Regaining Insight and Control on SMGW-based Secure Communication in Smart Grids

Jens Hiller
Communication and Distributed Systems
RWTH Aachen University
Aachen, Germany
hiller@comsys.rwth-aachen.de

Karsten Komanns
Communication and Distributed Systems
RWTH Aachen University
Aachen, Germany
komanns@comsys.rwth-aachen.de

Markus Dahlmanns
Communication and Distributed Systems
RWTH Aachen University
Aachen, Germany
dahlmanns@comsys.rwth-aachen.de

Klaus Wehrle
Communication and Distributed Systems
RWTH Aachen University
Aachen, Germany
wehrle@comsys.rwth-aachen.de

***Abstract*—Smart Grids require extensive communication to enable safe and stable energy supply in the age of decentralized and dynamic energy production and consumption. To protect the communication in this critical infrastructure, public authorities mandate smart meter gateways (SMGWs) to be in control of the communication security. To this end, the SMGW intercepts all inbound and outbound communication of its premise, e.g., a factory or smart home, and forwards it on secure channels that the SMGW established itself. However, using the SMGW as proxy, local devices can neither review the security of these remote connections established by the SMGW nor enforce higher security guarantees than established by the all in one configuration of the SMGW which does not allow for use case-specific security settings. We present mechanisms that enable local devices to regain this insight and control over the full connection, i.e., up to the final receiver, while retaining the SMGW's ability to ensure a suitable security level. Our evaluation shows modest computation and transmission overheads for this increased security in the critical smart grid infrastructure.**

***Index Terms*—Communication Security, Smart Meter Gateway, Smart Grid, TLS, Policy Language, Proxy, Internet of Things**

I. Introduction

Energy grids have to manage instabilities due to highly decentralized energy production in times of renewables as well as higher varying consumption caused by electrification of road traffic and smart homes. The high decentralization requires extensive communication between the stakeholders in smart grids to maintain a safe and stable energy supply.

This paper has received funding from the CONNECT project as part of the Electronic Components and Systems for European Leadership Joint Undertaking. Our work in the CONNECT project has received support from the European Unions Horizon 2020 research and innovation programme under grant agreement no. 737434 as well as the German Federal Ministry of Education and Research (BMBF) under funding reference no. 16ESE0154. This paper reflects only the authors views and the funding agencies are not responsible for any use that may be made of the information it contains.

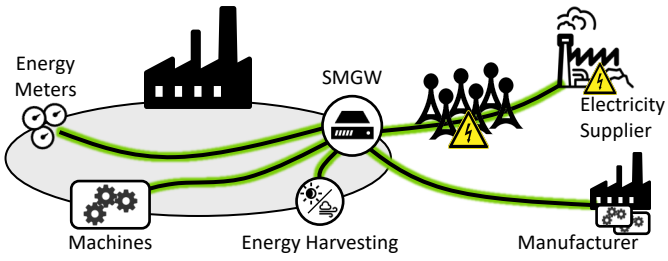

Fig. 1. Smart meter gateways (SMGWs) interconnect local and remote communication partners. Thereby, they act as TLS proxies, rendering devices unable to assess the connection security from the SMGW to the final target.

Specifically, in addition to today's adjustment across large-scale energy producers and grid operators, also the energy-harvesting and consuming machines of private or corporate end-consumers need to be controlled to realize fine-grained control on energy flows. Similarly, manufacturers must keep software up to date and retrieve long-time diagnostic data.

For this communication in smart grids, security and privacy is a fundamental requirement. Secure communication retains the privacy and business secrets of private and corporate end users, e.g., energy meter statistics or the existence of specific machines. Moreover, it prevents attackers from tampering with the critical smart grid infrastructure and is thus a key objective to guarantee a safe and stable energy supply in smart grids [1].

