Secure Communications for Ancillary Services

Matthias Krebs(✉), Stefan Röthlisberger, and Peter Gysel

School of Engineering, University of Applied Sciences
Northwestern Switzerland, Windisch, Switzerland
{matthias.krebs,stefan.roethlisberger,peter.gysel}@fhnw.ch

Abstract. Networking of distributed energy resources for ancillary services like control pooling introduces new security challenges. For economic reasons public IP networks are often used for the transport, resulting in sophisticated security requirements. Legacy devices as well as compliance with corporate network security policies must be taken into account. In this paper, we compare different communication technologies and discuss the problems of integrating legacy devices. We describe an approach that uses standardized technologies to provide secure communications for ancillary services, while at the same time requiring minimal configuration by administrators of corporate networks.

Keywords: Ancillary services · Security · Communications · Standards

1 Introduction

As far as communication in ancillary services is concerned, we consider a scenario as depicted in Fig. 1. Networked control devices, which can be controllers for different kinds of power-consuming or power-generating equipment such as heat pumps, furnaces or hydropower stations, are distributed in homes and industrial facilities. They are owned by private individuals or businesses, not grid operators. In order to participate in ancillary services, they communicate with service providers who offer services like optimization of energy consumption or control pooling. Communication takes place over corporate IP networks or even Internet, because participants can be distributed over a large area, and since the service providers are not grid operators, they have no access to communication over power lines. An other reason is that using IP is relatively inexpensive, considering most homes and businesses already have internet access.

With devices distributed across large areas and communicating over a network security becomes relevant. In order to ensure correct operation and billing, especially when participating devices are remotely controlled, they must be authenticated and data sent over the network must be protected against manipulation. Furthermore, critical systems like the provision of electrical power should not be left vulnerable to cyber attacks.

While new equipment designed for networking, such as energy management devices and load controllers, is generally being installed in new buildings or after

S. Gottwalt et al. (Eds.): EI 2015, LNCS 9424, pp. 141–152, 2015.
DOI: 10.1007/978-3-319-25876-8_12

fundamental renovation, many companies and homes already possess existing control devices. In some cases these devices can already be considered networked in some way, because they feature interfaces accessible via IP like Modbus [20], thus providing some level access to other systems. It is not to be expected that functional equipment is thrown away and replaced without necessity, after all considerable investments may have been made and should be amortized before new equipment is purchased.

Therefore, the integration of existing devices has to be taken into account, despite the fact that these devices might not fulfill the security standards necessary for communication over Internet. In that case, the devices are considered legacy. Integrating these legacy devices introduces additional challenges, because not only does it require secure communication with the outside world, but also protection of the local network.

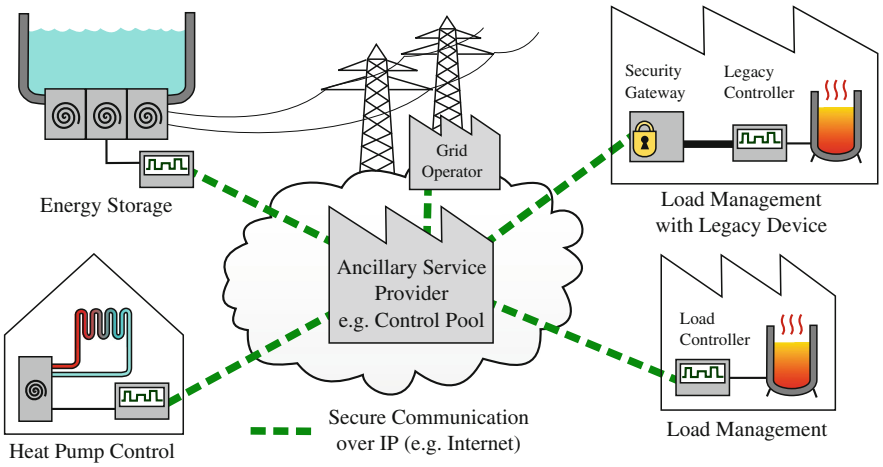


Fig. 1. Ancillary Services in a wide-area scenario

This paper focuses on the challenges of finding a secure IP communication approach that is suitable for ancillary services and complies with security requirements of corporate networks. In Sect. 2, we discuss related work. The challenges which arise when deploying a device in a corporate network are discussed in Sect. 3. We then compare potential protocols in Sect. 4. A use case of a distributed load management system is presented in Sect. 5. We conclude our paper with a reflection about the feasibility of the chosen approach and future developments.

