Smart Meter Data Security: Integrating ElGamal and Zero-Knowledge Proofs for Privacy-Preserving Billing

A research and development project realized at Aarhus University in the fall semester of 2023.

Alexander Stæhr Johansen
Department of Computer Engineering
Aarhus University
Aarhus, Denmark
201905865@post.au.dk

Henrik Tambo Buhl
Department of Computer Engineering
Aarhus University
Aarhus, Denmark
201905590@post.au.dk

***Abstract*—Smart meters have transformed energy management, but their data collection poses privacy risks. This paper proposes a privacy-preserving billing protocol using Elliptic Curve (EC) ElGamal encryption and Zero-Knowledge Proofs (ZKPs), including Bulletproofs for filtering consumption data, and a Zero-Knowledge Proof of Equivalence (ZKPe) for data integrity. A Proof-of-Concept (PoC) demonstrates the protocol's feasibility, focusing on encryption and computational efficiency. The successful PoC implementation of EC ElGamal and Bulletproof modules marks progress, with range-based filtering and ZKPe development as future objectives. This research advances cryptographic solutions for secure, private smart metering.**

***Index Terms*—Smart Meter Privacy, Data Integrity, Energy Consumption Analytics, Cryptographic Security in Smart Grids, Elliptic Curve Cryptography, Zero-Knowledge Proofs, EC ElGamal Encryption, Bulletproofs, Chaum-Pedersen Protocol.**

I. Introduction

In the evolving landscape of energy management, the role of smart meters has become increasingly prominent. These devices are essential in the modernization of the electrical grid, by enabling more efficient energy distribution and management. Unlike traditional meters, which offer limited and delayed feedback, smart meters provide accurate, near real-time insights into energy usage, which has multiple transformative properties [1].

Firstly, precise, timely data from smart meters enhances energy management by allowing utilities to balance supply with demand, reduce waste, and quickly adapt to usage changes [2]. This, in turn, leads to more accurate consumer billing and improved operational efficiency. Additionally, smart meters are crucial in grid management and predictive maintenance, using data patterns to foresee and prevent system issues, thus promoting a more resilient, cost-effective, and sustainable energy grid [3].

Secondly, smart meters provide consumers with detailed, immediate insights, encouraging energy conservation and more informed usage decision-making. This awareness is important for global sustainability efforts, helping to reduce overall consumption and the carbon footprint. Moreover, smart meters aid in integrating renewable energy sources into the grid, especially in regions with high renewable adoption, facilitating a smoother transition to sustainable energy systems [4].

However, as Samet Tonyali [5] highlights, while smart meters' detailed consumption data will be important for energy management, it poses significant privacy risks by revealing intimate household routines. This data, indicating when residents are likely to be awake, asleep, or away from home, can be used to construct a detailed profile of a household's daily life, including specific activities.

Such detailed insights raise serious privacy concerns [6]. The idea of outsiders accessing personal lifestyle information can be unsettling, and this data

can be valuable to marketers for targeted advertising or to criminals for planning crimes. Additionally, there is a risk of misuse by corporations or governments, potentially leading to surveillance and privacy invasion [7]. This situation underscores the need for a balanced approach in smart meter data handling, ensuring both security and respect for individual privacy.

A. Research objectives and contributions

This paper, 'Smart Meter Data Security: Integrating ElGamal and Zero-Knowledge Proofs (ZKPs) for Privacy-Preserving Billing' addresses privacy concerns in smart meters through a distinct combination of three cryptographic schemes. We propose a system where smart meters use Elliptic Curve (EC) ElGamal encryption [8, 9] for secure data storage, transmission, processing (confidentiality), and generate Zero-Knowledge range Proofs [10], specifically Bulletproofs [11], for abnormal transaction filtering. Furthermore, we exploit the mathematical structural similarity between an EC ElGamal ciphertext and the EC Pedersen Commitment, upon which the Bulletproof is built, to ensure the integrity of the ciphertext by creating a Zero-Knowledge Proof of Equivalence (ZKPe) proving that the Bulletproof indeed proves properties of the encrypted value. As such, the utility provider can reliably process bills via the control center using ElGamal's homomorphic properties on the Bitcoin curve, enabling billing on encrypted data without compromising consumer privacy, but it can also differentiate between normal and abnormal consumption data while preventing malicious intent. The research will conclude with an evaluation of this system's practical applicability.

