, is $\frac{1}{\sqrt{(8N+16)!}}$, which is negligible. Figure 3 depicts the log scale curve of the minimum success probability of an adversary, who uses at most $\lceil \log_2(8N+16) \rceil - 1$ queries. As shown in Figure 3, this probability is negligible.

5 PERFORMANCE ANALYSIS

In addition to security analysis, the performance of the integrity protection algorithm is also an important factor to consider, especially for real-time GOOSE

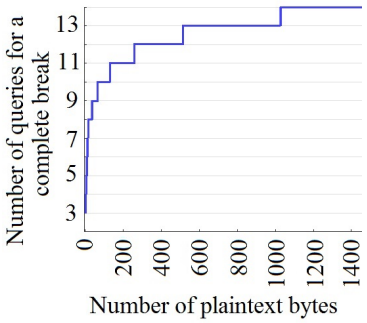

Figure 2: Number of required queries for a complete break.

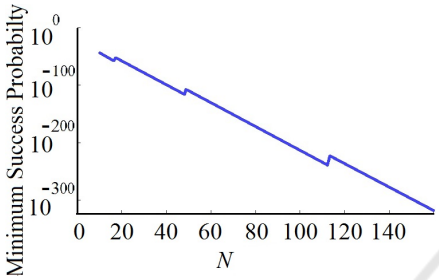

Figure 3: Log scale curve of the minimum success probability of an adversary who uses at most $\lceil \log_2(8N+16) \rceil - 1$ queries.

communications, which require a high level of efficiency. The computational complexity of the integrity protection algorithm is the summation of complexities of the 16-bit Fletcher checksum generation and the bit permutation process. Given an array of length n, the complexity of computing the 16-bit Fletcher checksum is $\alpha \cdot n$, where α is a constant. The computational complexity of the bit permutation process is the sum of complexities of the pseudo-random number generation and the radix sorting procedure. The computational complexity of generating n pseudo-random numbers using a linear congruential generator is $\beta \cdot n$, where β is a constant. In addition, given an array of length n, radix sorting rearranges the array in linear time. Therefore, in the worst case, a permutation of an array with length n is achieved using $\mu \cdot n$ computations, where μ is a constant. Accordingly, the computational complexity of the integrity protection algorithm is $(\alpha + \beta + \mu) \cdot n$.

Given that the maximum size of payload data is 1492 bytes, a feasible integrity protection method should have a minimum throughput of higher than 373 kilobytes per second (KB/s). This allows parties to communicate reliably within the maximum acceptable latency (4 ms). Since the most recent microprocessors in substation automation systems use ARM Cortex-M processor cores, we evaluated the throughput of the integrity protection algorithm on an ARM Cortex-M0 platform with 48 MHz frequency. We used $K = (1,2,3)$ as a representative key for the performance analysis. We also compared our performance results with that of MD5, ChaCha20-Poly1305, AES-128-CCM, AES-128-EAX, and AES-128-GCM on the same platform (Birr-Pixton, 2015; Mouha et al., 2014). The benchmark results on an ARM Cortex-M0 microprocessor are shown in Table 5. Benchmarks show that all methods have the same computational complexity $O(n)$, where n is the number of message bytes. $O(n)$ gives an asymptotic bound $c \cdot n$, which describes a linear growth for a significantly large n. However, since n is bounded, the constant factor c makes a difference in running time. Compared to previous schemes, as confirmed by the run-time analysis shown in Table 5, the proposed integrity protection method is much faster, and it is the only scheme that meets the minimum speed requirement, that is, 373 KB/s. This fast integrity checking opens the use of energy anomaly detection methods when multiple message exchange is required.

Table 5: Algorithm performance comparison.

