Towards trusted metering in the smart grid

Georgios Karopoulos*, Christos Xenakis†, Stefano Tennina‡ and Stefanos Evangelopoulos§
*National and Kapodistrian University of Athens, Panepistimiopolis, 15784 Ilissia, Greece
Email: gkarop@di.uoa.gr
†University of Piraeus, Karaoli and Dimitriou 80, 18534 Piraeus, Greece
E-mail: xenakis@unipi.gr
‡WEST Aquila srl, S.S. 17 Ovest 36, 67100 L'Aquila, Italy
Email: tennina@westaquila.com
§LINK Technologies S.A., Balkan Center Thermi, 57001 Thessaloniki, Greece
Email: stevenevs@three-pi.com

***Abstract*—With the evolution of the smart grid, most homes will be equipped with smart meters that support consumption reading, demand response and applications requiring two-way communications. In this context, security is a key aspect for smart grid adoption, especially since customers will have physical access to smart meters installed in their premises. Customers will be able to modify the software, firmware or hardware of the smart meter for their own benefit; additionally, remote attackers can mount cyber-attacks against smart meters. To address these challenges we have designed and evaluated a trusted computing environment for smart meters that provides secure storage of cryptographic keys and sensitive data, remote attestation, and protection to sensitive smart grid applications. The performance evaluation shows that our proposal is efficient and can be applied to embedded devices, like smart meters.**

I. Introduction

A smart meter constitutes a basic building block of the smart grid that is deployed in customers homes. These devices collect power metering data, while they can additionally receive commands from energy operators in order to perform different actions, e.g., reduce power usage to avoid a blackout in the broader area. This communication architecture between smart meters and utility companies is formally known as Advanced Metering Infrastructure (AMI), and provides companies with real-time data about power consumption; at the same time, it allows customers to make informed choices about energy usage based on the price at the time of use.

Security is particularly important in this context. As with general purpose Information and Communication Technology (ICT) systems, which are subject to numerous security attacks, smart grids inevitably inherit a number of security threats and vulnerabilities. Especially for the AMI, the following issues are considered of major importance. Unauthorized physical access to smart meters can have as effect the retrieval of sensitive data and encryption keys, as well as unauthorised hardware modification by adversaries; it can also lead to electricity theft by the final customer. Also, local or remote attackers might try to modify application code running on smart meters in order to get access to sensitive data or alter consumption readings (electricity theft). For all these cases, existing protection approaches have not been designed for unattended operation.

Summarizing from the above it is essential for a complete solution to:

- protect cryptographic material;
- provide assurances of running code integrity;
- safeguard critical application operations.

In order to achieve these goals, we need a mechanism to ensure that such unattended devices will have a predictable behaviour avoiding data leakages; trusted computing provides a suitable environment for our purpose.

The idea of hardware security modules in the smart grid is not new [1]; however, limited work related to the smart grid has been done [2]. Generally, two prevalent specifications for trusted computing currently exist: the Trusted Platform Module (TPM) [3] and the Trusted Execution Environment (TEE) [4]. The TPM is a co-processor, which provides basic cryptographic capabilities (like random number generation, hashing and encryption), which can be used to implement hardware-based security services such as device authentication, integrity measurement, and remote attestation. The most significant limitations of the TPM platform include [5], [6]: (a) the need for a separate module increases the cost of a device, (b) it offers no protection against runtime attacks, (c) it relies on the assumption that a TPM cannot be tampered, (d) it is not suitable for mobile and embedded devices, and (e) in case of compromise, the hardware module must be physically replaced. TEE can support configurations like the TPM above; in addition, it can support arrangements where a separate hardware module is not used, utilising two virtual processing cores with different privileges: a normal one for applications, and a secure one for security-sensitive code execution.