2 Related Work

The German Federal Office of Information Security has worked out a protection profile for smart meter gateways [1]. The document describes security objectives and their requirements. Such a gateway has a connection to the Internet and

accesses smart metering devices locally. The gateway's job is to collect, process and transfer data from the attached meters. In terms of security, the goals are protecting the privacy of the consumers, ensuring a reliable billing process and protecting the power grid as a whole. An important requirement is the usage of a security module (e.g. a smart card) [2] providing various functions related to encryption and authentication. A second premise is that all devices communicating with the gateway have to use encryption and mutual authentication, thus making the integration of legacy devices impossible.

The Internet Engineering Task Force (IETF) has published a document on how Internet communication protocols could be used with networked energy resources. It has been released as RFC 6272 [13]. The document covers virtually all aspects of networking, including network topology, secure communication and different application protocols. Since it can be thought merely of a reference, not an actual communication standard, the document does not provide concrete recommendations on which communication approaches should be chosen.

The design of a secure access gateway for home area networks is considered in [3]. Their article focuses on secure, real-time remote monitoring and control of managed devices using a smart phone. The proposed system architecture enables the managed devices to send alerts to the smart phone. The emphasis is on physical layer security of wireless networks (e.g. OFDM and GSM) and capacity challenges therein.

The European Telecommunications Standard Institute (ETSI) has approved a communication standard [8] for networked energy resources, the Open Smart Grid Protocol (OSGP) [7]. The standard specifies networking protocols and data models for the data transfer of smart meters. The default approach is communication over a power line channel (PLC), although it does not depend on a specific physical layer. Custom cryptography methods are used for security.

The authors of [4,5] consider using the WebSocket protocol [14] in machine-to-machine communications. The former focuses on electric vehicles communicating over cellular networks, whereas the latter describes a gateway that accesses a wireless sensor network and forwards the collected data through WebSocket.

3 Challenges

Searching for an approach to secure communication for ancillary services, we have identified a number of challenges. We explain these challenges in the following paragraphs.

Secure Communication becomes important whenever it takes place over an insecure channel. Current smart meters, as an example, often communicate through the power line (PLC) and not IP because the electricity provider can access it. Since everyone could tap into a power line PLC can be considered insecure. The same applies to devices that communicate over an IP network, because the data packets are routed across a geographically distributed infrastructure and could be tampered with anywhere along their way. It is therefore essential that all the data exchanged between a networked device and a service provider

is encrypted end-to-end, so that no manipulation can occur. It is also important that the service provider can be certain it is talking to the proper device, thus an authentication and identity assertion method is necessary. There must be no direct remote access to the participating devices, and the service provider's infrastructure must be secured in particular, because a potential attacker could gain access to all connected devices. We do not tackle denial-of-service attacks as they can never be fully prevented.

Compliance with corporate network security policies is important when a device is to be integrated into a corporate network. Many companies employ restrictive policies regarding network security. This means that they are not willing to lessen their security policy just for one device. Such companies generally use at least a firewall which blocks all inbound traffic from the Internet by default. Even more restrictive configurations include blocking most TCP/UDP ports from the corporate network to the Internet, except for very common ports (HTTP, HTTPS). A networked device has to respect that and provide means to operate under these circumstances. Inbound connections should thus be avoided whenever possible. Additional obstacles are introduced by the utilization of proxy servers that cache data, restrict and filter access to the Internet. These often limit access to web protocols like HTTP and HTTPS and do not allow other protocols to pass. A threat to secure communication is deep packet inspection (DPI), which some firewalls or proxy servers perform. It inspects even encrypted traffic by decrypting, analyzing and re-encrypting the packets prior to forwarding. A custom certification authority (CA) is declared trusted on the clients inside the corporate network, and instead of presenting the real server certificate to clients when they open an HTTPS web site, a certificate signed by the custom CA with the same server name is given. Because the clients trust the custom CA, they accept the forged certificate. The result is encrypted communication between the client and the proxy server, but the latter is able decrypt the data. After analysis and approval of the client's messages, the proxy then forwards them to the actual server, now using the real server certificate for encryption. The actual problem is that this qualifies as a man-in-the-middle attack, as the proxy could manipulate the data. End-to-end encryption would be impossible in such a situation.