Smart meter gateways (SMGWs) can take an important role to ensure secure and privacy-preserving communication in smart grids. As illustrated in Figure 1, SMGWs interconnect end-user premises, e.g., factories or smart homes, with back-ends of major smart grid stakeholders, e.g., grid operators, energy producers, and manufacturers of energy consuming or harvesting machines. To leverage this position to ensure communication security and privacy in smart grids, the German Federal Office for Information Security (BSI) developed security guidelines [2] for any communication that

passes through an SMGW. Most importantly, they require any communication to use Transport Layer Security (TLS) [3] which is the prevalent security protocol used in the Internet [4]. However, typically TLS provides *end-to-end security*, i.e., it secures the communication from the sender up to the final receiver, not allowing any entity on the communication path to decrypt or modify data. In contrast, the BSI mandates the SMGW to control the security of in- and outbound communication by intercepting and forwarding communication data. To this end, local devices and the SMGW use *splitTLS* [5], i.e., the local device in the end-user premise establishes a TLS connection to the SMGW which in turn establishes a (second) TLS connection to the final receiver. Subsequently, the SMGW forwards data between both TLS connections acting as *proxy* for any communication between end-user premise and entities in the smart grid back-end. This way, the SMGW can control the security parameters of both connections and, e.g., remove personal identifiable information to protect user privacy [2].

However, splitTLS has been shown to risk communication security, e.g., by inadvertently reducing the security due to the use of weaker ciphers, too broad acceptance of authenticating certificates, or security problems of the implementation [6]. For the end-points, these problems remain hidden as they control only the connection security up to the proxy, but lack information on the security between proxy and final receiver. To overcome these limitations of splitTLS, we present *splitTLS-insight* and *splitTLS-control* which enable local devices to assess and control the security properties of the remote connection between SMGW and final receiver. While doing so, our approaches still retain the benefits of using the SMGW as proxy which enforces suitable communication security and privacy. More specifically, our contributions are as follows:

- We analyze the security of smart grid communication as proposed by the BSI and reveal the problems of its TLS-proxy approach which does not account for use case-specific security and privacy requirements.
- We first enable local devices to review the security properties of connections from the SMGW up to the final receiver by relaying handshake information. Second, we allow them to use policies to instruct the SMGW to increase the security for these connections.
- We implement and evaluate the performance of our design showing that we achieve increased communication security for smart grids with only modest runtime and transmission overheads.

II. Risks and Limits of TLS-Proxies in Smart Grids

Smart meter gateways (SMGWs) do not only aggregate and report energy meter data for billing purposes, but more importantly also interconnect any energy consumer or producer on the local end-user premise with back-end services in the smart grid (cf. Figure 1). These back-end services range from firmware update services offered by machine manufacturers up to reporting systems for energy consumption or production estimates that enable grid operators to offer a safe and stable energy supply. It is thus important to guarantee

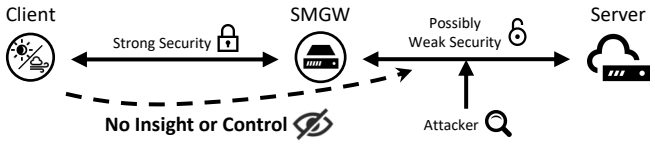

Fig. 2. Security guidelines of the BSI require the SMGW to act as proxy for TLS connections. This does not allow the client to assess or control the connection security between the SMGW and the server.

communication security such that attackers cannot influence the management of the critical smart grid infrastructure by manipulating communicated data.

To this end, the German Federal Office for Information Security (BSI) mandates the use of Transport Layer Security (TLS) [3] for all connections [2]. However, outdated implementations or weak TLS configurations could still pose a risk. Thus, to guarantee a suitable level of security for TLS connections, the BSI requires the SMGW to intercept all connections between devices in the local network and back-end services using *splitTLS*: To connect to a back-end service, a local device first establishes a secure TLS connection to the SMGW and instructs the SMGW to extend this connection up to the desired back-end service. Second, the SMGW establishes a TLS connection to the back-end service and forwards data between both ends. This way, the SMGW is in full control on the security of both connections and thus can enforce a suitable security level, especially for connections over the Internet.