The contribution of this paper is an integrated trusted environment for smart meters that meets the main challenges mentioned above by providing: (a) secure storage of cryptographic keys and sensitive data, (b) remote attestation for ensuring running code integrity, and (c) protection to sensitive smart grid applications. A trusted computing platform can manage cryptographic keys, authenticate the configuration of a platform (i.e., attestation), support asymmetric key generation, encryption, decryption, signing, random number generation, and hashing using the crypto-processor (i.e. an internal processor dedicated to cryptographic operations). Additionally, the internal memory provides storage for sensitive data. A key concept in trusted computing is remote attestation [7], where modifications in one entity can be detected by remote authorised entities. For our proposal, we have selected the TEE as a trusted computing environment without using a separate hardware module, due to the limitations of the TPM.

The paper is organised as follows. Section II overviews related work. Next, in Section III, our proposal is presented, while Section IV provides details about its performance. Finally, Section V concludes the paper.

II. Related work

In [8], the authors present an attested metering architecture based on virtualisation application isolation, integrity measurement and protection, and mandatory access control. The proposed architecture is based on TPM, having the disadvantages discussed above; the TPM is used for remote attestation and billing, not taking into account critical operations like authentication and key management. The authors present a working prototype as well, based on commodity hardware and emulation; however, they neither utilize it for running any scenarios nor do they provide experimental results.

In [9], the authors present a concept for smart metering taking into account both security and privacy. The proposed solution provides assurance of proper smart meter operation through remote attestation; it also supports billing, bearing similarities with the previous proposal. This proposal is also based on TPM, presenting the aforementioned limitations of TPM platforms. Moreover, in the cited paper no implementation and no experimental results are provided to assess the performance of the solution.

In [10] a framework for assured and trusted real-time communication for smart grids is presented by employing: (a) a TPM interface for built-in non-migratable trust, (b) a Kerberos multicast authentication service, and (c) a real-time attribute-based access control system. The TPM interface will enforce the integrity of communicated data (by releasing appropriate message authentication keys), and depend on the application to provide confidentiality of communicated data. This proposal, however, focuses on smart grid substations and not on limited-resource smart meters.

Similarly, Kuntze *et al.* [11] describe a system for using non-migratable TPM keys for device identification in smart grid environments. The device authentication is done via a challenge-response protocol that involves the signature of a non-migratable asymmetric TPM signing key. Such a key is generated by the TPM itself and the private part never leaves the TPM. Apart from using a TPM platform, the downside of this system is that it does not perform any verification when the key is used.

The authors of [2] present a system based on trusted computing to protect device private keys in the smart grid context. In their implementation the TEE is provided by the open-source Flicker project [12] and has two main use cases: (a) the generation of the device private key within the TEE and (b) a handshake of the standard TLS protocol. The downside of this proposal is that it does not describe any remote attestation mechanism which is essential for smart meters that are physically exposed to anyone.

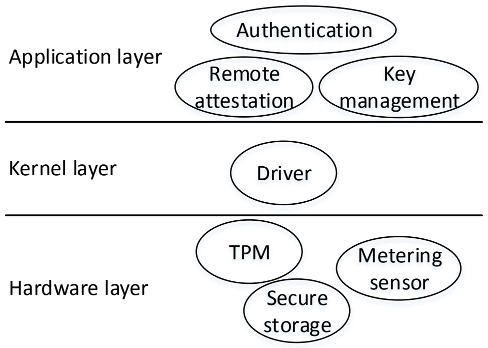

Fig. 1. Trusted meter reference architecture

III. Trusted computing architecture

One of the main reasons for adopting trusted computing in the smart grid, is to protect smart meters from customers or other attackers that would want to manipulate them. With this requirement in mind, we designed a system to: (a) protect device private keys and sensitive data through secure storage, (b) endorse remote attestation, and (c) secure critical applications, like authentication and key management. The proposed smart meter architecture is depicted in Fig. 1.

A. Secure storage

The TEE supports secure storage of data and keys using functions similar to `fopen`, `fclose`, `read`, `write`, and `seek`. Like applications store and read data, a Trusted Application puts object data into files. The individual file size is exposed to the normal world, but the object contents and identifier are encrypted. Thus, while the normal world is able to modify objects residing in the secure storage, modifications are detected and reported to the application with the error code `TEE_ERROR_CORRUPT_OBJECT`. This way, TEE provides integrity checking, while it allows application developers a safe place to store data, cryptographic keys, sensitive configuration or user private data. The secure storage procedure follows well established techniques like [13].