In summary, the key contribution of this study includes the proposal of a protocol that integrates:

- An EC ElGamal scheme for secure storage, transmission, processing, and privacy-preserving billing.
- A ZKP (Bulletproof) for abnormal observation filtering.
- A ZKPe for ensuring the integrity of the encrypted smart meter consumption data.

B. Report structure

The report will start by giving the reader an introduction to related academic works, specifically those using homomorphic encryption and range proofs in smart meters, and then elaborate on how our research positions itself among these. We then present our system model in section III, where we elaborate on our system architecture, threat, and trust models. Following this, in section IV we present the theory fundamental to this project, specifically the EC version of ElGamal, and ZKPs. In section V we present our proposed solution and a proof-of-concept (PoC) of the key modules in our solution, and in section VI we test this PoC. Finally, in section VII we give some conclusive remarks, and in section VIII we elaborate on aspects of the solution that need to be improved in the near future.

II. Related Works

In this section, we classify work related to this research. In section II-A we summarize works discussing privacy-preserving computations made on smart meter data. In section II-B, we present works related to implementing range proofs in smart meters, and, finally, in section II-C, we motivate the position of this paper among the related works.

A. Homomorphic Encryption

Homomorphic Encryption (HE) is a cryptographic technique that allows computations to be performed on encrypted data, producing an encrypted result which, when decrypted, matches the result of operations performed on the plaintext. In the context of smart meters, HE enables the analysis of energy consumption data while preserving the privacy of consumers.

The Paillier cryptosystem is widely used in this domain for facilitating privacy-preserving data aggregation using its additive homomorphic properties. For example, Erkin and Tsudik [12] applied the Paillier cryptosystem for private computation of spatial and temporal power consumption on a single-household level. Li et al. [13] extended this application by aggregating encrypted data from multiple households, enabling utility providers to calculate the total energy usage of an entire neighborhood.

Advancing the field, Dong et al. [14] proposed an ElGamal-based scheme for aggregating large-scale smart grid data. ElGamal is known for its multiplicative, which by modification can be converted to additive, homomorphic properties, but it also offers a lower computational overhead compared to the Paillier encryption. Furthermore, it is more resilient

against sophisticated attacks as it is based on the Discrete Logarithm Problem (DLP) instead of the Decisional Composite Residuosity Problem (DCRP) [8].

Further advancing the field, an adaptation of ElGamal using Elliptic Curve Cryptography (ECC) was introduced [15], significantly reducing the computational overhead further, thus enabling data aggregation in large-scale smart grids.

B. Range Proofs

Range proofs are cryptographic techniques used for verifying that a secret value, such as a numerical reading from a smart meter, lies within a specific range without revealing the actual value [10]. In the context of smart meters, this technique is useful for filtering abnormal readings, as it allows for the detection of readings that lie outside of a statistically defined expected range.

An example of this usage is presented in Ni et al. [16] article, where they adopt range-proof-like methods in their approach to differentially private smart metering, which includes fault tolerance and range-based filtering.

However, traditional range proofs pose challenges due to computational complexity and inefficiency, especially with large datasets [11]. Such inefficiency can lead to increased processing times and excessive resource consumption, making them less feasible for extensive smart grid operations.

To overcome these challenges, Bulletproofs [11] have been developed as an advanced form of non-interactive zero-knowledge (NIZK) proof. Bulletproofs are shorter and faster to verify compared to conventional range proofs and do not require a trusted setup, thus rendering them more practical for applications like smart metering. Unfortunately, not much research has been conducted in the domain of Bulletproofs integrated with smart meters.

C. Elliptic Curve ElGamal and Bulletproofs

As such, our research strives to propose an integration of ECC with the ElGamal encryption scheme, coupled with the implementation of Bulletproofs. The reasoning is that the EC ElGamal scheme is adept at handling the encryption of substantial volumes of meter data, ensuring consumer privacy in a manner conducive to the demands of large-scale utility systems. Concurrently, Bulletproofs offer an efficient mechanism for the verification of data integrity within encrypted datasets, markedly reducing the computational resources and time required for such processes.