However, using the SMGW as TLS-proxy also bears risks for connection security. The desired security level for communication depends on the sensitivity of data, e.g., personal identifiable information or critical control information require a higher security level than arguably less critical status information. As the SMGW lacks knowledge on the purpose of the communication, it handles any communication in the same way, i.e., regardless of the sensitivity of data. As methods which provide higher security typically decrease the performance due to more heavyweight computations, the SMGW uses a compromise that yields good security and acceptable performance. Thus, the uniform handling of all connections by the SMGW today does not take into account the use case knowledge only available at the (local) devices themselves.

As shown in Figure 2, local devices, however, cannot assess if the security properties of the remote TLS connection match the requirements of the use case as any information on the remote connection is hidden by the SMGW. Similarly, the BSI design does not enable local devices to modify the security of the remote connection to account for use case-specific security requirements. Specifically, local devices should be able to check for or request stronger security for the remote connection in the following ways: (i) For specifically sensitive data, local devices should be able to enforce the use of stronger encryption algorithms, e.g., longer key sizes for the Advanced Encryption Standard (AES), or future more secure ciphers. For data that requires long-term security such as personal information, the secure key negotiation should use public key mechanisms that also secure data in face of future availability

of quantum computers. (ii) Similarly, restricting the acceptable cryptographic hash algorithms, e.g., to the recent version SHA-3 [7], achieves higher security in the face of upcoming attacks against SHA-1 and expected future attacks against SHA-2. (iii) With respect to security against down-grade attacks, local devices should also be able to enforce the use of (optional) TLS extensions that prevent down-grade attacks or even require the use of TLS 1.3 with its enhanced security methods regarding down-grade attacks. (iv) Likewise, several features during authentication via certificates have influence on the connection security. Local clients should thus be able to verify certificates of the remote connection themselves to evaluate the corresponding security features. First, a local client should be able to verify the issuer of a certificate to assess the trust in this issuer. Problems regarding a large number of trusted issuers are already evident from the huge WebPKI [8]. Specifically, a single trusted but misbehaving issuer risks the security of the whole system. In contrast to the SMGW, the local client can have more detailed knowledge on which issuers can be trusted for a specific domain, e.g., when communicating with infrastructure of their own manufacturer. Second, *Signed Certificate Timestamps* (SCTs) ensure that a certificate is logged in a public certificate transparency log [9]. As these logs enable independent audition of certificates [9], [10], a certificate with SCT can be considered more secure which makes this a valuable feature that local devices should be able to check. (v) Finally, local devices should also be able to control the performance of the TLS connection, e.g., by requesting the use of the recent version TLS 1.3 with its enhanced performance features. Such higher performance can be required in disaster scenarios that require fast communication to counter physical risks for the infrastructure.

In summary, TLS-proxy functionality of SMGWs is useful to enforce a suitable security and privacy level for smart grid communication. However, it does not allow for a use case-specific higher connection security. To overcome this issue, we propose to enable local devices to review the security properties of the remote connection by relaying corresponding information, e.g., the server certificate and negotiated ciphers, and to allow them to employ policies to communicate their use case-specific security demands to the SMGW.

III. Insight and Control For Remote Connections

To tackle the discussed disadvantages of splitTLS, we enhance splitTLS by a two-fold approach that allows the local device (client) to *assess* and *control* the security of the remote connection that the SMGW established. First, *splitTLS-insight* enables the client to assess the established security. To this end, the SMGW reports properties of the connection such as the certificate or selected cipher suite to the client after both TLS connections have been established (Section III-A). Thus, the client can perform similar checks as if it would establish a TLS connection to the target directly, e.g., verify the certificate.

