Securing the Cloud-Assisted Smart Grid

Kubilay Demir[1], Hatem Ismail, Tsvetoslava Vateva-Gurova, and Neeraj Suri

Dept of CS, TU Darmstadt, Germany

Abstract

Rapid elasticity, ubiquitous network access, and highly-reliable services are some of the desirable features of cloud computing that are attractive for building cloud-assisted data-intensive Smart Grid (SG) applications. However, the Distributed Denial-of-Service (DDoS) attacks represent a serious threat to the cloud-assisted SG applications. To mitigate the risk related to the DDoS threat, we propose an SG-relevant Hierarchical Hybrid Cloud-Extension Concept (HHCEC) along with a DDoS attack defense mechanism, termed as Port Hopping Spread Spectrum (PHSS). HHCEC is a cloud-assisted architecture designed to meet scalability and security requirements of the SG applications in the cloud. To prevent transport or application-layer DDoS attacks on HHCEC, PHSS switches the open port of server as a function of time and a secret shared between authorized clients and server, and thus efficiently dropping packets with closed port number. In addition, PHSS spreads the data packets over all the servers versus a single server to provide a robust protection against volume-based DDoS attacks that would affect some of the servers. This packet spreading approach enables PHSS to instantiate replica servers to take over the attacked servers without blocking the whole traffic by utilizing the rapid-elasticity characteristic of the cloud. Moreover, PHSS leverages a *shuffling-based containment mechanism* in order to quarantine malicious clients in a notably short time. Accordingly, the effect of a DDoS attack based on the compromised secret of the malicious clients is minimized. We evaluate our approach by building a proof-of-concept prototype using Amazon's EC2 and the PlanetLab test-bed. In a DDoS attack scenario, the proposed approach obtains a significant availability enhancement of >38% that highlight its efficiency in comparison to existing approaches. The results also indicate negligible overhead for the proposed approach compared to the plain system i.e., no additional latency and less than 0.01% throughput degradation .

Keywords: Availability, Security, Cloud, DDoS attack, Smart Grid

1. Introduction

The Smart Grid (SG) is a cyber-physical system linking communication, computation and control functions across the SG services to enable distributed generation on the power grid. To manage millions of SG devices and to handle large amounts of data in a reliable, scalable, and cost-effective way, the SG utilities increasingly extend their communication-based management system to the advocated cloud computing platforms for enabling reliable and on-demand access to varied computing resources [1, 2]. Despite the advantages of the cloud, its usage of the public network and shared resources can expose the SG to security risks considering both the cyber and physical systems, e.g., power grid/appliances. In particular, DDoS attacks represent a major threat to the SG applications running in the cloud, considering SG applications' stringent latency requirements (in the range of 100 ms to 5 s) and reliability requirements (99.00 %–99.99%) [1].

As availability constitutes a safety property for SG applications (especially for control functions), deploying proactive defense mechanisms becomes indispensable for SG communication. Proactive defense mechanisms, e.g., moving/hiding the target [3, 4, 5, 6], are introduced as countermeasures increasing the cost on the attacker to overwhelm the victim's resource. However, since these proactive defense mechanisms are mainly designed to mitigate DDoS attacks in typical web applications, they are not suitable for the SG applications'

[1]Email Adresses: {kubidem, hayman, vateva, suri}@cs.tu-darmstadt.de,
Corresponding Author: Kubilay Demir,
Research supported in part by EC H2020 CIPSEC GA #700378.
Dept of CS, TU Darmstadt Hochschulstr. 10, 64289 Darmstadt, Germany Phone: +49-6151-16-25225 Fax: +49-6151-16-25230

context due to the SG specific requirements of high availability and responsiveness [7].

Contributions

To fill this gap, we propose a hybrid hierarchical cloud-extension concept (HHCEC), which is a SG-relevant cloud-assisted architecture. HHCEC provides high responsiveness and security with its (a) hybrid and geographically dispersed structure, and (b) specialized broker-based publish-subscribe communication system. Second, we propose a novel approach termed Port Hopping Spread Spectrum (PHSS), which acts as a strong defense mechanism against transport and application layer DDoS attacks, as well as the high-volume DoS/DDoS attacks, against the broker servers. PHSS is equipped with two distinctive features: (1) *port hopping*, changing the open port of the broker server as a function of the time and a secret shared between the broker server and the publishers[2], and (2) *packet spreading*, diffusing consecutive data packets over a number of broker servers versus a single broker server. This approach enables PHSS to instantiate replica broker servers to take over the attacked broker servers without blocking the whole traffic by taking advantage of the rapid-elasticity characteristic of the cloud.

The existing *port hopping* approaches assume that the secret (a cryptographic seed), if compromised, can be renewed by an Authorization Server using a public key-based rekeying approach. However, this approach increases the computational complexity, and is thus not practical for different SG devices (cf., [8, 4]). Moreover, as the secret is compromised, the adversary can mount an DDoS attack on the open ports and render the broker inaccessible during the long rekeying time of the public key-based approach. In such cases, the broker server becomes unavailable during the re-keying process for all publishers, which in turn severely impacts the SG applications' service provision. Accordingly, to minimize the impact of DDoS attacks against the open ports of broker servers as a result of compromising the secret, we introduce (1) *a token-based authentication mechanism* that allows for a light-weight periodic transmission of the secret to each client (publisher), and (2) *a shuffling-based containment mechanism* that quarantines *malicious clients*, without rendering the attacked broker server inaccessible. To do this, the containment mechanism repositions/shuffles the clients over the ports of the broker server with a negligible overhead.

To assess the efficiency of the proposed approach, we construct a proof-of-concept prototype using EC2-micro instance [9] and PlanetLab (http://planet-lab.org) test-bed. We evaluate PHSS's effectiveness in providing network availability by using the *shuffling-based containment mechanism* against DDoS attacks using the compromised secret. Availability in this paper refers to the success rate of delivery of the messages in predefined time interval through the network. We also compare our approach with the public key-based rekeying method used by the existing *port hopping* mechanisms. Our results show that by containing the impact of the DDoS attack using the compromised secret in a notably shorter time period, PHSS provides high network availability of over 98% during the attack versus the typical ~60% availability achieved by using the public key-based rekeying method. Furthermore, after assessing the overhead (in terms of broker server throughput and response latency), the experimental results show that our proposed mechanism causes neither significant throughput degradation (i.e., <0.01% throughput degradation), nor additional latency compared to the system without our mechanism. To summarize, our contributions are:

- A SG-relevant cloud extension, termed HHCEC, which utilizes a hybrid and geographically dispersed structure to meet the responsiveness and reliability requirements of SG applications.
- A strong proactive DDoS attack defense mechanism, called PHSS, which dynamically changes the open ports of the broker servers to efficiently drop the invalid packets in the firewall. Furthermore, PHSS diffuses consecutive data packets over a number of servers versus a single server to rapidly recover the attacked system in the cloud.
- A token-based authentication mechanism to impede secrets compromise, as well as *a shuffling-based containment mechanism* to contain the damage of the DDoS attack utilizing the compromised secret in a shorter time.
- The proposed system can also be easily adapted to all mission and safety critical applications requiring high availability and low latency in the use of public network and cloud.
- A proof-of-concept platform using Amazon's EC2 [9] and PlanetLab nodes to evaluate our approach

[2]The terms client/publisher and server/broker are interchangeably used in the rest of the paper. In addition, while every SG device/application server can be publisher and/or subscriber, the brokers are dedicated servers for their respective roles.

in terms of the availability of service provision for the SG applications over DDoS attacks and the overhead imposed by our approach.