Furthermore, we enhance the security architecture by employing a Zero-Knowledge Proof of Equivalence (ZKPe) using the Chaum-Pedersen protocol. We do this by leveraging the structural similarities between the EC Pedersen Commitment, on which the Bulletproof is based, and the EC ElGamal ciphertext. By doing so, we ensure the integrity of the encrypted, transmitted data. As such, the ZKPe enables us to assert that the same data encrypted by EC ElGamal is committed in the EC Pedersen Commitment, without revealing the actual data. This approach enhances the system's resilience against potential tampering, while also upholding the confidentiality that is important for consumer privacy.

III. System Model

In this section, we will present the system architecture, the threat model, and the trust model, respectively, of the proposed system.

A. System Architecture

This section presents the system architecture encompassing Smart Meters, the Control Center (operated by the utility company), and the Trusted Authority. Be advised that we have abstracted networking hardware away, as the project is solely focused on the privacy-preserving protocol, as defined in section I-A. Overall, the system architecture can be visualized as in figure 1.

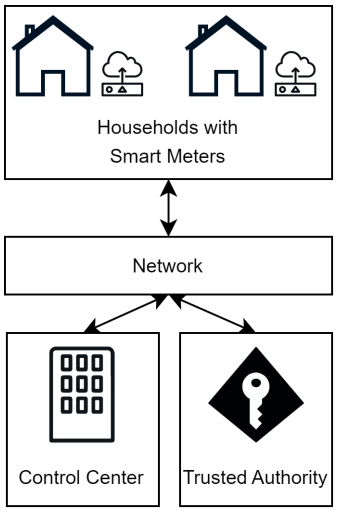

Fig. 1: Visualization of the system architecture.

In the proposed system, each household has a smart meter, which collects energy consumption data. The Smart Meter is responsible for creating an EC Pedersen Commitment, thus committing to the value, a Bulletproof, encryption of the measured data using ECElGamal (ensuring confidentiality), and a ZKPe using the encrypted value and the EC Pedersen Commitment (integrity).

The Control Center is responsible for verifying the ZKPe (ensuring authenticity and integrity of the measurement), the Bulletproof (which is used for filtering abnormal measurements), and calculating the energy consumption bill using the ciphertext and returning the bill to the smart meter.

Finally, the Trusted Authority oversees the entire cryptographic infrastructure. It is responsible for the secure generation, distribution, and management of cryptographic keys, playing a crucial role in maintaining the system's security and integrity.

B. Threat Model

In our Threat Model, we establish a set of fundamental assumptions that highlight the potential threats of the smart metering system. These assumptions are derived from concrete threat scenarios.

For example, a significant concern within our system is the possibility of both independent tampering and collusion between internal entities, such as smart meter users and control center personnel. For instance, a household might attempt to reduce its energy bill by tampering with the meter's recorded data. Conversely, there might be situations where control center staff, manipulate data to inflate charges, for personal gain or due to external pressures. In a more complex collusion scenario, the same household could bribe a staff member at the control center to alter the data in their favor, resulting in a reduced bill. Such collusion would not only amplify the risk but also make detection more challenging, as it involves insiders with a deep understanding of the system. These scenarios underscore the system's vulnerability to internal manipulation, highlighting the necessity for encryption. However, in our implementation, we apply HE, and in a more complicated scenario, we could imagine that the same threat actors might exploit the malleability of HE. HE, while powerful for allowing computations on encrypted data, inherently possesses a property where the encrypted data can be manipulated in a controlled manner. This malleability could be exploited by savvy threat actors to alter the outcome of computations performed on encrypted values. For instance, they might manipulate the encrypted energy consumption data in such a way that, when decrypted, it reflects a lower/higher usage than occurred, depending on the motivations of the threat actor.

In summary, we make the following two assumptions about our threat actors' activities:

1) Assumption of Data Tampering: We assume the risk of internal or external entities attempting to manipulate the energy consumption data, thus we encrypt it.
2) Assumption of HE Malleability Exploitation: We assume that there is a threat of sophisticated attackers exploiting the malleability of HE to manipulate ciphertexts, which can compromise data integrity. Therefore we use Bulletproofs and ZKPe to ensure confidentiality and integrity.