Second, we design *splitTLS-control* which uses policies sent by the client to the SMGW to inform the latter about (increased) security requirements for the connection to the target (Section III-B). With this mechanism in place, clients cannot only assess, but actively control the security of the remote connection established by the SMGW. For both approaches, we utilize the existing trust relationship between client and SMGW in smart grids, i.e., the local client trusts the SMGW to report correct values and to follow the provided policy. Optionally, we can alleviate this trust requirement when additionally adapting the final receiver such that it testifies the report of the SMGW as we detail in our security discussion.

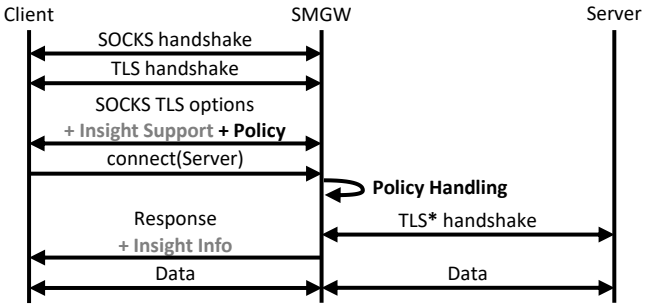

Fig. 3. SplitTLS with our *splitTLS-insight* (**gray bold**) and *splitTLS-control* (**black bold**) mechanisms. In splitTLS-control, the policy configures the TLS connection setup between SMGW and server (TLS*).

A. Regaining Insight on TLS Connections with Proxies

To enable local clients to assess the security of remote connections, we provide it with the information of the corresponding connection establishment. More specifically, we instruct the SMGW to relay important handshake information to the client when the SMGW successfully established the remote connection. As we show in Figure 3, this additional communication seamlessly integrates into the splitTLS connection setup. To establish a splitTLS connection with a remote target over the SMGW, the client first establishes a SOCKS [11] connection to the SMGW as it will later use the SOCKS protocol to inform the SMGW about the final target. To secure further communication (including the information on the target), client and SMGW subsequently establish their TLS connection. Following, client and SMGW exchange TLS-specific SOCKS options. Yet, these options only support, e.g., a specific negotiation for handling UDP traffic [12], but not the insight or control features which are required for the smart grid use case. With our splitTLS-insight design, we extend this exchange of options to negotiate the relaying of splitTLS information between local client and SMGW. To this end, the client signals in its options message that it expects the SMGW to relay remote connection setup information and the SMGW acknowledges its support in the response.

To then initiate the extension of the connection up to the final receiver, the client sends a *SOCKS-connect* message with target address information to the SMGW which in turn establishes a TLS connection to the target. The SMGW reports the success of this connection establishment back to the client within a *Response* message. We integrate the relaying of splitTLS-insight information into this message, i.e., we append, e.g., TLS version information, the negotiated cipher,

and all certificates presented by the target for its authentication. When the client receives this data, it performs similar checks as if it would establish a TLS connection itself. For example, it checks if the TLS version and cipher provide suitable security for its use case, checks the validity of the certificate using its own set of trusted root store certificates, and checks optional features such as inclusion of the certificate in a public certificate transparency log (by checking for an SCT). Only if all these checks pass, the client starts to exchange data over the established connection. Otherwise, it aborts the connection with a corresponding handshake failure error.

SplitTLS-insight allows the local client to assess the security of the remote connection on the splitTLS path. This enables the client to reject a connection that does not meet its security requirements. However, relaying information does not yet allow a local device to adapt the connection security to its needs. To also realize this control, we introduce a policy-based exchange of security demands for splitTLS (splitTLS-control).

B. Enabling Control on Security of Remote Connections

To account for use case-specific security demands, clients must be able to control the security of the remote connection. To this end, we allow them to instruct the SMGW to establish a connection with higher security guarantees. More specifically, we design clients to encode their requirements on the remote TLS connection security in a *policy* which they send to the SMGW during the setup of the splitTLS connection. The SMGW evaluates this policy and adapts its handshake behavior accordingly, e.g., only offering the most current TLS version or advertising support for only exceptionally strong ciphers.