The remainder of the paper is organized as follows: Section 2 details the system model and problem statement. Section 3 introduces the HHCEC, followed by the PHSS approach in Section 4 and their evaluation in Section 5. We present the related work in Section 6. Section 7 concludes the paper.

2. System Model, Problem Statement and Assumptions

We now describe the system model in addition to the problem statement and assumptions driving our approach.

2.1. System model

We consider the established SG model where the utility uses a heterogeneous network (i.e., public and private) and a hybrid hierarchical cloud infrastructure (HHCEC), taking into account the availability requirements of SG applications and the cost-effectiveness. HHCEC is detailed in Section 3 and illustrated in Fig. 1. As publish-subscribe (pub-sub) systems inherently provide scalability and proactive DDoS attack defense for the constrained SG devices, we employ a broker-based pub-sub system on HHCEC.

A system administrator, which considers the geographical distance and the latency between the brokers and publishers, assigns each publisher to a broker bundle. Furthermore, the system administrator monitors/maintains the latency between the broker bundles and the publishers to re-assign the publishers to a new broker bundles in case of detecting intolerable latency.

To mitigate DDoS/DoS attacks that target the traffic of SG applications running on HHCEC, we develop a defense mechanism, termed PHSS, which is discussed in Section 4. PHSS distinguishes between authorized and unauthorized traffic before it reaches the resource-constrained SG devices, thus countering the DDoS attacks in the well-provisioned broker servers in terms of computation capacity and bandwidth. To filter the unauthorized traffic with minimal cost in the broker servers, we use the port hopping mechanism, which changes the open port numbers of the broker server as a function of time and a secret known by the broker server and all publishers. Thus, the broker servers are resilient against application and transport-layer DDoS attacks with minimal cost. However, in the port hopping approach, disclosure of the shared secret allows DoS/DDoS attacks against the open ports. To minimize the effect of such attack, we develop a port-shuffling-based containment mechanism, which quarantine the compromised client(s) and deliver a new secret to innocent clients.

2.2. Problem statement

Objective of the proposed pub-sub system is to guarantee secure transmission of the published data to the corresponding subscribers within the time window specified in the application requirements. To intercept the data transmission, an attacker should overwhelm the resource of one of the following devices: publishers, intermediary underlay routers, broker servers, or subscribers inaccessible.

Note that the IP addresses of publishers and subscribers are not public. In addition, publishers do not use any channel to receive data, while subscribers are allowed to receive data only from predefined IP addresses. Therefore, we do not expect a direct DDoS attack against the publishers and subscribers. An attack against the backbone routers is also out of the scope of this paper. However, since the IP addresses of the broker servers are public, they are vulnerable to DoS/DDoS attacks. Therefore, we focus on developing a defense mechanism for the broker servers against DoS/DDoS attacks.

Moreover, as our approach employs a port hopping mechanism that uses a secret shared with all publishers, the broker servers can be brought down using low-rate DDoS attack once the shared secret is compromised by an attacker. The existing port hopping based DDoS mitigation approaches [8, 10, 4] assume that the compromised secret can be renewed by delivering a new secret to all publishers using the public-key infrastructure. However, during the long rekeying time of the public-key based containment, the broker server might be inaccessible. Since SG applications have strict latency requirement (i.e., $< 1s$), the delayed measurement due to the inaccessibility might result in safety risks for the power grid. To mitigate those risks, any DDoS attack that exploits the compromised secret must be eliminated by containing the impact of the DDoS attack in a reasonable time period. Therefore, we focus on developing a containment method that quarantine the compromised client to mitigate the DDoS attack in notably shorter time.

We consider a strong threat model where the attacker:

- controls a minority of publishers/clients that behave maliciously, referred to as *malicious clients*.

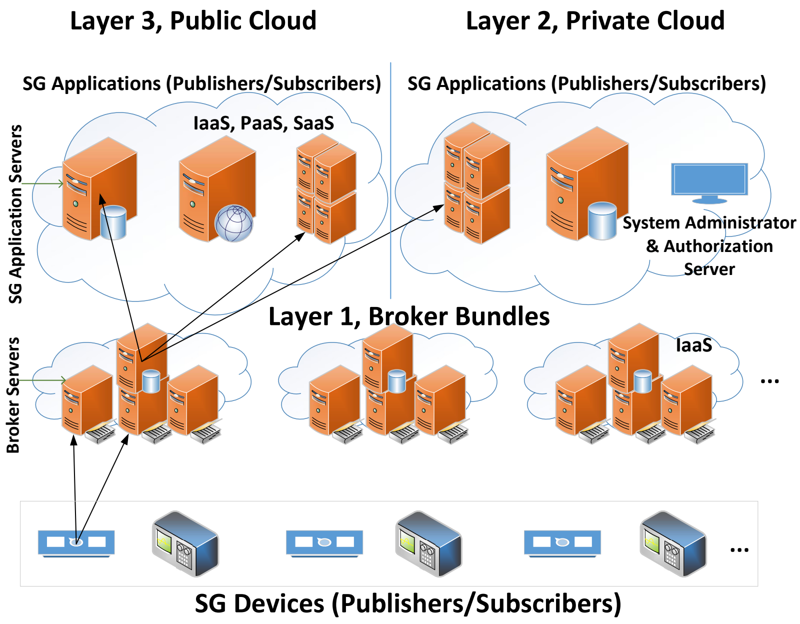

Figure 1: Hybrid Hierarchical Cloud Concept (HHCEC)

- can eavesdrop, capture, drop, resend, and alter some of the traffic between the publisher and the brokers to launch DDoS attacks against brokers.
- can disclose the secret of the *malicious clients*. Accordingly, the attacker can launch a DDoS attack against the open port of the broker server.

2.3. Assumptions

As in contemporary attack models, we assume that (a) publishers obtain only the IP addresses of the broker servers and (b) valid certificates are issued by a Certification Authority to all brokers/publishers/subscribers[3] and to Authorization Servers in a secure way. Since we focus on the broker defense against DDoS attacks, the protection of the Authorization Server is beyond the scope of this paper.

It is worth mentioning that the pathological case of attackers that can fully saturate the Internet backbone links for HHCEC is beyond the scope of this approach.

3. Cloud Computing for Smart Grid

In this section, we motivate the utility of the cloud for SG applications. Afterwards, we highlight the existing limitations behind the direct usage of the cloud structure in the SG context. Finally, we describe the technical details behind our proposed cloud-assisted architecture that addresses such limitations. We also present existing approaches related to the adoption of cloud computing for the SG in Section 6.2.

Typically, the realization of smart grids causes a very large increase in data volume due to the implementation of real time metering, monitoring and pricing applications. This massive data needs to collect and process in real time. As control decisions are solely based on such data, they significantly affects the stability and reliability of the SG. Thus, data parallelism and high computational capabilities play key roles in analyzing and processing this large amount of data [1].

However, the variable resource needs of the SG applications, as matching the varying SG operational behavior, is a challenge for the SG utilities. These applications operate in idle mode on dedicated hardware until a particular situation occurs, e.g, detected abnormality in the grid voltage. This results in inefficient resource usage. Consequently, using a cloud computing platform becomes a viable solution to address these issues due to its featured rapid elasticity [1]. In fact, as the SG applications have strict availability, response time and security requirements, the direct usage of the cloud for the SG encounters the following limitations [1].

1. *Guaranteed Service Availability*: while availabil-

[3] We suppose that our approach is deployed on SG devices that possess enough resources for asymmetric-key cryptography

ity, real-time responsiveness, guaranteed consistency, and fault tolerance are the properties indirectly affecting safety of the SG, they are typically liveness properties for cloud service providers. Avoiding single point of failure scenarios and potential communication bottlenecks is a must to achieve high availability in the use of the typical cloud for the SG.