Uncomplicated configuration is desirable, because the installation of a new device should not force the administrator to configure complex firewall rules or even open inbound ports. Using standardized Internet protocols on standard ports can circumvent this problem.

The *integration of legacy devices* adds an extra layer of complexity, because by our definition a legacy device itself cannot communicate in a secure fashion. Therefore, the insecure communication must be encapsulated by a secure communication protocol, which is then used for data exchange over the Internet. This problem can be addressed through the introduction of additional infrastructure, such as a virtual private network (VPN) gateway that routes any network traffic over an encrypted tunnel [6], or a security gateway (SGW) as depicted in Fig. 2, which exclusively has access to the Internet and acts as a protocol converter encapsulating the legacy device's messages.

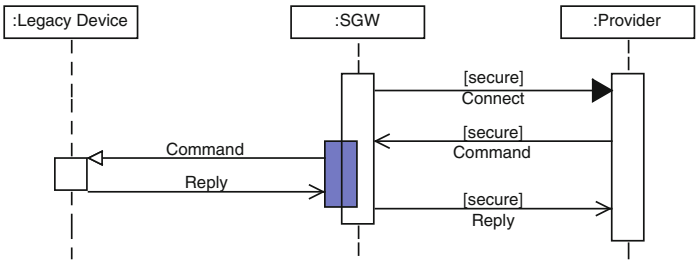


Fig. 2. Communication between a service provider and a legacy device, using a security gateway as intermediary

4 Evaluation of Communication Protocols

To find a suitable approach, we have based our evaluation on the following five requirements for communication between a device and the service provider. They have been chosen to allow for seamless adoption into a corporate environment.

1. Efficiency, because bandwidth may be limited (e.g. GSM/GPRS)
2. End-to-end encryption and integrity protection with mutual authentication
3. No obligation for firewall modifications, especially not opening a port to allow Internet traffic into the company's network
4. Ability to pass through intermediary proxy servers
5. Bi-directional communication ability.

4.1 Security

We have considered two approaches to secure the communication channel, independent of the transport protocol to be used.

Transport Layer Security. The Transport Layer Security (TLS) protocol [11] provides privacy, data integrity and authentication using a combination of asymmetric and symmetric cryptography. When a connection is established, a handshake is performed first, using certificates containing a public and private key to exchange a symmetric key that will be used to encrypt the data packets (e.g. AES-256). By obfuscation of the transmitted data through encryption, privacy is guaranteed. Message authentication codes (MAC) prevent message tampering and forgery. At least a server certificate is mandatory for an encrypted connection, and an optional client certificate can be given to act as authentication credentials. TLS adds a transparent security layer to any TCP connection, thus being compatible with any transport protocol built upon TCP.

Virtual Private Network. A VPN offers a solution to interconnect two remote networks through a bi-directional encrypted tunnel. A client connected to a

VPN is logically in the same LAN as the VPN server is attached to. A VPN tunnel encapsulates the whole network stack, so it can transport any kind of network traffic transparently. Depending on the implementation, different encryption methods are used.

An advantage is that any application protocol, e.g. one of a legacy device, can be accessed over a secure connection from a remote location. As an example, a solution that uses VPN gateways and a cloud-based VPN concentrator is offered by ADSTec [6]. It uses TLS for encapsulation and allows remote access to devices in a local network.

There are some disadvantages, too. The encapsulation of the whole network stack results in considerable overhead, which adds to the overhead introduced by TLS. Without careful configuration, new security holes may be introduced as well, because unlike a TCP connection using TLS, whose encapsulated channel remains a single point-to-point connection, a VPN can link entire network segments and thus has the potential to expose services that should not be exposed.