The deployment as well as the update of policies is done by the machine manufacturer which can tie them to the different communication use cases and corresponding security demands. Optionally, the machine owner may supply its own policies to strengthen the security based on its own preferences. We first detail the policy-based splitTLS-control connection establishment, before we discuss requirements for a policy language to achieve high efficiency in the constrained smart grid environment, especially discussing the Compact Privacy Policy Language (CPPL) [13] as fitting candidate. Also, we detail the management, distribution and updates of policies.

1) Policy-based Connection Security Negotiation: To use policies for control of the remote connection security, we further extend the SOCKS options exchange as shown in Figure 3. If both, client and SMGW, signal support for splitTLS-control in the SOCKS TLS options exchange, the client proceeds by sending the policy to the SMGW. To comply with the policy, the SMGW adapts the configuration for the TLS connection to the target. For example, it removes ciphers that do not fulfill the policy from the supported ciphers list and only advertises TLS versions that are compatible with the policy. After the following TLS establishment for the remote connection, and as in the traditional splitTLS approach, the SMGW reports the success to the client which can start to exchange application data with the desired target.

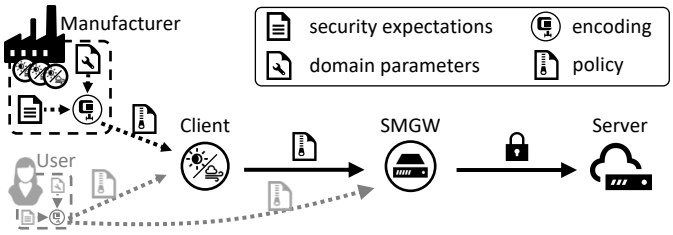

Fig. 4. **(CPPL) policy management in splitTLS-control:** Local devices send *use case-specific* policies provided by their manufacturer or user to the SMGW which configures the TLS connection to the backend service respecting the policy. Users may also deploy *device-specific* policies at the SMGW.

Due to the trust between local devices and the SMGW, the local device can rely on the correct usage of the policy by the SMGW. Optionally, a client can, however, still combine splitTLS-insight and splitTLS-control to also verify the security settings of the remote connection itself (cf. Figure 3).

2) Policy Management and Deployment Considerations: The efficient use of policies for remote connection security negotiation has several requirements. To cope with the limited processing capabilities of clients and SMGWs, especially considering the high number of connections handled by the latter, (i) the processing overheads for policy use must be small. Furthermore, as many connections have to be established, also (ii) the additional bandwidth required for the policy transmission should be limited. Finally, (iii) stakeholders, i.e., machine manufacturers or device owners, must be able to adapt policies to new security developments.

To address these requirements, we select a policy language that provides good performance in environments with constrained resources and propose a management process for policies in smart grid environments. More specifically, we propose to use the *Compact Privacy Policy Language* (CPPL) [13] to realize control on remote connection security. This policy language was originally proposed to negotiate expectations of users on the handling of their data by cloud services. It provides a very compact representation of policies for their efficient transmission and thus adds only marginal bandwidth overhead to connection establishment. To this end, it uses a domain specific compression mechanism to transform a human readable representation of the policy into an efficient binary format that is used for transmission. Second, based on this compressed policy representation, CPPL offers a very efficient evaluation mechanism that combines the policy with technically supported features, e.g., the capabilities of a cloud computing server, to derive instructions for the data handling.