2. *High Responsiveness*: for data efficiency in the Cloud, an outer layer of the Cloud can be built to provide data aggregation and multiplexing towards the main applications. This would eliminate the potential data transfer bottleneck and contribute to the responsiveness of the applications.
3. *Data Confidentiality*: some SG applications require high confidentiality to prevent data sharing or information leakage, which the cloud service providers typically do not provide. On the other hand, some SG applications need relatively less security protection. This security diversity forces the SG utility to employ diverse resources with different security assurances in the cloud adoption.

In the next section, we introduce an SG related cloud-extension concept that overcomes the above-mentioned limitations resulting from the direct usage of the cloud in the SG context.

3.1. Hybrid hierarchical cloud concept (HHCEC) for the SG

Providing the specific SG requirements is the driver behind proposing a 3-layer HHCEC cloud-assisted architecture, as depicted in Fig. 1. *The first layer* is composed of *Broker Bundles*, which are dispersed based on the grid topology throughout the utility territory. Each *Broker Bundle* can consist of several broker servers. The goal of the *Broker Bundles* is to handle the time-sensitive data in a location surrounding the source rather than in a remote center. This layer provides an interface to support data concentration, data pre-processing, short-term redundant data storage (using replica shards), proactive defense against DoS/DDoS attacks and multiplexing for applications running in the other layers. Since this layer is composed of public cloud infrastructures, data requiring high privacy is either anonymized or encrypted in the publishers so that it can be decrypted solely by the destination [6].

The second layer is an in-house private cloud infrastructure comprised of application servers that process data requiring high availability and/or confidentiality. This layer controls and monitors the *Broker Bundles* of the first layer and assigns the SG devices to the corresponding *Broker Bundles*. Furthermore, the second layer accommodates applications performing analysis, batch processing, permanent archiving, and visualization functions.

Applications/data requiring less security are delegated to *the third layer*, which consists of public cloud infrastructure(s). This layer communicates and shares corresponding data with third parties.

While the public clouds in the first layer are built using the infrastructure as a service (IaaS) model, the public clouds in the third layer can be constructed using IaaS, platform as a service (PaaS), and/or software as a service (SaaS) models depending on the applications' requirements. On the flip side, the private cloud in the second layer is located in-house to strictly ensure no physical data access by third-party.

We utilize a pub-sub system as a communication platform on HHCEC for SG applications. The brokers of this pub-sub system reside in the *Broker Bundles*. The communication between the SG devices and the layers 2 and 3 is not direct, but goes through the *Broker Bundles*, as shown in Fig. 1. The application servers and the SG devices can be either publishers or subscribers. We assume that their roles are assigned by a system administrator residing in the in-house cloud architecture provided by the second layer.

As a summary, the proposed cloud-assisted architecture HHCEC accommodates the pub-sub based SG communication platform while taking into account the SG security requirements. We next describe the proposed DDoS attack defense mechanism, PHSS, that guards the broker servers residing in the *Broker Bundles*.

4. Port Hopping Spread Spectrum (PHSS)

In this section we detail the technical concepts behind our proposed defensive mechanism required for securing the aforementioned cloud-assisted SG structure. The proposed PHSS constitutes of two main mechanisms: (1) *port hopping* and (2) *packet spreading*, which provide for a robust DDoS protection for the pub-sub broker servers.

4.1. Port hopping

The *port hopping* system of PHSS periodically changes the open port of the broker server over time, as illustrated in Fig. 2, according to a pseudo-random sequence known by both the clients and broker server. This sequence is produced by the broker and the clients

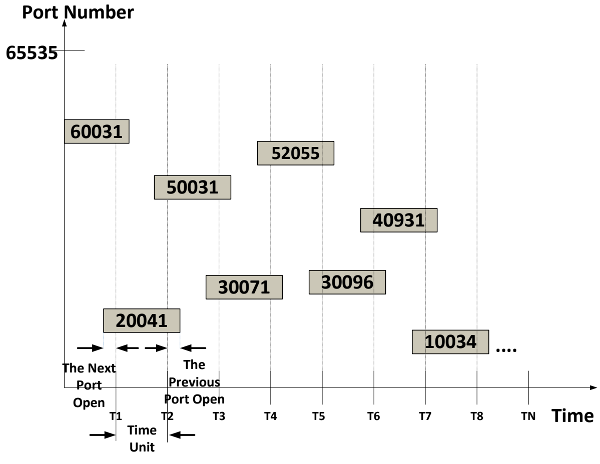

Figure 2: Port Hopping Approach

using a shared secret, the time and a pseudo random function (PRF). In addition, to avoid clients sending packets to the previous or the next port due to time sync error or communication latency, the broker server leaves the previous or the next ports open for a certain time period in the time period of the current port, corresponding to the maximum latency between the broker and the clients [4] (see Fig. 2). In this context, two challenges must be considered: (1) time synchronization attacks or clock drift [4] and (2) compromising of the shared secret by the attacker.

4.1.1. Time synchronization attacks/clock drift

To address the first challenge, PHSS takes advantage of a secure synchronization approach between the brokers and clients. To perform the secure synchronization, each client first obtains a respective session key (128 bits symmetric key) and an authentication ticket (which also includes the session key) from an Authorization Server via a secure channel during the process of joining the network (see messages # 1 and # 2 in Fig. 3). The authentication tickets (akin to Kerberos ticket [11]) are encrypted and signed using a shared key[4] known by the broker servers. The session key of a given client is derived by decrypting the authentication ticket (inside the sync-request message of the client) by using the shared key in the broker servers. Thus, the syn-request messages integrity is checked using the session key by the broker servers.

To synchronize the secret and time, each client sends a sync-request message to the broker including the respective authentication ticket and time-stamp. As a response to this, a sync-reply message, including the current secret, the life-time of the secret and a time-stamp, is issued by the broker server. The sync-reply messages are issued to each client by encrypting and signing with the respective session key. This synchronization process is illustrated in Fig. 3 (3. and 4. messages).

A client receiving the sync-reply message can synchronize the time with the broker server, as reported in [4]. The life-time of the secret is randomly generated to avoid synchronization attacks. Before the end of the life-time of the current secret, each client issues a new sync-request message to the broker server to derive a new secret and time-sync info[5]. The regular re-synchronization employed by our approach provides protection against clock drift and time synchronization attacks, which are the main concerns of the existing *port hopping* approaches [8, 4].

4.1.2. Shared secret compromise by the attacker

Another concern associated with the second challenge, is the compromise of the secret shared among all clients, which poses a high security threat for the system. In such a case, the malicious client spreads the secret to the botnet to launch a DDoS attack against the open ports. Since the open port numbers are a function of the secret and time, the attacker can easily discover and target the ports by using the botnet. The existing *port hopping* approaches use a PRF and a long-term clients secret, which increases the risk of compromise of the secret [8, 4]. As a consequence of compromising the secret, SG applications would experience an unacceptable degradation of availability till new secrets are issued to all clients via the secure channel (using public key). To address this issue, in PHSS, each client regularly requests the current secret from the broker server, as mentioned above.

The regular renewal of the secret by using the token-based authentication provides a limited mitigation since the attacker can continuously compromise the clients' secrets and thus, launch a direct DDoS attack against the open port. In PHSS, to effectively contain the damage of the attack on the broker server and to meet the availability requirements of SG applications during the DDoS attacks, we develop *shuffling-based containment mechanism*. This mechanism, in a short period of time, quarantines *malicious clients* in addition to renewing the secret for the innocent clients.

[4] A symmetric key.

[5] The synchronization is fulfilled a few times in a day by each client. The overhead of this process is negligible in comparison to the daily traffic of client/broker server.