B. Remote attestation

The remote attestation procedure enables a smart meter to verify that another smart meter operates an un-tampered firmware or software/application versions. As a prerequisite, each smart meter should store:

- a 2048 bit RSA public/private key pair (PU_M, PR_M);
- the utility's 2048 bit RSA public key (PU_U);
- a SHA256 hash of PU_U;
- its own digital certificate C_M signed by the utility; and

- firmware or software/application executables (i.e., the binary code of the firmware or applications like authentication and key management) in its storage.

Apart from the utility, other smart grid entities can be considered, like the network operator that provides connectivity to smart grid devices, the smart meter manufacturer or the grid operator. For the sake of simplicity, we assume that all devices run the same versions of firmware or software/applications. The node initiating a remote attestation procedure is referred as the attester (denoted as ATT), while the recipient of the attestation request is the target (denoted as TG). The attestation procedure (illustrated in Fig. 2) includes the following steps:

1) The ATT node issues an attestation request to a target node after a random waiting time. The waiting time interval is relevant to the application; for example, in an aggregation scenario the waiting time interval can be between 0 and 15 minutes which is the interval of high frequency measurements [14].
2) Once the waiting timer expires, ATT randomly selects one of its neighboring smart grid entities as the target (TG) and generates a 128bit random nonce N using a Random Number Generator (RNG). It then transmits an attestation request message (i.e., Challenge) to the TG, which contains ATT's digital certificate C_A and N (see Equation 1 below), as well as generates a SHA256 hash (H_R) of the relevant binary executable concatenated with N. This is the hash that will be used as a reference and compared with the response sent by TG later on.

$$Challenge : C_A, N \tag{1}$$

3) Upon the reception of the attestation request, TG first verifies the authenticity of ATT's public key PU_A, using the accompanied certificate C_A and the utility's public key PU_U. If the public key is authentic, it generates a SHA256 hash of the binary executable (B) residing in TG's storage, concatenated with N (see Equation 2 below), and a 128bit AES session key K.

$$H_C = SHA256(B|N) \tag{2}$$

4) The TG generates an attestation reply message (i.e., Response), which includes: (i) the TG's certificate C_T; (ii) the hash of the binary executable and N (i.e., H_C) encrypted by the session key K; and finally, (iii) the generated session key K as well as the received N encrypted, first, by the TG's private key PR_T and, subsequently, by the ATT's public key PU_A.

$$Response : C_T, K(H_C), PU_A\{PR_T(K, N)\} \tag{3}$$

5) Upon the reception of the Response, ATT first verifies the authenticity of TG's public key PU_T, using the accompanied certificate C_T and the utility's public key PU_U. If it is authentic, it uses its private key PR_A and TG's public key PU_T to decrypt the session key K and N. It then compares the received N with the previously generated one, in order to verify that the reply message is legitimate and not a replay attack. Finally, it uses the session key K to decrypt H_C and compares it with its own generated hash H_R. If TG runs an untampered version of the considered executable, then the two hashes should match, considering that: (i) all nodes run the same executable version, (ii) the executable is compiled for the same OS, and (iii) both nodes used the same N value in the computation of the hashes.

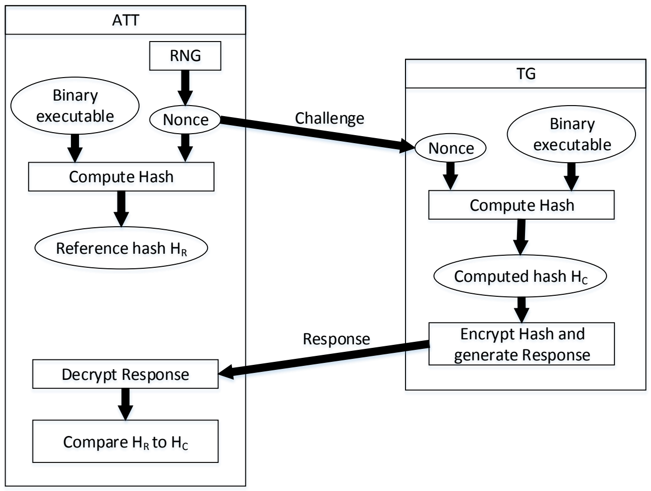

Fig. 2. Remote attestation procedure

Since nodes ATT and TG have already established a shared secret session key K, any subsequent attestation requests between them can be performed using this key. The ATT may submit an attestation request to the TG, by generating and transmitting N, while TG's reply encompasses the hash of the executable concatenated with N and encrypted with the session key (i.e., $K\{H_C, N\}$).