4.2 Communication Protocols

Our goal is to find an efficient approach which can be used without compromising the security of a corporate network. We have analyzed several communication protocols. In the following sections, we compare them in terms of efficiency and how well they meet the previously mentioned challenges. An overview of the discussed protocols and their fulfillment of the requirements can be found in Table 1.

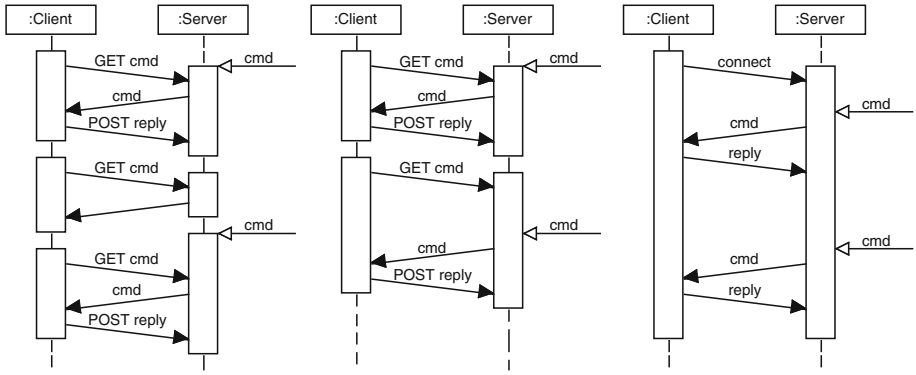
HTTP Polling and Long Polling. The Hypertext Transfer Protocol (HTTP) is a widely used request-response-based protocol used to communicate with web servers. We have looked at two variants. *Polling* (Fig. 3a) repeatedly sends requests to the server. The server immediately responds either with new information or an empty response. The second variant is *Long Polling* (Fig. 3b). The main difference is that Long Polling does not send empty responses back to the client, instead the connection is kept open until a request can be answered with new information.

Server-Sent Events. *Server-Sent Events* are currently being standardized as part of HTML5 by the W3C [12]. It offers a light-weight approach to push messages from the server to the client. The client initiates the connection which is basically an HTTP GET request with the *Content-Type* header set to *text/event-stream*. The server keeps the connection open and sends (pushes) multiple messages to the client until the connection is explicitly closed by the server or the client.

WebSocket. *WebSocket* (see Fig. 3c) is a bi-directional protocol using a single socket for communication. It is specified in RFC6455 [14] and is a W3C working draft [15]. A WebSocket client establishes a connection using the HTTP *upgrade* header during the initial handshake. The HTTP connection is then upgraded to a

WebSocket connection. After that, it is no longer considered an HTTP connection. After being established, the connection persists until a participant closes it explicitly. WebSocket provides message-based communication with minimal overhead.

Raw TCP Sockets. An established TCP connection (see Fig. 3c) provides a bi-directional communication channel that transports a stream of binary data. Tasks like distinguishing individual messages have to be adopted by a higher-level application protocol, as a raw TCP connection provides no such means itself. Examples of protocols built upon TCP sockets are XMPP or SIP, as specified in RFC6272 [13].



(a) HTTP Polling (b) HTTP Long Polling (c) WebSocket or Raw TCP

Fig. 3. Communication between client and server

4.3 Firewall Friendliness

Assuming that a firewall allows outgoing TCP connections on all ports from the company's network to the Internet, all of the described protocols will work without a hassle. In contrast, if the firewall is restricted to only allow Internet access over standard HTTP ports (80 for plain HTTP, 443 for HTTPS), which is likely in many corporate networks, an application protocol based on a raw TCP socket cannot be used if it is configured to use a different port.

Transport protocols based on HTTP are usually configured to use a standard port by default, so no modifications have to be made to the firewall. If a protocol uses these ports but is in fact not based on HTTP, it might be blocked by a packet-inspecting firewall. Protocols like WebSocket circumvent this problem by using an actual HTTP request to initiate the connection and then perform an HTTP Upgrade to switch to the actual protocol, so the firewall treats it as HTTP-based.

VPN implementations using TLS, like OpenVPN, can operate on a single socket on port 443. Other technologies, like IPsec, may require a dedicated port to be opened on a firewall and sometimes enabling specific configuration options such as *VPN pass-through*, or even inbound ports. This would contradict requirement 3.