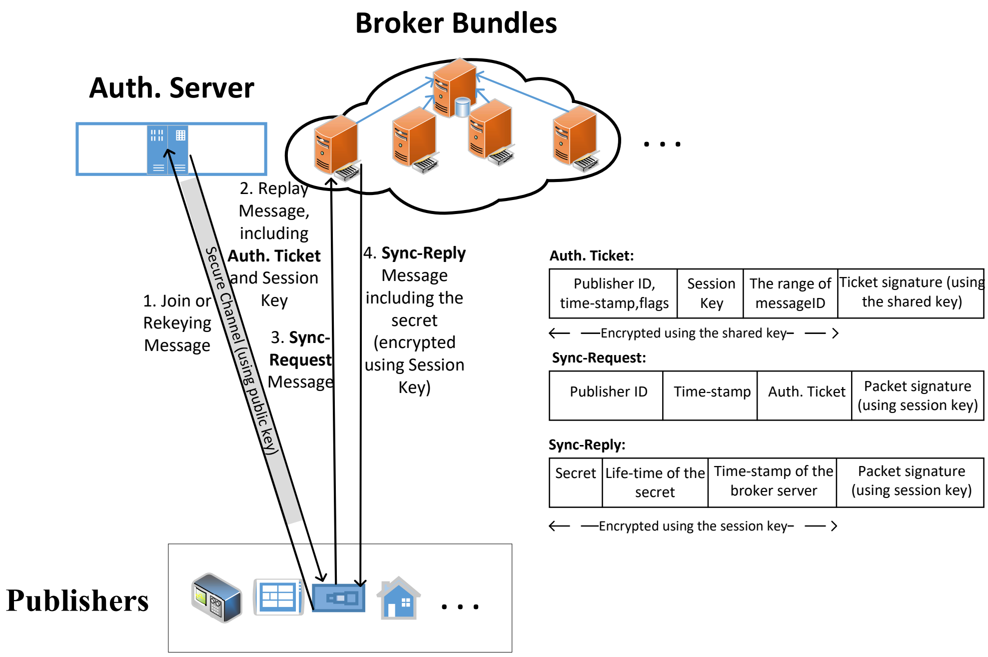

Figure 3: Authentication and synchronization protocol

Shuffling-based containment mechanism. We develop a *shuffling-based (repositioning) containment mechanism*, which contains the impact of *malicious clients* by localizing/quarantining them and then renewing their keys via Authorization Server, as illustrated in Fig. 4. The shuffling idea is roughly inspired by [5], but our mechanism does not require moving target servers and additional servers, unlike [5]. In the *shuffling-based containment mechanism*, when the broker server detects the DDoS attack on the open port[6], it randomly shuffles and consequently splits all clients N into p clusters by considering that all clients are suspicious clients N_s, ($N_s = N$). New secrets[7] are then transmitted to each of the p clusters. This process is simply called a *shuffling* iteration. After the clients start using their new secrets, the port(s) under attack indicate that the corresponding secret(s) are compromised. The clients who do not use these compromised secrets are removed from N_s[8]. Then, the clients of N_s are shuffled and re-clustered by issuing new secrets for each new cluster. This technique progressively quarantines the *malicious clients*, which provides a quick localization of the *malicious clients* c without disturbing the whole traffic. The number of *shuffling* iterations is denoted as x. Also, an overview of the variables and constants used in the shuffling-based process is given in Table I.

To investigate the effects of p and c on the number of *shuffling* iteration x (indicating also the containment duration), we perform a mathematical analysis as follows:

Lemma. *For a fixed N, if $|N|/(p/c)^x \leq 1$, then the compromised clients c are localized in x shuffling iterations by splitting the N_s into p clusters in each shuffling iteration.*

Proof. To localize a *malicious client* in x *shuffling* iterations, first, N_s is set equal to N ($N_s = N$) and then it is split into p clusters (p is equal to $|N|^{\frac{1}{x}}$). The broker server issues a different secret for each cluster. Af-

[6]To detect the attack we simply probe the port periodically, but more complicated methods can be used for the detection like [12].

[7]For each secret, the broker server concurrently opens the corresponding ports. A client using a given secret communicates over the port opened for that secret

[8]The benign clients can continue the transmission over the last issued secrets/ports without disturbing their traffic.

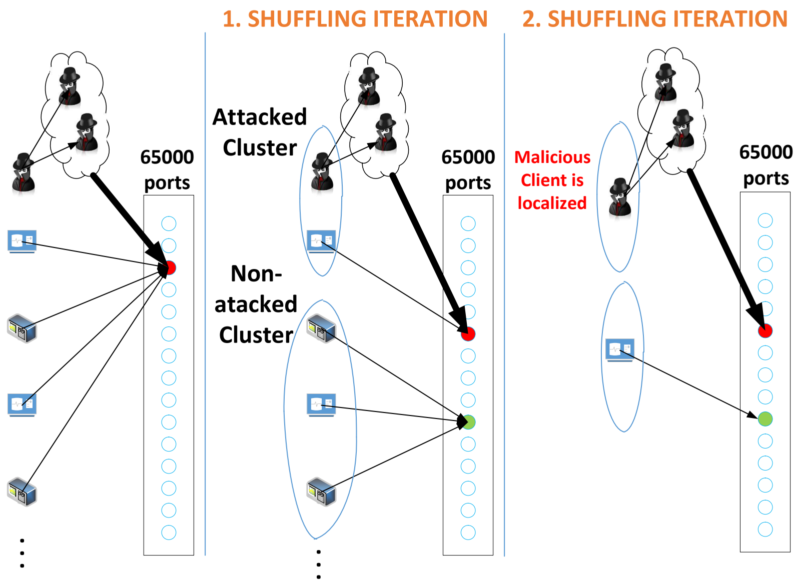

Figure 4: Port *Shuffling*

Table 1: Variables and Constants Definition.

Symbol	Definition
N	The set of clients
N_s	The set of suspicious clients
p	The number of clusters/secrets/open ports
x	The number of *shuffling* iterations
c	The number of *malicious clients*
S_a	The set of secrets used by attacked ports

ter the first *shuffling* iteration, the clients of the cluster(s) whose secret(s) are not used to launch an attack on the corresponding port(s) are removed from N_s. This *shuffling* iteration continues for N_s until a different port is assigned to each suspicious client ($|N_s| \leq p$), which enables to localize the *malicious client*. In addition, if $c \geq 1$, N_s is further split into p clusters in each clustering/*shuffling* iteration, and p is assigned to ($p = |N|^{\frac{1}{x}} * c$).

A speedy localization of the *malicious client(s)* minimizes the loss of network availability. To this end, in the extreme case, we can assign each client to a different cluster, namely issuing a different secret per client ($p = |N|$), and thus finding the malicious one after a *shuffling* iteration ($x = 1$) based on the above lemma. However, opening a large number of ports poses a high risk of being vulnerable to attacks that target the entire port range. In addition, building larger clusters in each *shuffling* iteration, e.g., splitting into two clusters ($p = 2$) in each *shuffling* iteration increases the duration of the containment, thus affecting the network availability. Thus, we need to localize the *malicious clients* c in a minimum number of *shuffling* iterations x, and open a minimum number of ports p (equals to the number of the clusters and the issued secrets) in each *shuffling* iteration. To minimize the two parameters (p and x) for N clients, we create a corresponding optimization problem:

$$\text{minimize } A(p, x) = p * x \quad (1)$$

$$\text{subject to } |N|/(p/c)^x \leq 1 \quad (2)$$

To find the minimum values of x and p, inequality (2) is expressed as

$$|N|/(p/c)^x \leq 1 \Longrightarrow |N| \leq (p/c)^x \Longrightarrow p \geq c * |N|^{1/x} \quad (3)$$

and the result is substituted into equation (1) in order to express $A(p, x)$ as a function of one variable:

$$A(x) = (c * |N|^{1/x}) * x, \;\; x \neq 0 \quad (4)$$

To compute the minimum value of (4), the Closed Interval Method [13] is used. We have to solve $A'(x) = 0$.