C. Trust Model

Similarly to our threat model, we establish a set of assumptions underlining the trust in our system, based on a set of scenarios.

Firstly, there is an inherent trust in the smart meter's capability to measure energy consumption accurately. This trust extends to the hardware and software components of the smart meter, assuming that they function as intended without any inherent flaws or vulnerabilities that could lead to incorrect measurements. However, as we implement range-based filtering using Bulletproofs to distinguish abnormal from normal measurements, we significantly reduce the risk of erroneous data, and, in turn, the amount of trust placed in this capability.

Secondly, we trust the utility provider's capability to operate smart meters correctly. This includes the proper installation, maintenance, and regular updating of the meters. The utility provider is also trusted to respond appropriately to any identified issues, whether they are technical faults or security breaches.

Finally, we trust the trusted authority to securely generate, distribute, and manage cryptographic keys. This involves creating cryptographic keys with secure methods, ensuring their safe transmission to smart meters and the control center, and overseeing

their entire lifecycle. The trusted authority is responsible for safeguarding these keys against unauthorized access, handling their renewal and revocation as needed.

In summary, we make the following assumptions about our system.

1) Assumption of trust in a smart meter's capability to measure correctly.
2) Assumption of trust in a utility provider's capability to operate smart meters correctly.
3) Assumption of trust in the trusted authority's ability to securely generate, distribute, and manage cryptographic keys correctly.

IV. Background

In this section we present theory underpinning the proposed solution. In section IV-A we introduce the EC ElGamal scheme, while we present ZKPs, specifically the Bulletproof and the ZKPe, in section IV-B.

A. Elliptic Curve ElGamal

The ElGamal encryption system, presented by Taher ElGamal in 1985, represents an asymmetric cryptographic scheme based on the discrete logarithm problem. As previously mentioned, traditional ElGamal encryption, while robust, is not sufficiently optimized for IoT devices, leading to computational inefficiencies and significant data overhead. In response to these challenges, the adaptation of ElGamal encryption to ECC, known as ElGamal EC ElGamal, provides a valuable solution.

EC ElGamal increases efficiency by using the mathematical properties of elliptic curves over finite fields [9]. The key lies in the smaller key sizes required for equivalent levels of security compared to traditional ElGamal. For instance, a 256-bit key in an EC system offers comparable security to a 3072-bit key in a classical RSA or ElGamal system [17]. This dramatic reduction in key size translates to lower data requirements and faster computational processes.

Moreover, the efficiency of EC ElGamal is significantly augmented through its employment of point multiplication on elliptic curves, a procedure that proves to be more computationally efficient than the modular exponentiation integral to traditional ElGamal [18]. This efficiency is further amplified by the choice of particular elliptic curves, notably curve25519 or secp256k1, which are known for their balance of security and performance [19].

In addition to these efficiency improvements, EC ElGamal also offers enhanced security. The scheme's security foundation is the complexity of the Elliptic Curve Discrete Logarithm Problem (ECDLP). Unlike the discrete logarithm problem in classical finite field cryptography, the ECDLP remains a significant challenge to solve, even with advanced algorithms like the Pollard rho method [20]. This complexity renders EC ElGamal more resistant to certain cryptographic attacks, such as the index calculus method, which is more effective against traditional ElGamal [21].

In algorithm 1 to 3 we will explain the key generation, encryption, and decryption protocols for EC ElGamal, inspired by the work of Lawrence C. Washington in his book "Elliptic Curves, Number Theory, and Cryptography, Second Edition" [9].