To use CPPL in smart grids, i.e., to negotiate remote connection security properties to the SMGW, we designed corresponding domain parameters for the domain-specific compression. These domain parameters enable us to create policies that specify TLS connection properties such as allowed ciphers, certificate issuers, or the mandatory presence of an SCT (cf. Section II). As we show in Figure 4, these policies are specified by the trust-worthy device manufacturer and already compressed at its side such that devices already obtain the compressed policy. Thus, resource-constrained de-

vices save the overhead for compression and can directly send the compressed policy to the SMGW during connection establishment (cf. Section III-B1). To derive suitable TLS connection settings, the SMGW combines the capabilities of its TLS implementation with the received policy using CPPL's policy evaluation mechanism which yields instructions for the TLS configuration. Subsequently, the SMGW leverages these instructions to set TLS handshake options as well as callbacks that check server-provided information, e.g., the server certificate, with respect to conformance with the policy.

Optionally, also the device owner who has control over the SMGW can add its own policies to control security for specific devices. To this end, the SMGW can handle *device-specific* policies for connected devices as provided by the device owner. Notably, we store manufacturer controlled policies at the client itself to flexibly adapt the policy based on the connection's use case. To also enable device owners to deploy *use case-specific* policies, the device itself needs to provide an interface to install these policies as the SMGW stays oblivious regarding the actual use case of a connection. Finally, to update a once deployed policy, a manufacturer or device owner can supply an updated policy to its devices or the SMGW.

IV. Evaluation

To show the feasibility of splitTLS-insight and splitTLS-control, we implemented it based on CPPL [13] as well as OpenSSL and deployed it on two NXP MCIMX7SABRE (2x ARM Cortex-A7@1 GHz, 1 GB RAM), one acting as local device and one acting as SMGW. Furthermore, we deployed a server based on OpenSSL on a full grade desktop machine (Intel i5@3.3GHz, 16GB RAM). Using this testbed, we evaluate the connection establishment runtime. Additionally, we analyze packet sizes to assess the transmission overhead.

A. TLS Insight and Control is Feasible for Client and SMGW

To analyze the performance, we compare the connection establishment runtime of splitTLS-insight and splitTLS-control with the traditional BSI splitTLS approach and show the results in Figure 5. We report on the arithmetic mean of 100 measurements and show 99% confidence intervals.

Considering the connection setup and negotiation between the local device and the SMGW (C→SMGW), the runtime is dominated by the TLS establishment (202 ms) including corresponding certificate checks (45 ms) which both do not differ between all approaches. The negotiation of support and use of splitTLS-insight and splitTLS-control does not add overhead as it is piggybacked on already existing messages. Only the subsequent transmission of the policy in case of splitTLS-control introduces a marginal overhead of 1 ms.

When the client instructed the SMGW to connect to the server, splitTLS-control first requires the SMGW to process the policy to configure TLS accordingly. For our smart grid specific policy, this processing requires 12 ms and is thus again marginal compared to the inevitable overhead of TLS connection setup. SplitTLS-insight does not add any additional processing in this phase (compared to the traditional system).

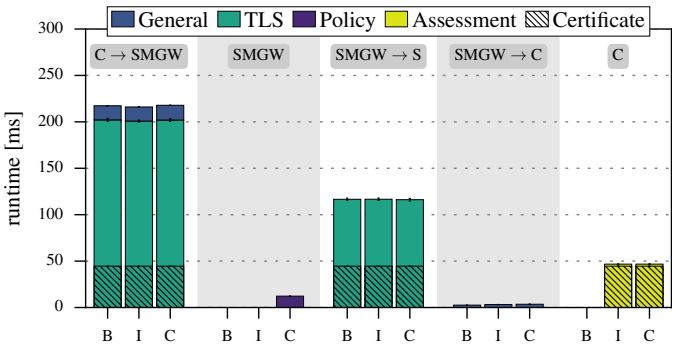

Fig. 5. Connection setup runtime for traditional splitTLS (B) compared to splitTLS-insight (I) and splitTLS-control (C). SplitTLS-insight introduces only small overhead at the client (certificate check). SplitTLS-control introduces small overhead at SMGWs as they have to process the policy.