Thus,

$$A'(x) = c * (|N|^{\frac{1}{x}} - \frac{|N|^{\frac{1}{x}} \ln(|N|)}{x}) = 0,\ x \neq 0 \quad (5)$$

Solving the above equation gives

$$x = \ln(|N|) \quad (6)$$

Substituting the solution (6) into (2) results in $p = |N|^{1/ln(N)} * c$.

Algorithm 1 Containment Algorithm

Input: A set $N = \{n_1, n_2, \ldots, n_i\}$ of clients, $c = 1$ as the first estimation
Output: Suspicious clients $N_s = \{n_{s1}, n_{s2}, \ldots, n_{sj}\}$ equal to compromised clients
$N_s \leftarrow N$
$(p, x) \leftarrow OPTIMUM(Ns, c)$
$CLUSTER(Ns, p)$
while $|N_s| \geq p$ **do** ▷ if $|N_s| \leq p$, the compromised ones are contained
  Check the ports to find the attacked ports.
  Remove the clients not using the attacked ports/the secrets $S_a = \{s_{a1}, s_{a2}, \ldots, s_{ak}\}$ from N_s
  if $c \geq |S_a|$ **then** $CLUSTER(Ns, p)$
  else
    $c \leftarrow |S_a|$
    $OPTIMUM(Ns, c)$
procedure OPTIMUM(Ns, c) ▷ finds min p and x
  $x = \ln(|N_s|)$
  $1 > |N_s|/(p/c)^x \implies p = \sqrt[\ln(|N_s|)]{|N_s|} * c$
  return p, x
procedure CLUSTER(N_s, p)
  Randomly split N_s into p-clusters and then issue p-secrets to the corresponding clients

4.1.3. Adaptive algorithm

We embody an adaptive optimization algorithm, which sets $c = 1$ and then computes the optimum p and x by solving the optimization problem above. After the execution of each *shuffling* iteration, if the number of compromised secrets is higher than c, the algorithm increases the number of issued secrets (clusters) p based on the number of compromised secrets (c)[9]. The pseudocode of the optimization-based containment algorithm is shown in Algorithm 1.

[9] An intelligent attacker who can pause his/her attack over time and/or cooperate with the others cannot evade this containment algorithm but might delay it.

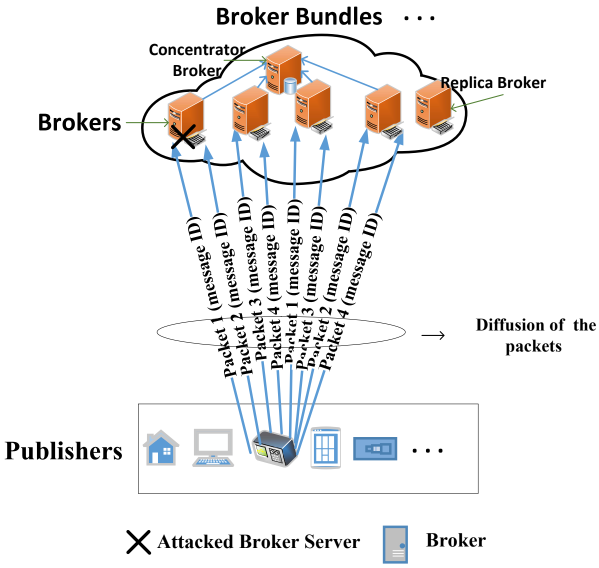

Figure 5: Packet Spreading

As a conclusion, PHSS consists of two main mechanisms i.e., port hopping based defense and packet spreading mechanism (see Section 4.2), which provide robust protection from DDoS attacks. Furthermore, to address the clock drift and compromising the secret key issues in the port hopping mechanism, we develop *a token-based authentication mechanism* and *a shuffling-based containment mechanism*. The idea behind the token-based authentication is to complicate the compromise of secrets. The *shuffling-based containment mechanism* is further introduced to localize the compromised secrets without rendering the broker server inaccessible for all the clients, unlike typical *port hopping* [8] [4] or moving target mechanisms [5].

4.2. Packet spreading

An attacker who controls a larger Botnet can bring down targeted brokers by flooding their entire ports or saturating the access link and thus, overcoming the *port hopping* mechanism. In such a case, the time period for re-establishing the connection could violate the availability requirements. To address this issue, we employ the data spreading mechanism [7, 6], which transmits by spreading consecutive data packets to broker servers within a *Broker Bundle* in a pseudo-random manner, as illustrated in Fig. 5. As shown in the figure, a *Broker Bundle* might consist of normal brokers, concentrator

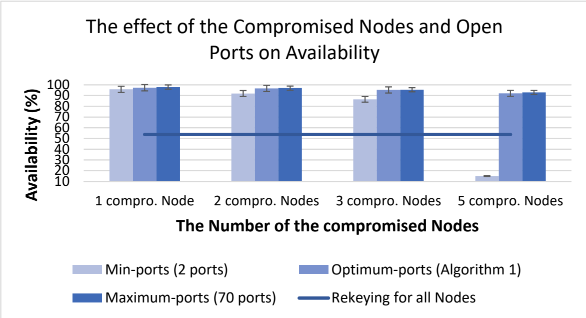

Figure 6: Implementation of PHSS on EC2 instance/server for 21 PlanetLab nodes

brokers and replica brokers. The role of the concentrator broker is to reassemble the packets received from the normal brokers. When some of the broker servers are brought down by the DDoS attack, we employ transmitting duplicate packets methods to ”recover” the dropped data and meet the availability requirements. In that way, the dropped packets do not affect the reassembling process, as the concentrator broker uses the duplicate packets for reassembling.

Moreover, utilizing the rapid-elasticity characteristic of the cloud computing, new/ready replica broker server(s) are instantiated to take over the attacked broker server(s). This provides an efficient attack mitigation also in cases of persistent threat. The IP addresses of the new replicas can be delivered to the publishers in encrypted form by performing a process similar to the sync-process.

5. Evaluation

In order to validate and provide realistic results on the efficiency of the proposed approach, we build a proof-of-concept prototype which consists of two EC2 micro instances (EC2) [9] and 21 PlanetLab nodes. To represent the SG applications with their strict requirements, we deploy a pseudo-state estimation application, which requires a latency of less than one second ($< 1s$) and a minimum of 30 samples per second [14] for a power grid that spans continental Europe. Hence, we employ all the properly functioning PlanetLab nodes (21 nodes) in Europe as publisher clients of the SG and two EC2 instances in EU-Central-1 (Frankfurt). The first EC2 instance represents a broker server in a *Broker Bundle*, while the second EC2 instance is a subscriber running the SG application in the third layer of HHCEC.

5.1. Evaluation metrics

The evaluation metrics used to assess our approach are availability, and throughput and latency overheads.

1. Availability: As responsiveness is a dominant concern for SG applications, we focus on network availability that refers to the success rate of timely delivery of the pseudo-state estimation application messages from SG publishers to subscribers over the broker server. This metric is used to measure the level of achieved network availability between beginning of the attack exploiting the compromised secret and the containment of the impact of the attack. For the containment of the impact of the attacks we use PHSS’s *shuffling-based containment mechanism* and the classical approach, which launches a rekeying process for all the clients using public key. Then, we compare their efficiency in providing availability during the same attack period.
2. Throughput and latency overhead: Throughput is defined as the successful forward of the pseudo-state estimation application messages to the subscribers over the broker server. The throughput overhead refers to the throughput decrease caused by PHSS on the broker server by comparing it with the simple transmission overhead. Furthermore, the additional latency imposed by PHSS is used as metric in the evaluation of our approach.