Algorithm	Speed (KB/s)
Proposed method	424
MD5	147
ChaCha20-Poly1305	94
AES-128-CCM	70
AES-128-EAX	70
AES-128-GCM	41

6 CONCLUSION

In this paper, we proposed a lightweight and secure integrity protection algorithm for maintaining the integrity of PMU data in protective relaying applications. To be more precise, our proposed algorithm is used in the absence of integrity protection in IEC 61850-90-5, when the HMAC identifier is set to zero. The proposed method computes a 16-bit Fletcher checksum of the payload data, embeds it into the payload data, and then, shuffles the payload bits using a fast permutation-only encryption scheme. We analyzed the security of our method with respect to a query adversary. Our analysis showed no weaknesses in the proposed method, and demonstrated no simple method of recovering the secret key. It also confirmed the security of the proposed integrity protection method against ciphertext-only attacks and known/chosen plaintext attacks. A comparison with a number of existing integrity protection methods showed that despite having the same level of computational complexity, the proposed method is

much faster, and it is the only integrity protection scheme that meets the speed requirement.

Since the proposed integrity protection method uses a symmetric encryption algorithm, its security highly depends on the initial key agreement, rekeying, and revocation. To this end, our future research will be focused on secure and efficient key management between PMUs and PDCs in transmission and distribution substations.

REFERENCES

Abuadbba, A. and Khalil, I. (2015). Wavelet based steganographic technique to protect household confidential information and seal the transmitted smart grid readings. *Information Systems*, 53:224–236.

Abuadbba, A., Khalil, I., Ibaida, A., and Atiquzzaman, M. (2016). Resilient to shared spectrum noise scheme for protecting cognitive radio smart grid readings- BCH based steganographic approach. *Ad Hoc Networks*, 41:30–46.

Adamiak, M., Schiefen, M. J., Schauerman, G., and Cable, B. (2014). Design of a priority-based load shed scheme and operation tests. *IEEE Transactions on Industry Applications*, 50(1):182–187.

Barker, E., Barker, W., Burr, W., Polk, W., and Smid, M. (2016). Recommendation for key management part 1: General (revision 4). *NIST special publication*, 800(57).

Bellare, M., Goldwasser, S., and Micciancio, D. (1997). "pseudo-random" number generation within cryptographic algorithms: The DDS case. In *Annual International Cryptology Conference*, pages 277–291. Springer.

Birr-Pixton, J. (2015). Benchmarking modern authenticated encryption on €1 devices. https://jbp.io/2015/06/01/modern-authenticated-encryption-for-1-euro/.

Briat, C. (2011). Convergence and equivalence results for the Jensen's inequality–Application to time-delay and sampled-data systems. *IEEE Transactions on Automatic Control*, 56(7):1660–1665.

Cogranne, R. and Retraint, F. (2013). An asymptotically uniformly most powerful test for LSB matching detection. *IEEE transactions on information forensics and security*, 8(3):464–476.

Cohen, A. S. and Lapidoth, A. (2002). The gaussian watermarking game. *IEEE Transactions on Information Theory*, 48(6):1639–1667.

Dworkin, M. J. (2007). *Recommendation for block cipher modes of operation: The CMAC mode for authentication*. NIST special publication 800-38b, National Institute of Standards and Technology (NIST).

Fan, Y., Zhang, Z., Trinkle, M., Dimitrovski, A. D., Song, J. B., and Li, H. (2015). A cross-layer defense mechanism against GPS spoofing attacks on PMUs in smart grids. *IEEE Transactions on Smart Grid*, 6(6):2659–2668.

Gilad, Y. and Herzberg, A. (2014). Off-path TCP injection attacks. *ACM Transactions on Information and System Security (TISSEC)*, 16(4):13.

Guo, H., Li, Y., and Jajodia, S. (2007). Chaining watermarks for detecting malicious modifications to streaming data. *Information Sciences*, 177(1):281–298.

IEC 61850-90-5 (2012). Use of IEC 61850 to transmit synchrophasor information according to IEEE C37. 118.