4.4 Proxy Server Traversal

Compatibility with proxy servers depends on the kind of proxy being used. A *transparent* proxy is integrated into the network so that traffic is automatically routed through the proxy, with a client being unaware of its existence. The proxy usually just forwards traffic and might provide caching or filtering in the case of plain HTTP.

An *explicit* proxy is a different story. A client must be configured to use the proxy, or it does not get any Internet access. The proxy is limited to HTTP connections, so in theory all HTTP-based protocols should work and raw TCP connections should fail. TLS should fail, too, because to the proxy it looks like a raw TCP connection. However, HTTP-based protocols, which includes Server-sent Events and WebSocket, first issue a plain HTTP CONNECT request when TLS is used in combination with an explicit proxy [18]. In that case, the proxy just forwards all subsequent encrypted traffic unmodified. Other TCP clients and VPN clients that do not support the HTTP CONNECT method cannot be used in combination with an explicit proxy.

Proxy servers performing deep packet inspection on TLS connections prevent end-to-end encryption (requirement 2). The only solution is to configure an exception rule for those clients requiring end-to-end encryption.

4.5 Performance

Performance is influenced by several factors. One is protocol overhead. Regarding HTTP, request and response headers are a common cause of overhead. The amount of header information heavily depends on the application, it can take from 100 bytes to more than 1 KB for each request or response. We assume a scenario where commands may be issued from the server to the device, so when using HTTP Polling, the device has to poll the server with HTTP GET requests to check for new messages. If there is a message it is included in the HTTP response body. Otherwise, the response contains just overhead. Messages from the device are sent to the server as HTTP POST requests.

HTTP Long Polling is similar, but the device polls and then waits for a response, so there is less overhead generated than through continuous polling.

When using Server-Sent Events, the server can push messages to the device, because a persistent uni-directional connection from the server to the device is established. Nevertheless, the device has to use additional HTTP POST messages to send data back to the server, since Server-Sent Events do not offer a bi-directional channel.

WebSocket requires just one HTTP request to initiate the connection, after that it is a bi-directional and persistent communication channel, similar to a raw TCP socket. Each WebSocket message has an overhead of 2 to 12 bytes, depending on the payload size.

In [19] the authors compare HTTP Polling with WebSocket in terms of protocol overhead. The HTTP header they use is 871 bytes long, and a WebSocket

Table 1. Comparison of protocol capabilities

	Efficient	TLS	No firewall setup needed	Proxy server pass-through	Bi-directional
HTTP Polling	X	✓	✓	✓	X
HTTP Long Polling	X	✓	✓	✓	X
Server-sent Events	X	✓	✓	(✓)	X
WebSocket	✓	✓	✓	(✓)	✓
Raw TCP socket	✓	✓	(✓)	X	✓

message has an overhead of 2 bytes. They test 1'000, 10'000 and 100'000 simultaneous requests per second to illustrate the difference. At 10'000 requests per second HTTP Polling uses 66 Mbps, compared to 0.153 Mbps using WebSocket. In a comparison between HTTP Long Polling, WebSockets and raw TCP sockets, the TCP socket was always faster and had more throughput than the others [17]. The larger the payload was, the larger the difference became. Similar results are shown in [4], where the bitrate of WebSocket is measured 60–70% lower than HTTP Polling, and TCP being the most efficient.

Besides protocol overhead, the use of TLS encryption introduces additional overhead. Establishing the connection can require 6 to 10 KB, because certificates have to be exchanged during the hand-shake. The exact size depends on the cipher suites being used and the certificate's key length. In an open connection, overhead of an encrypted data packet would not exceed 60 bytes (including a maximum of 31 bytes for AES-256 padding). Thus TLS shows best efficiency in persistent connections.

In cases where a VPN connection is chosen to secure communication, further overhead is generated. A TLS-based VPN like OpenVPN has to encapsulate the full network stack, including IP and TCP or UDP headers of each packet to be encapsulated. These packets are then broken down into TLS records and again into TCP packets.