5.2. Proof-of-concept prototype-based evaluation

Our proposed software architecture is a middleware between the network stack and the pub-sub layer which runs on the broker servers and publishers. The middleware in broker server (i.e., the server stack) conducts the following tasks: (1) switching the open port depending on the output of PRF for the current secret and time, (2) answering the clients’ synchronisation messages and (3) executing Algorithm 1 to contain the impact of the DDoS attack utilizing the compromised secret.

The client side middleware (i.e., client stack) is responsible for: (1) producing the corresponding open port number of the broker servers using current secret and time to PRF, and (2) synchronising/updating the time/secret by sending a sync-request message to the broker servers. Moreover, to obtain a new secret while an attack is ongoing, each client stack sends a sync message each time when the server stack transmits a message requesting for a sync-request message.

The number of open ports p and the number of the *malicious clients* c are the two key factors for the efficiency of *shuffling-based containment mechanism* of PHSS during the clustering in each *shuffling* iteration, as pointed out in Section 4.1. Therefore, we evaluate the efficiency of *shuffling-based containment mech-*

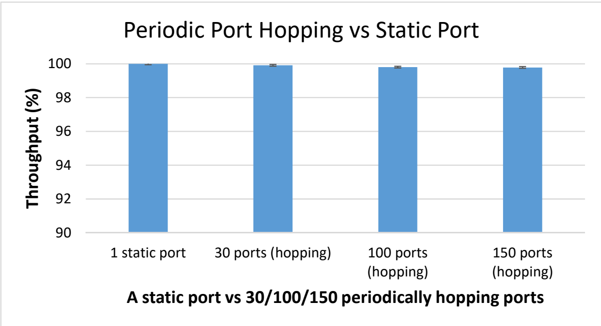

Figure 7: The effect of PHHS on the Throughput

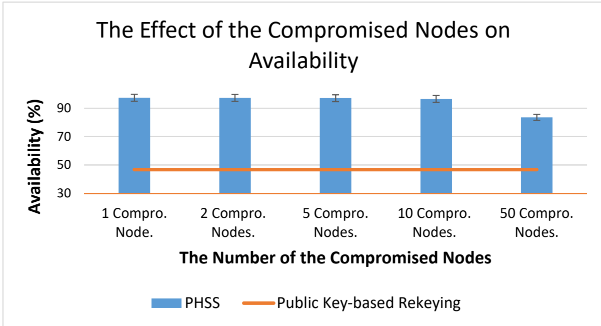

Figure 8: The effectiveness of PHSS while increasing the *malicious clients*

anism for these factors by comparing with the public key-based rekeying process.

In the public key-based approach the Authorization Server issues different secrets to each client to localize the *malicious clients* and mitigate the impact caused by the DDoS attack. However, this also increases the risk of attacks targeting the entire port range, since the broker server opens a different port each client.

Benchmark attack duration for our experiments is the period for containing the DDoS attack's impact through a public key-based approach. During this period, the successful message delivery rate of the pseudo-state estimation application refers to the network availability provided by the containment mechanisms. As the state estimation is one of the critical SG applications, we employ 4096 bits public key in our evaluation when comparing *shuffling-based containment mechanism* with the public key-based containment mechanism.

To the best of our knowledge, our proof-of-concept implementation-based experiment is the first real-world experiment of the *port hopping* approach in the literature. The related existing approaches focus only on the local network performance in case of a DoS attack or clock accuracy of *port hopping* mechanism [8, 10, 4].

5.2.1. Results discussion

Fig. 6 demonstrates that the number of open ports *p* significantly affects the availability, especially with the increase in the number of *malicious clients*. However, instead of opening the maximum number of ports (21 ports in this experiment, i.e., a port for each client), opening an optimum number of ports computed using Algorithm 1 provides availability close to the maximum availability provided by the 21 ports even when the number of *malicious clients* increases.

The straight line in Fig. 6 shows the successful delivery rate in the time period between the beginning of the attack and the containment of the impact of the DDoS attack exploiting the compromised secret by using the public key-based approach. PHSS using Algorithm 1 provides an **availability** over 98% in each case, whereas the public key-based approach caters an availability under 60%. The only case where PHSS provides lower availability than the public key-based containment approach is when using the minimum ports (2 ports) in each *shuffling* iteration despite the existence of more than a single *malicious client*.

Another aspect of the evaluation of our approach is the overhead in terms of service degradation of the broker server and the additional latency induced when PHSS is operating. Hence, we run the pseudo-state estimation application on the proof-of-concept prototype using both static port and *port hopping* mechanisms with variant numbers of the open ports. Fig. 7 shows that with up to 150 hopping ports, neither the switching ports nor the opening ports result in a significant impact. The **throughput** degradation of the broker server is <0.01% for 30 ports, which implies a successful response rate of the broker server for the pseudo-state estimation application. Opening more than 150 ports causes abnormal behavior of the broker server, but thanks to our optimization used by Algorithm 1, PHSS does not need such a high number of open ports, *p*. Moreover, we did not observe any significant additional **latency** when using our approach.

5.3. Emulation-based evaluation

To assess the effectiveness of our approach in large networks, we emulate the proof-of-concept in EC2's local network by creating 100 clients[10]. We employ Algorithm 1 to find the optimum number of open ports *p* in each run. In addition, the network includes different number of *malicious clients* in each run.

[10]More than 100 clients are not supported by the EC2-micro instance.

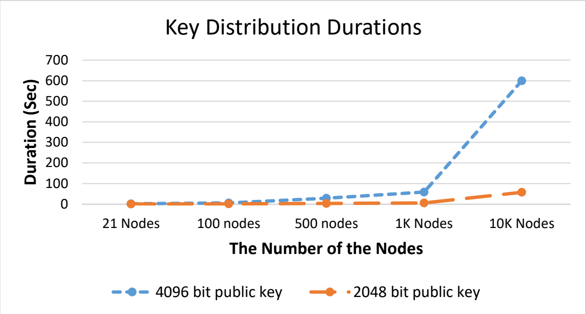

Figure 9: The rekeying process duration of different size of keys and nodes

Fig. 8 shows that with the increase in the number of clients from 21 (see Fig. 6) to 100, the public key-based containment method is able to contain, in a relatively longer time period, the impact of the DDoS attack that uses compromised secrets to discover server's open port. Accordingly, a notably higher loss of availability occurs.

Considering the case where PHSS is deployed, PHSS maintains an **availability** performance of up to 98% even where the *malicious clients* are up to 10%. After that the performance linearly degrades, as depicted in Fig. 8. The reason for the degradation is the increase in the number of the quarantined *malicious clients* that need to obtain new keys using the public key. Hence, if all clients are malicious, our approach loses its efficiency. However, PHSS takes advantage of the different session key for each client, which eliminates a high fraction of potential key breaches.

Finally, we demonstrate the duration of the key distribution ranging from 21 to 10K nodes for different sizes of the public key (i.e., 2048 bits and 4096 bits) in the case of usage of public key-based rekeying employed by the existing approaches. Fig. 9 shows that the increase in the number of clients strongly impacts the duration of containment of the damage of the DDoS attack as well as the network availability indirectly. As PHSS does not need the public key to sanitize all clients except the *malicious clients*, it significantly outperforms the public-key based rekeying approach when the number of clients increases. In addition, the key size is also an important factor: as shown in Fig 9, the rekeying process using 4096 bits key takes ten times longer than in the case of 2048 bits.