Khan, E., El-Kharashi, M. W., Gebali, F., and Abd-El-Barr, M. (2007). Design and performance analysis of a unified, reconfigurable HMAC-Hash unit. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 54(12):2683–2695.

Krawczyk, H. (2001). The order of encryption and authentication for protecting communications (or: How secure is SSL?). In *Annual International Cryptology Conference*, pages 310–331. Springer.

Kundu, P. and Pradhan, A. K. (2014). Synchrophasor-assisted zone 3 operation. *IEEE Transactions on Power Delivery*, 29(2):660–667.

L'Ecuyer, P. and Simard, R. (2007). TestU01: A C library for empirical testing of random number generators. *ACM Transactions on Mathematical Software (TOMS)*, 33(4):22.

Martin, K., Hamai, D., Adamiak, M., Anderson, S., Begovic, M., Benmouyal, G., Brunello, G., Burger, J., Cai, J., Dickerson, B., et al. (2008). Exploring the IEEE standard C37. 118–2005 synchrophasors for power systems. *IEEE transactions on power delivery*, 23(4):1805–1811.

Martin, K., Zwergel, A., and Kapostasy, D. (2015). PMU installation and configuration requirements, Public Manual, version no. 1.0, MISO energy. https://www.misoenergy.org/Library/Repository/.

Mouha, N., Mennink, B., Van Herrewege, A., Watanabe, D., Preneel, B., and Verbauwhede, I. (2014). Chaskey: an efficient MAC algorithm for 32-bit microcontrollers. In *International Workshop on Selected Areas in Cryptography*, pages 306–323. Springer.

Nakassis, A. (1988). Fletcher's error detection algorithm: how to implement it efficiently and how to avoid the most common pitfalls. *ACM SIGCOMM Computer Communication Review*, 18(5):63–88.

Oechslin, P. (2003). Making a faster cryptanalytic time-memory trade-off. In *Annual International Cryptology Conference*, pages 617–630. Springer.

Pappu, V., Carvalho, M., and Pardalos, P. (2013). Optimization and security challenges in smart power grids. page 157. Springer.

Plumstead, J. B. (1983). Inferring a sequence generated by a linear congruence. In *Advances in Cryptology–CRYPTO*, pages 317–319.

Rosenbaum, U. (1993). A lower bound on authentication after having observed a sequence of messages. *Journal of Cryptology*, 6(3):135–156.

Seyed Reza, F., Vanfretti, L., Ruiz-Alvarez, A., Mahmood, F., Hooshyar, H., and Cairo, I. (2016). An IEC 61850-90-5 gateway for IEEE C38. 118.2 synchrophasor data transfer. In *IEEE PES General Meeting, 17th to 21th*

of July, 2016. Institute of Electrical and Electronics Engineers (IEEE).

Sobolewski, J. S. (2003). Cyclic redundancy check. In *Encyclopedia of Computer Science*, pages 476–479. John Wiley and Sons Ltd.

Sofia, A., Lewis, C., Jacobi, C., Jamsek, D. A., Riedy, D. F., Vogt, J.-S., Sutton, P., and John, R. S. (2015). Integrated high-performance data compression in the IBM z13. *IBM Journal of Research and Development*, 59(4/5):7–1.

Weng, Y., Negi, R., Faloutsos, C., and Ilić, M. D. (2016). Robust data-driven state estimation for smart grid. *IEEE Transactions on Smart Grid*, pages 1–12.

Wetterwald P. and Raymond J. (2015). Deterministic networking utilities requirements, IETF draft, detnet.

Zander, S., Armitage, G., and Branch, P. (2007). A survey of covert channels and countermeasures in computer network protocols. *IEEE Communications Surveys & Tutorials*, 9(3):44–57.

Zhang, W., Liu, Y., Das, S. K., and De, P. (2008). Secure data aggregation in wireless sensor networks: a watermark based authentication supportive approach. *Pervasive and Mobile Computing*, 4(5):658–680.