C. Critical applications

As a critical application example, here we consider the authentication and key management protocol for the smart grid proposed in [15]. In the aforementioned solution authentication is performed based on digital certificates; the secret keys of these certificates will be stored in the TEE and will never leave this protected environment. For efficient key management, a Distributed Hash Table (DHT) based on Chord [16] is implemented for fast certificate look-up operations. We use this DHT to store certificates in a distributed fashion for scalability and in order to cope with network outages and intermitted communications. Each node stores locally a low number of certificates during joining the DHT network; when a node D requests a certificate C, the network directs D to the node that is responsible for C. Moreover, each certificate can be signed by multiple endorsers so that the compromisation of a single certificate (like the Certification Authority's) cannot render invalid the majority or even all smart grid's certificates. Certificate signing is also a critical operation that will have to be performed internally in the TEE. Other critical operations

TABLE I
BASIC CRYPTOGRAPHIC OPERATIONS DELAY

Operation	Mean delay (ms)
Creation of a 2048-bit RSA key pair	220.75
Generation of a 128bit AES key or nonce	5.68
SHA256 hash of $(B\|N)$ (55KB)	7.97
Encryption of a SHA256 hash with a 128-bit AES key	7.12
Encryption of an AES key: $PU_n\{PR_m(K\|N)\}$	325.8
Decryption of the above: $PRK_n\{PU_m(K\|N)\}$	226.82
Verification of a digital signature or certificate	39.73
Decryption of an encrypted SHA256 hash with the AES key (128 bits)	4.64
Signing of a digital certificate	38.47

TABLE II
DELAY OF COMPLEX OPERATIONS

Application	Mean delay (ms)
Secure storage	220.75
Remote attestation	617.76
Authentication	78.2
Key management (node join)	1422.35

that need to be executed inside the TEE include certificate verification, hashing and key generation.

IV. PERFORMANCE EVALUATION

For our proof of concept implementation, the TEE is provided by the open-source Open-TEE project [17]. Open-TEE is a virtual trusted execution environment and its main advantage is that it is implemented based on GlobalPlatform's specifications [4] and any applications developed for Open-TEE will operate on any TEE following these specifications. In this section, we have implemented and measured the basic cryptographic functions that are performed by the TEE (e.g., hashing, encryption); the results are shown in Table I. Based on this list of basic functions we are able to compute the total delays for complex operations executed in applications like remote attestation, authentication and key management; the respective results are presented in Table II. The experiments were run on a Raspberry Pi with Open-TEE installed, in order to approximate the hardware of an embedded device like a smart meter. Its hardware specifications include: a single core 700MHz CPU, and 512MB @ 400MHz SDRAM.

Secure storage: It involves the operations described in [13]. The calculation of its performance is counted by creating a 2048-bit RSA key pair inside the Open-TEE; thus, the delay of secure storage is 220.75 ms.

Remote attestation: The remote attestation procedure follows the steps presented in Fig. 2. Its main operations involve the following:

- Generation of a 128bit nonce using a random number generator;
- SHA256 hash of a binary executable together with a nonce $(B|N)$. For the key management application, the binary has a size of 55 KBs;
- Encryption of a SHA256 hash with the AES key;
- Encryption of the AES key first with a RSA private key and then with a public key: $PU_n\{PR_m(K|N)\}$;
- Decryption of the above: $PR_n\{PU_m(K|N)\}$;
- Verification of a public key using the certificate and the CA certificate; and
- Decryption of the above encrypted SHA256 hash with the AES key

The total delay of a full round of a remote attestation procedure is 617.76 ms.