5.4. Synopsis

The evaluation of our approach focuses on the availability of the network and the induced overhead (i.e., throughput and latency). The experimental results denote that during DDoS attacks using the compromised secret, PHSS can provide network availability which is higher than 98% compared to the public key-based rekeying mechanism that provides availability below 60%. An increase in the number of the clients does not have a significant effect on the performance of PHSS, whereas it considerably affects the public key-based rekeying mechanism.

Unless all clients are malicious, PHSS significantly outperforms the public-key based rekeying approach. In addition, PHSS introduces negligible throughput and latency overheads, as depicted in the results.

6. Related Work

In this section, we hightlight the related work that fits into our context. The related works span three distinct areas: (1) securing Smart Grid, (2) adoption of cloud computing for Smart Grid, and (3) countermeasure techniques against DDoS attacks.

6.1. Securing Smart Grid

Since critical infrastructures (CI)s rely to an ever-larger extent on ICT, cyber security and resilience of CIs became more important cf. [15]. Security vulnerabilities of typical ICT can expose safety risk for CIs, particularly for the SG. Recently, many studies and projects have been introduced to identify potential vulnerabilities and threats and to develop new defense mechanisms. The CRISALIS [16] project focus on securing critical infrastructures from targeted attacks in addition to detecting vulnerabilities and attacks.

The authors in [17] propose the VIKING project which targets building methodologies for the analysis, design and operation of secure and resilient network-based industrial control systems for power transmission and distribution networks. C-DAX [18] takes advantage of a pub-sub paradigm to separate communication parties in space, time, and synchronization. To provide a secure communication, C-DAX provides authentication of nodes, end-to-end integrity and confidentiality of the messages, and topic access control. Despite the absense of a defense mechanism against DDoS attacks, they provide promising features to incorporate with our approach to enable secure and resilient communication and control for critical infrastructures.

6.2. Cloud computing for Smart Grid

Multiple features of cloud computing, such as on-demand service, flexibility, pay-for-use and instant network access, are continuously attracting the attention of researchers working on system development for potential future power grids [2].

In order to design a prototype and present a well-defined software platform with the aim of realization of the requirements of the future power grid in the cloud, GridCloud [1] was proposed. GridCloud develops a cloud architectural model for monitoring, management and control of the power systems, which is achieved by integrating some of the technologies such as GridStat, Isis 2, TCP-R and GridSim [1].

A contemporary approach for power system frequency monitoring system (FNET) [19] is proposed as a wide-area monitoring system. The main architecture of FNET includes a broadly deployed network of frequency disturbance recorders (FDR) which returns phasor readings to either local central point or a remote data center with Ethernet. Handling the data of the FNET application with diverse configuration requirements (number of CPU, memory, etc.) by using in-house infrastructures doesn't result in a cost-effective solution for the power grid entity. Leveraging the cloud computation for the FNET applications would be the most feasible solution [1].

The authors in [20] propose a framework, GridCloud, which enables PMU-based state estimation application on a cloud infrastructure. To identify the limitations of the current standard cloud infrastructures, the authors carry out a real-world implementation, using the Red-Cloud and PlanetLab infrastructures. As the results indicate, the authors infer that a best effort state estimation can be fulfilled by using the timely arrived data. Otherwise, the outdated data can be used for historical analysis.

[21] introduced a smart-frame, which consists of three hierarchical levels, i.e., top, regional and end user, for the SG application based on cloud computing. This framework is designed to provide scalable, flexible and secure information management for those applications. In addition, to address information security issues in this frame, a security solution based on identity-based encryption and signature, and identity-based proxy re-encryption are proposed.

The aforementioned existing work provides the basic inspiration behind the design of HHCEC. However, our contribution is a dispersed and hybrid design architecture in HHCEC to provide secure and high responsiveness for the SG applications.

6.3. DDoS attack defense mechanisms

The traditional security solutions, e.g, firewalls, intrusion detection systems (IDS), or Virtual Private Networks (VPN), are both widespread and effective. However, since the SG devices typically have constrained computational, bandwidth and memory resources, the direct use of these traditional security mechanisms is mostly not possible [6, 22]. Hence, for providing the required security for SG communication systems, security solutions that proactively counter the attacks should be employed. Within this context, we develop our approaches based on the following proactive approaches.

[3, 5, 23] are proactive DDoS attack defense mechanisms, which aim at hiding or moving the position of the application sites to prevent DDoS attacks based on the available information about their locations.

An overlay-based target hiding technique is proposed in [7] where the authors propose to spread the duplicated data packets over the overlay nodes between the client and the target. This ensures a robust protection against DDoS attacks that make some of the overlay nodes unavailable at the expense of latency and packet overheads. An enhancement to [7], the authors in [6] use a multihoming-based quick recovery strategy which transmits consecutive packets to several network interfaces of overlay nodes. This enables a rapid request for the dropped packets when one of the interfaces is under attack. Although these approaches provide a robust defense mechanism against DDoS attack, investment and maintenance costs, as well as the high latency, render these approaches difficult to deploy for latency sensitive applications of the SG.

Further examples of the moving target defense are port and address hoping techniques. [8] presents a random port hopping (RPH) technique where the server uses time-varying UDP/TCP port number, as well as a shared secret between the server and clients. [4] states that the RPH in [8] undergoes time differences due to the local clock drift. In order to address time-synchronization issue in [8], [4] proposed two algorithms, BiGWheel and HoPerAA, which enable the RPH for multiple servers and clients in the presence of clock-drift. In this approach, the secret is used by the clients without a restricted time duration, which poses the risk of compromising the secret. With the compromised secret, the communication will be interrupted for a certain time duration because of the direct attack against the ports.

The time synchronization issue is also addressed by [10] through an ack-based port hoping strategy. However, in case of losing acknowledgment packet in the network, the two sides are forced to communicate on a common port for a longer time period. This enables the attacker to obtain the port number to start a directed attack to disrupt the communication. Moreover, this scheme is not a practical scheme for communication when multiple users exist.

Demir at al. [24] propose a defense approach, which hides the open port number by switching the subflows of Multipath TCP for SG applications that need long duration TCP connection. However, many critical SG applications are compatible with UDP connection.

A shuffling-based moving target defense mechanism is also proposed to reduce the level of large-scale DDoS attacks with the help of cloud computing properties [5]. Replacing attacked servers with newly instantiated replica servers and optimally shuffling client-to-server assignments, this solution can gradually isolate DDoS attacks on the network and computation resources and thus, restore quality of service for benign-but-affected clients. This method is actually a reactive method and not convenient for applications requiring high availability.

Based on the above discussion of the related work, our proposed novel cloud-assisted architecture (HHCEC) and the DDoS defense mechanism (PHSS) are developed by considering the SG security threats and requirements associated with the network model.

Moreover, As our approach focus on solely filtering unauthorized traffic with minimum cost, it could not counter high volume DDoS attacks (multi Terabit). To counter such attack, our approach can be complemented by some approaches that characterize the regular traffic of a service depending on the IP-prefix to detect the attacks. This approach also takes advantage of rapid-elasticity of cloud computing to continue the service during filtering the suspicious traffic, cf. [25].

7. Conclusion

We have proposed a cloud-assisted DDoS attack resilient communication platform. Our first contribution was a hierarchical hybrid cloud-assisted architecture (HHCEC), aimed at meeting scalability and security requirements of the SG applications in the cloud adoption. We employed a publish-subscribe system on the HHCEC, as the pub-sub message passing paradigm matches with the SG's data acquisition paradigm. However, DDoS attacks against the brokers pose availability risk for time-sensitive critical SG applications. To cope with this, we proposed the port hopping spread spectrum (PHSS). The *port hopping* mechanism of PHSS basically prevents the brokers from transport and application layer DDoS attacks by switching the open port over time in a pseudo-random manner.