An other performance indicator of a communication channel is latency. In [16] the authors compare the latency of HTTP Polling, HTTP Long Polling and WebSocket. With polling it is measured 2.3 to 4.5 times higher than with WebSocket. HTTP Long Polling has achieved both lower and higher latency in comparison to WebSocket, depending on the situation. Over the longest distance (Canada to Japan), the average latency of WebSocket is 3.8 to 4.0 times lower compared to HTTP Long Polling.

4.6 Decision

After comparing the capabilities of the protocols we analyzed, we have decided to go with WebSocket, as it fulfills our requirements in Table 1 best. It offers efficient bi-directional communication and plays well even in corporate environments. Its high level API provides connection control and message handling, thus an

application protocol does not need to implement these functions. Security can be implemented using standardized TLS encryption and authentication without much additional overhead, because WebSocket is already designed for persistent connections, which is also the most efficient scenario for TLS.

We have decided against VPN for multiple reasons. Since VPN is designed to interconnect whole networks, malicious traffic could be injected. This happened in 2003 [10], where the SQL slammer worm propagated through VPNs. A similar situation could happen if the service provider's endpoint is compromised, whereby an attacker would potentially gain access to all client networks connected to the endpoint. Complex configuration is necessary, for example with IPsec, which requires specific firewall settings and inbound connections, Also, a VPN approach adds more complexity in case of a single device, which most likely just uses a single application protocol. When legacy devices are to be integrated, we opt for a security gateway (SGW) that provides a secure connection and converts the legacy device protocol.

We have also looked at existing standards such as OSGP. One problem is that it is designed for PLC, while we need an approach that works over IP networks. The major problem, however, is security. A cryptographic analysis [9] has found several weaknesses, such as non-standard digest functions and the use of RC4. Due to the security issues we would not use OSGP in its current form.

5 Use Case of a Distributed Load Management System with Legacy Devices

In a project funded by the Swiss Federal Commission for Technology and Innovation (CTI) we have developed a prototype of a distributed load management system for control pooling. The load controllers are of legacy kind. They are already widely deployed and still perfectly capable of performing their duty, but due to technical limitations they cannot be upgraded with a secure communication method. Hence they are not suitable for Internet communication.

To address this problem, we have developed a security gateway (SGW), which is an embedded device that is installed in the customer's corporate network together with the load management device. The SGW subsequently provides a secure communication channel to a centralized control service acting as the service provider. The actual data exchange with the load controllers remains in the customer's network. Based on our evaluation we have chosen WebSocket for communication between the SGW and the control service. The SGW device features an ARMv7 processor and is based on an embedded Linux platform. The gateway software is implemented in Java running on Java SE Embedded. The control service is also implemented in Java and thus shares part of its code base with the SGW. Messages exchanged between the SGW and the control service are encapsulated using a custom data model based on compact XML. We have chosen this approach for reasons of flexibility.

The secure connection employs TLS v1.2 with AES-256 [11]. We use the mutual authentication feature of TLS, giving each SGW its own certificate. The certificates are signed by a custom certificate authority (CA) created solely for use in our system. Since our CA is the only one trusted by the SGW and control service, we can ensure that man-in-the-middle attacks are not possible. Certificates signed by a different CA would automatically be rejected. All certificates can be revoked through a certificate revocation list (CRL), which is queried by the SGW and the control service upon establishing a connection. This allows us to deny access in case a device or its certificate is stolen.

We have conducted feasibility tests in a real-world environment, where communication effectively takes places over a public Internet connection. The load controllers were accessed through the SGW five times a second, with an XML message of 480 bytes. Results have shown that the SGW's processor can easily handle the TLS-encrypted WebSocket protocol. Round-trip time is mostly affected by the number of hops and the connection with the lowest bandwidth. Sending 480 bytes over a 20 kbps GPRS connection would take 192 ms and 9.6 ms over 400 kbps DSL. We achieved a total round-trip time of 50 ms between the control service and controller using cable or fiber connections. We have successfully tested the system with an UMTS connection as well. Even with a GPRS connection, those five message could still be sent in less than a second. The control service performance was tested with software clients and was able to handle at least 1000 concurrent connections.