Next, the SMGW establishes the TLS connection to the server (SMGW→S). SplitTLS-insight does not alter this exchange thus achieving the same performance as in the traditional case. For splitTLS-control, the actual overhead depends on the changed security settings and added TLS features as requested by the policy. As splitTLS-control only enables or configures already existing TLS mechanisms, we still achieve typical TLS connection setup times for the remote connection and possible overhead is inevitable to achieve the desired security level for the specific use case of the connection.

After setting up the remote TLS connection, the SMGW informs the client about the success (SMGW→C). For splitTLS-insight, it also sends the information on the remote connection that enable the client to assess the corresponding security level. This results in a negligible overhead of 1 ms (also observed for splitTLS-control as we activated the (optional) splitTLS-insight features for our splitTLS-control measurements).

Before sending application data, the client analyzes the received insight information when splitTLS-insight is enabled (and optionally for splitTLS-control). These checks are dominated by the certificate validation (45 ms) which is in line with the corresponding runtime during the (unchanged) TLS handshake with the SMGW. After all, they enable the client to check if the required security level for its use case is met.

The server uses the standard TLS implementation and thus only encounters overhead corresponding to stronger TLS security as configured by policies. Similarly, our approaches do not alter the exchange of application payload except for traditional TLS configuration changes such as the use of stronger ciphers. In summary, the runtime overhead for splitTLS-insight and splitTLS-control is well-manageable for all peers including local devices and SMGWs, enabling smart grids to profit from both, a guaranteed minimal connection security level enforced by the SMGW and a use case-specific security control.

B. Low Transmission Overhead

Our mechanisms realize insight and control for splitTLS with additional information exchanges, namely negotiation of support, relaying of handshake information and providing policies. The negotiation of insight and control support requires

only two additional bytes, each, as we integrate it into the SOCKS TLS options exchange (cf. Figure 3). For splitTLS-insight, the transmission overhead totals to feasible 1971 byte, dominated by the certificate information. Furthermore, the policy transmission for splitTLS-control requires additional 320 bytes (depending on the policy size) to allow clients to increase the security of the remote TLS connection.

Summing up our evaluation, our approaches allow clients the former unavailable assessment and control of the remote connection security, thus enabling a use case-specific higher connection security for the critical smart grid infrastructure.

V. Security Discussion

Most importantly regarding security, our approaches only allow for a (use case-specific) higher security level than traditionally established by the SMGW. To this end, the SMGW does not allow policies to decrease the security level.

Similar to traditional splitTLS, our design does not achieve end-to-end security. Specifically, the SMGW can read and alter any communication. However, this is part of the requirements set forth by the BSI which requires the SMGW to be able to inspect traffic and enforce a minimal level of security [2]. Importantly, the SMGW is a highly regulated device that is generally trusted by the participants in the smart grid.

Optionally, also the server can send a signed report on the connection security to the client such that the client can verify the correctness of the properties reported by the SMGW. However, due to the above noted trust relationship between SMGW and local devices, we deem this as unnecessary in the regulated smart grid scenario. Still, our approaches are necessary to enable *use case-specific* and *device-specific* configurations of security levels for which the SMGW lacks any knowledge in the traditional approach.

Considering the connection security, we note that our design uses only standardized TLS features and thus does not alter the security features that TLS provides. After all, our design enables clients to assess the former invisible security of the remote connection and even allows for the former unavailable use of use case-specific and device-specific security levels.

VI. Related Work

One line of research focuses on integrating middlebox functionality into the end-to-end security protocol TLS [14], [15]. However, these approaches do not support middleboxes to be in full control of the connection. Instead, the communication end-points have full control on what content middleboxes can read or modify. This renders the SMGW unable to enforce minimal connection security and thus does not comply with the requirements as set forth by administrative bodies for smart grid communication. Furthermore, most of these approaches require changes to the server, while our mechanisms only require changes to local devices and SMGWs.