This enables the broker to drop the invalid packets in the firewall to avoid the application-based filtering. In addition, to overcome the relatively high-volume flooding attack, PHSS spreads the consecutive packets over the brokers in a *Broker Bundle* by duplicating them depending on the application priority.

Furthermore, the existing *port hopping* mechanisms use a secret shared between all parties to produce the same open port number in the same time, which pose a high security risk in the case of the compromise of the secret. The containment of the impact of the DDoS attack utilizing the compromised secret is fulfilled by an Authorization Server using a secure channel (a public key-based rekeying process). However, the brokers become unavailable until the public key-based rekeying process is done, which leads to the loss of availability that violates the requirements of the SG applications. To address this issue of the existing approaches, we employ a *token-based authentication mechanism*, enabling the brokers to regularly issue the secret in encrypted form by using the session key of each publisher. Moreover, to contain the damage of the DDoS attack employing the compromised secret, we introduce a *shuffling-based containment mechanism*, which delivers new secrets (causing new ports) to each cluster after shuffling and clustering the publishers. By repeating this process on the clients in the cluster(s) whose secrets are still used for the attack, the port *shuffling* mechanism progressively isolates the *malicious clients*.

Using a proof-of-concept platform consisting of Amazon EC2 micro instances and PlanetLab nodes, we evaluated the effectiveness of our approach in providing availability in the case of DDoS attacks exploiting privilege of the compromised secret by comparing with the public key-based rekeying mechanism. The results show that our approach significantly increases availability in comparison to the public key-based rekeying mechanism, since it contains the impact of the DDoS attack utilizing the compromised secret in a notably shorter time period.

[1] David Bakken and Krzysztof Iniewski, Smart Grids: Clouds, Communications, Open Source, and Automation, CRC Press, 2014, pp. 1–32.

[2] Samaresh Bera, Sudip Misra and Joel Rodrigues, in: Cloud Computing Applications for Smart Grid: A Survey, in: IEEE transactions on Parallel and Distributed Systems, 2015, pp. 1477–1494. Vol. 26, No. 5.

[3] Angelos Stavrou, Angelos Keromytis, Jason Nieh, Vishal Misra and Dan Rubenstein, MOVE: An End-to-End Solution to Network Denial of Service, in: Proc. of the 12th ISOC Symposium on Network and Distributed System Security (SNDSS), 2005, pp. 81–96.

[4] Zhang Fu, Marina Papatriantafilou and Philippas Tsigas, Mitigating Distributed Denial of Service Attacks in Multiparty Applications in the Presence of Clock Drifts, in: IEEE transactions on Dependable and Secure Computing, 2012, pp. 401–413. Vol. 9, No. 3.

[5] Quan Jia, Huangxin Wang, Dan Fleck, Fei Li, Angelos Stavrou and Walter Powell, Catch Me If You can: A Cloud-enabled DDoS Defense, in: Proc. of the 44th annual IEEE/IFIP international conference on Dependable Systems and Networks (DSN), 2014, pp. 264–275.

[6] Kubilay Demir and Neeraj Suri, SeReCP: A Secure and Reliable Communication Platform for the Smart Grid, in: Proc. of the 22nd IEEE Pacific Rim International Symposium on Dependable Computing (PRDC), 2017, pp. 175–184.

[7] Angelos Stavrou and Angelos Keromytis, Countering DoS Attacks with Stateless Multipath Overlays, in: Proc. of the 12th ACM conference on Computer and Communications Security (CCS), 2005, pp. 249–259.

[8] Henry Lee and Vrizlynn Thing, Port Hopping for Resilient Networks, in: Proc. of the 60 IEEE Vehicular Technology Conference, 2004, pp. 3291–3295.

[9] Amazon Web Services, Amazon Elastic Compute Cloud (EC2), Last accessed on 20/01/2017.

[10] Gal Badishi, Amir Herzberg and Idit Keidar, Keeping Denial-of-Service Attackers in the Dark, in: IEEE transactions on Dependable and Secure Computing, 2007, pp. 191–204. Vol. 4, No. 3.

[11] Clifford Neuman and Theodore Ts'o, Kerberos: an Authentication Service for Computer Networks, in: IEEE Communications Magazine, 1994, pp. 33–38. Vol. 32, No. 9.

[12] Thomer Gil and Massimiliano Poletto, MULTOPS: A Datastructure for Bandwidth Attack Detection, in: Proc. of the 10th USENIX Security Symposium, 2001, pp. 23–28.

[13] Eldon Hansen and William Walster, Closed Interval Systems, in: Global Optimization Using Interval Analysis, CRC Press, 2003, pp. 43–82.

[14] Yannan Wang, Pradeep Yemula and Anjan Bose, Decentralized Communication and Control Systems for Power System Operation, in: IEEE transactions on Smart Grid, 2015, pp. 885–893. Vol. 6, No. 2.

[15] Randy H. Katz at al., An information-centric energy infrastructure: The berkeley view, pp. 7 – 22. Vol. 1 No. 1.

[16] Marco Caselli, Intrusion detection in networked control systems: From system knowledge to network security, Universiteit Twente, 2016. Eemcs-eprint-27385.

[17] Teodor Sommestad and Hannes Holm and Mathias Ekstedt, Estimates of success rates of remote arbitrary code execution attacks, pp. 107–122. Vol. 20 No. 2.

[18] Florian Heimgaertner et al., A security architecture for the publish/subscribe c-dax middleware, in: IEEE International Conference on Communication Workshop (ICCW), pp. 2616–2621.

[19] Yingchen Zhang, Penn Markham, Tao Xia, Lang Chen, Yanzhu Ye, Zhongyu Wu, Zhiyong Yuan, Lei Wang, Jason Bank, Jon Burgett, Richard Conners and Yilu Liu, Wide-Area Frequency Monitoring Network (FNET) Architecture and Applications, in: IEEE transactions on Smart Grid, 2010, pp. 159–167. Vol. 1, No. 2.

[20] Ketan Maheshwari, Marcus Lim, Lydia Wang, Ken Birman and Robbert Van Renesse, Toward a Reliable, Secure and Fault Tolerant Smart Grid State Estimation in the Cloud, in: Proc. of the 4th IEEE/PES Innovative Smart Grid Technologies Conference (ISGT), 2013, pp. 1–6.

[21] Joonsang Baek, Quang Hieu Vu, Joseph Liu, Xinyi Huang and Yang Xiang, A Secure Cloud Computing Based Framework for Big Data Information Management of Smart Grid, in: IEEE transactions on Cloud Computing, 2015, pp. 233–244. Vol. 3, No. 2.

[22] Saman Taghavi Zargar, James Joshi and David Tipper, A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks, in: IEEE Communications Surveys Tutorials, 2013, pp. 2046–2069. Vol. 15, No. 4.

[23] Angelos Keromytis, Vishal Misra and Dan Rubenstein, SOS: Secure Overlay Services, in: Proc. of the ACM SIGCOMM conference on Applications, technologies, architectures, and protocols for computer communications, 2002, pp. 61–72.

[24] Kubilay Demir and Neeraj Suri, Towards ddos attack resilient wide area monitoring systems, in: Proc. of the 12th International Conference on Availability, Reliability and Security, ARES.

[25] Vincenzo Gulisano and Mar Callau-Zori and Zhang Fu and Ricardo Jimenez-Peris and Marina Papatriantafilou and Marta Patino-Martinez, Stone: A streaming ddos defense framework, 24, pp. 9620 – 9633. Vol. 42, No. 24.