Authentication and key management: The solution presented in Section III-C, mainly requires the following operations to be executed for authentication:

- Signing of a digital certificate (OpenPGP format); and
- Verification of a digital certificate (OpenPGP format)

Thus, the total delay of an authentication session is 78.2 ms.

Key management involves the two aforementioned operations as well as the bootstrapping phase described in [15] that comprises two distinct steps: (a) a node join operation on the Chord level, and (b) signing of the digital certificates that will be assigned to the node by Chord.

We first measured the mean delay of a single node joining a Chord ring. The simulation was made with OMNeT++ [18] and OverSim [19] and the results are presented in Fig. 3. From these results it is evident that the majority of delays are below 2 seconds; we also calculated the average delay, which is 1.23 seconds.

Then, we calculated the number of certificates that should be signed by the joining node. Assuming we have the same number of nodes and hash keys ($N = K$), each node in the Chord ring will be responsible for a low number of certificates as shown in Table III. The calculations were based on Theorem 1 from [16]; the theorem defines the maximum number of hash keys that each node is responsible for:

$$\frac{(1+\epsilon)\times K}{N} \tag{4}$$

where ϵ has an upper bound of $O(logN)$ when consistent hashing is implemented. The signing delay comprises the signing of all the digital certificates the node is responsible for; from Table I the delay for a single certificate is 38.47 ms.

For computing the total delay presented in Table II, we assume a network of 5,000 nodes. In this case, the total maximum delay is $1,230+5\times 38.47 = 1422.35$ ms. However, the delay is not expected to increase much, even with large network sizes; for example, assuming we have a network of 500,000,000 nodes and that the average Chord join delay will not increase substantially, the total node join delay in the key management solution will be in the order of 2 seconds.

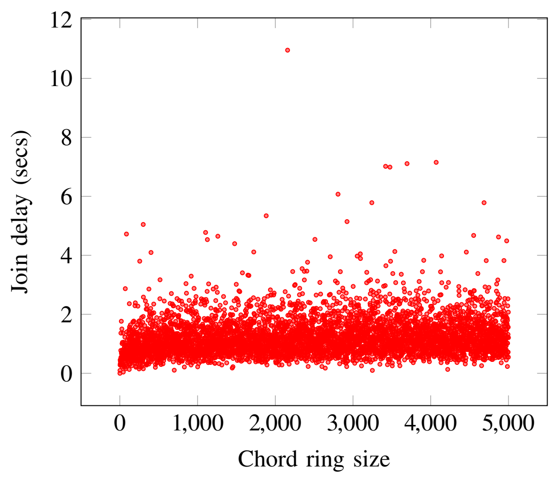

Fig. 3. Chord node join delay

TABLE III
CERTIFICATE SIGNING DELAY ON SOMA-S

Nodes	Certificates	Signing delay (ms)
50	3	119.19
5,000	5	198.65
500,000,000	10	397.30

V. Conclusion

Secure operation is a critical factor for smart grid evolution. Each smart meter belonging to the smart grid will have private cryptographic keys used for critical operations like authentication and remote attestation; these keys must be protected against theft or misuse. Although various solutions based on trusted computing exist for protecting keys and sensitive data, most of these bear additional cost due to separate hardware needed, and so are not suitable for use in a large scale in the smart grid environment.

We proposed a system for protecting cryptographic keys, sensitive data and critical operations in the context of smart grid, using trusted computing technologies. We chose the TEE platform due to lower hardware cost, as well as extensibility, since the same code can be used with a separate hardware TEE module if it is necessary in the future. In our system, private keys never leave the TEE; moreover, remote attestation and other critical applications are executed inside the TEE. We have experimentally evaluated the performance of the proposed system and demonstrated that it is efficient even in the presence of a large number of smart meters, while satisfying the defined security requirements. Overall, the system presented here is a promising approach towards protecting smart meters against internal or external adversaries and achieving security in the smart grid.

Acknowledgement

This research has been funded by the European Commission as part of the SMART-NRG project (FP7-PEOPLE-2013-IAPP Grant number 612294).