We employ policies to control the security of the remote connections. The use of policy languages to efficiently negotiate rules or instructions with remote entities was already proposed for other domains, e.g., data handling, resource management, network configuration, or failure handling in cloud computing [13], [16]. Henze et al. performed an extensive analysis of existing policies with respect to requirements in the cloud computing domain [13]. Our decision to use CPPL as policy language in smart grids is based on this analysis.

VII. Conclusion

Secure communication in smart grids guarantees a safe and stable energy supply as it prevents attackers from tampering with the critical smart grid infrastructure. To this end, smart meter gateways (SMGWs) ensure a suitable security for all connections by acting as proxy for all inbound and outgoing communication of an end-user premise, e.g., a smart home or factory. However, currently this proxy-approach is incapable of addressing use case-specific security requirements, e.g., higher security demands for personally identifiable information, and does not allow local devices to assess the connection security between the SMGW and the final communication partner.

To tackle these shortcomings, we propose mechanisms that provide insight and control for TLS connections with proxies. Specifically, we enable local devices to assess the security of the remote connection by relaying corresponding information from the (trusted) SMGW. Our policy-based approach further allows local devices, which know about the use case-specific security demands, to instruct the SMGW to use a specific (stronger) security configuration for its connection to the final communication partner. Our evaluation shows that our design achieves insight and control for smart grid communication with modest runtime and transmission overheads for connection establishment. Thus, also resource-constrained devices in the smart grid can easily adopt our approaches.

Our solution still allows the SMGW to guarantee security but in addition enables clients to assess the full connection security and makes a use case-specific security control possible. Thus, we overall realize a better tailored approach to secure communication in the critical smart grid infrastructure.

References

[1] S. Soltan, P. Mittal, and H. V. Poor, "BlackIoT: IoT Botnet of High Wattage Devices Can Disrupt the Power Grid," in *USENIX Security*, 2018.

[2] German Federal Office for Information Security, "Technische Richtlinie BSI TR-03109-1, Version 1.0," Tech. Rep., 2013, german.

[3] E. Rescorla and T. Dierks, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246, 2008.

[4] D. Naylor *et al.*, "The Cost of the "S" in HTTPS," in *CoNEXT*, 2014.

[5] J. Jarmoc and D. Unit, "SSL/TLS interception proxies and transitive trust," in *Black Hat Europe*, 2012.

[6] X. de Carné de Carnavalet and M. Mannan, "Killed by proxy: Analyzing client-end TLS interception software," in *NDSS*, 2016.

[7] NIST Computer Security Division, "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions," Tech. Rep., 2014.

[8] R. Holz *et al.*, "The SSL Landscape: A Thorough Analysis of the x.509 PKI Using Active and Passive Measurements," in *ACM IMC*, 2011.

[9] B. Laurie, A. Langley, and E. Kasper, "Certificate Transparency," RFC 6962, 2013.

[10] Q. Scheitle *et al.*, "A First Look at Certification Authority Authorization (CAA)," *SIGCOMM CCR*, vol. 48, no. 2, May 2018.

[11] M. D. Leech, "SOCKS Protocol Version 5," RFC 1928, 1996.

[12] M. VanHeyningen, "Secure Sockets Layer for SOCKS Version 5," IETF, Internet-Draft draft-ietf-aft-socks-ssl-00, 1997, work in Progress.

[13] M. Henze *et al.*, "CPPL: Compact Privacy Policy Language," in *ACM Workshop on Privacy in the Electronic Society*, 2016.
[14] H. Lee *et al.*, "maTLS: How to Make TLS middlebox-aware?" in *NDSS*, 2018.
[15] D. Naylor *et al.*, "Multi-Context TLS (mcTLS): Enabling Secure In-Network Functionality in TLS," in *ACM SIGCOMM*, 2015.
[16] J. Hiller *et al.*, "Giving Customers Control Over Their Data: Integrating a Policy Language into the Cloud," in *IEEE IC2E*, 2018.