6 Conclusions and Future Work

During the development of our prototype, we have decided to go with the approach of using the WebSocket protocol in combination with TLS encryption and mutual authentication. We have considered it a feasible approach, because it is based on already standardized technologies and can be easily implemented, works with all kinds of IP connections and performs well in terms of bandwidth and latency. Network configuration is simple, as it does not have higher requirements than Internet access for web browsers. This, and the guaranteed end-to-end encryption will help gain the acceptance of potential customers. To integrate legacy devices, we have found that an SGW is a good solution. It is inexpensive, easy to install and requires little configuration.

Currently, we use our prototype system in a flagship project, where we use an SGW to remotely access the process control system of some water suppliers in Switzerland. Their pump strategy is optimized on a regular basis to reduce energy costs and provide tertiary control energy, while still maintaining a reliable water supply.

Development of our solution is still in progress. Future research includes further improvement of security and reliability, secure management of certificates and secure remote administration and maintenance of devices.

References

1. Federal Office for Information Security, Germany. Protection Profile for the Gateway of a Smart Metering System (Smart Meter Gateway PP). Version 1.2, 18 March 2013. https://www.bsi.bund.de/DE/Themen/SmartMeter/Schutzprofil_Gateway/schutzprofil_smart_meter_gateway_node.html
2. Federal Office for Information Security, Germany. Protection Profile for the Security Module of a Smart Meter Gateway (Security Module PP). Version 1.0, 18 March 2013.https://www.bsi.bund.de/DE/Themen/SmartMeter/Schutzprofil_Security/security_module_node.html
3. Li, T., Ren, J., Tang, X.: Secure wireless monitoring and control systems for smart grid and smart home. IEEE Wireless Commun. **19**(3), 66–73 (2012)
4. Pérez, J., Nurminen, J.K.: Electric vehicles communicating with WebSockets - measurements and estimations. In: 4th IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe), Copenhagen, 6–9 October 2013
5. Shuang, K., Shan, X., Sheng, Z., Zhu, C.: An efficient ZigBee-WebSocket based M2M environmental monitoring system. In: 2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing (2014)
6. ADS-Tec Big-Linx Remote Service Cloud using OpenVPN. http://www.ads-tec.de/industrial-it/cloud-big-linx/big-linx.html
7. The Open Smart Grid Protocol Alliance. http://www.osgp.org/
8. ETSI Approves Open Smart Grid Protocol (OSGP) for Grid Technologies. http://www.etsi.org/news-events/news/382-news-release-18-january-2012
9. Jovanovic, P., Neves, S.: Dumb Crypto: Practical Cryptanalysis of the Open Smart Grid Protocol. Cryptology ePrint Archive, Report 2015/428 (2015)
10. North American Electric Reliability Council: SQL slammer worm lessons learned for consideration by the electricity sector, 20 June 2003. http://www.utexas.edu/law/journals/tlr/sources/Issue90.1/Thompson/NAERCSlammerReport.pdf
11. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2., August 2008. http://tools.ietf.org/html/rfc5246
12. Hickson, I.: Server-Sent Events, W3C Candidate Recommendation, 11 December 2012. http://www.w3.org/TR/eventsource/
13. Baker, F., Meyer, D.: Internet Protocols for the Smart Grid, June 2011. http://tools.ietf.org/html/rfc6272
14. Fette, I., Melnikov, A.: The WebSocket Protocol, December 2011. http://tools.ietf.org/html/rfc6455
15. Hickson, I.: The WebSocket API, W3C Candidate Recommendation, 20 September 2012. http://www.w3.org/TR/websockets/
16. Pimentel, V., Nickerson, B.G.: Communicating and displaying real-time data with WebSocket. IEEE Internet Comput. **16**, 45–53 (2012)
17. Agarwal, S.: Real-time web application roadblock: performance penalty of HTML sockets. In: IEEE ICC 2012 - Communication QoS, Reliability and Modeling Symposium, pp. 1225–1229 (2012)
18. Lubbers, P.: How HTML5 Web Sockets Interact With Proxy Servers, 16 March 2010. http://www.infoq.com/articles/Web-Sockets-Proxy-Servers
19. Lubbers, P., Greco, F.: HTML5 Web Sockets: A Quantum Leap in Scalability for the Web, March 2010. http://soa.sys-con.com/node/1315473
20. The Modbus organization: The Modbus standard specification. http://www.modbus.org