References

[1] N. Kuntze, C. Rudolph, M. Cupelli, J. Liu, and A. Monti, “Trust infrastructures for future energy networks,” in *Power and Energy Society General Meeting, 2010 IEEE*, July 2010, pp. 1–7.

[2] A. J. Paverd and A. P. Martin, “Hardware security for device authentication in the smart grid,” in *Smart Grid Security*. Springer, 2012, pp. 72–84.

[3] Trusted Computing Group, “TPM mobile with trusted execution environment for comprehensive mobile device security,” Whitepaper, 2012.

[4] GlobalPlatform, “TEE System Architecture, version 1.0,” December 2011, last accessed 4-10-2016. [Online]. Available: https://www.globalplatform.org/specificationsdevice.asp

[5] B. Kauer, “Oslo: Improving the security of trusted computing.” in *USENIX Security*, vol. 7, 2007.

[6] M. Strasser and H. Stamer, “A software-based trusted platform module emulator,” in *International Conference on Trusted Computing*. Springer, 2008, pp. 33–47.

[7] G. Coker, J. Guttman, P. Loscocco, A. Herzog, J. Millen, B. O’Hanlon, J. Ramsdell, A. Segall, J. Sheehy, and B. Sniffen, “Principles of remote attestation,” *International Journal of Information Security*, vol. 10, no. 2, pp. 63–81, apr 2011. [Online]. Available: http://dx.doi.org/10.1007/s10207-011-0124-7

[8] M. LeMay, G. Gross, C. A. Gunter, and S. Garg, “Unified architecture for large-scale attested metering,” in *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, Jan. 2007, p. 115.

[9] R. Petrlic, “A privacy-preserving concept for smart grids,” *Sicherheit in vernetzten Systemen*, vol. 18, pp. B1–B14, 2010.

[10] M. Burmester, J. Lawrence, D. Guidry, S. Easton, S. Ty, X. Liu, X. Yuan, and J. Jenkins, “Towards a secure electricity grid,” in *Intelligent Sensors, Sensor Networks and Information Processing, 2013 IEEE Eighth International Conference on*. IEEE, 2013, pp. 374–379.

[11] N. Kuntze, C. Rudolph, I. Bente, J. Vieweg, and J. von Helden, “Interoperable device identification in smart-grid environments,” in *2011 IEEE Power and Energy Society General Meeting*, July 2011, pp. 1–7.

[12] J. M. McCune, B. J. Parno, A. Perrig, M. K. Reiter, and H. Isozaki, “Flicker: An execution infrastructure for tcb minimization,” in *ACM SIGOPS Operating Systems Review*, vol. 42, no. 4. ACM, 2008, pp. 315–328.

[13] H. Löhr, A.-R. Sadeghi, and M. Winandy, “Patterns for secure boot and secure storage in computer systems,” in *Availability, Reliability, and Security, 2010. ARES’10 International Conference on*. IEEE, 2010, pp. 569–573.

[14] C. Efthymiou and G. Kalogridis, “Smart grid privacy via anonymization of smart metering data,” in *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*. IEEE, 2010, pp. 238–243.

[15] F. F. Demertzis, G. Karopoulos, C. Xenakis, and A. Colarieti, “Self-organised key management for the smart grid,” in *Ad-hoc, Mobile, and Wireless Networks*. Springer, 2015, pp. 303–316.

[16] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, “Chord: a scalable peer-to-peer lookup protocol for internet applications,” *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17–32, Feb. 2003.

[17] B. McGillion, T. Dettenborn, T. Nyman, and N. Asokan, “Open-TEE – an open virtual trusted execution environment,” in *Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA*, ser. TRUSTCOM ’15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 400–407.

[18] A. Varga and R. Hornig, “An overview of the omnet++ simulation environment,” in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*. Brussels, Belgium: ICST, 2008, pp. 60:1–60:10.

[19] I. Baumgart, B. Heep, and S. Krause, “OverSim: A flexible overlay network simulation framework,” in *Proceedings of 10th IEEE Global Internet Symposium (GI ’07) in conjunction with IEEE INFOCOM 2007, Anchorage, AK, USA*, May 2007, pp. 79–84.