Algorithm 1: Key Generation for Elliptic Curve ElGamal

Input: Elliptic curve E over finite field F_q, Point P on E
Output: Public Key (E, F_q, P, B), Private Key s
$s \leftarrow z \in \mathbb{Z}$
$B \leftarrow sP$
return (E, F_q, P, B), s

The Key Generation algorithm for EC ElGamal in algorithm 1 starts with an elliptic curve E defined over a finite field $\mathbb{F}_q$, and a point P on this curve. The core of the process involves generating the private key s and the public key components. The private key s is randomly selected from the set of integers (denoted by $\mathbb{Z}$). The public key is a combination of the elliptic curve E, the finite field $\mathbb{F}_q$, the point P, and a new point B. The point B is computed by multiplying the private key s with the point P on the elliptic curve. This multiplication utilizes elliptic curve arithmetic, which underpins the security of the algorithm. The output of this algorithm is a pair of keys: the public key $(E, \mathbb{F}_q, P, B)$ and the private key s, which are used in subsequent encryption and decryption processes.

The EC ElGamal Encryption algorithm, presented in algorithm 2 is responsible for securing a message using the public key. It takes as input the public key components $(E, \mathbb{F}_q, P, B)$ and the message M,

Algorithm 2: Elliptic Curve ElGamal Encryption

Input: $Publickey(E, F_q, P, B)$, Point $M \in E(F_q)$
Output: Ciphertext (M_1, M_2)
$k \leftarrow z \in \mathbb{Z}$
$M_1 \leftarrow kP$
$M_2 \leftarrow M + kB$
return (M_1, M_2)

which is represented as a point on the elliptic curve $E(\mathbb{F}_q)$. The algorithm begins by selecting a random integer k from the set of integers $\mathbb{Z}$. This random integer is important to ensure the uniqueness and security of each encrypted message. The algorithm then computes two new points, M_1 and M_2, using elliptic curve arithmetic. M_1 is obtained by multiplying k with the point P, and M_2 is derived by adding the message M to the product of k and B. The pair (M_1, M_2) forms the ciphertext, which is the encrypted form of the message.

Algorithm 3: Elliptic Curve ElGamal Decryption

Input: Ciphertext (M_1, M_2), Private key s
Output: Decrypted message M
$M \leftarrow M_2 - sM_1$.
return M

The Elliptic Curve ElGamal Decryption algorithm, as visualized in algorithm 3, is designed to retrieve the original message from its encrypted form. The decryption process requires the ciphertext (M_1, M_2) and the private key s as inputs. The algorithm operates by performing elliptic curve arithmetic on these inputs to reverse the encryption process. Specifically, it computes the original message M by subtracting the product of the private key s and the point M_1 from the point M_2. The result of this subtraction is the original message M, now successfully decrypted. However, also the ECC version of ElGamal has an additive homomorphic property, as defined in algorithm 4.

In algorithm 4 we can see how addition of plaintexts is allowed while working in the encrypted domain. This algorithm takes as input two ciphertexts: $(M_{1,1}, M_{2,1})$ and $(M_{1,2}, M_{2,2})$. The algorithm then computes the new ciphertext $(M_{1,3}, M_{2,3})$ as follows. The first part of the new ciphertext, $M_{1,3}$,

Algorithm 4: Homomorphic Addition for Elliptic Curve ElGamal

Input: Ciphertexts $(M_{1,1}, M_{2,1})$ and $(M_{1,2}, M_{2,2})$
Output: Ciphertext $(M_{1,3}, M_{2,3})$
$M_{1,3} \leftarrow M_{1,1} + M_{1,2}$
$M_{2,3} \leftarrow M_{2,1} + M_{2,2}$
return $(M_{1,3}, M_{2,3})$

is obtained by adding the first components of the input ciphertexts, $M_{1,1}$ and $M_{1,2}$, using elliptic curve point addition. Similarly, the second part, $M_{2,3}$, is calculated by adding the second components of the input ciphertexts, $M_{2,1}$ and $M_{2,2}$. The output $(M_{1,3}, M_{2,3})$ is the ciphertext that corresponds to the product of the plaintexts of the original input ciphertexts.

B. Zero-Knowledge Proofs

Zero-Knowledge Proofs is a concept in cryptography, enabling one party (the prover) to prove to another party (the verifier) that a given statement is true, without conveying any information apart from the fact that the statement is indeed true. Introduced in the 1980s by Shafi Goldwasser, Silvio Micali, and Charles Rackoff [10], ZKPs hinge on the notion that it is possible to prove the knowledge of a value or a secret without revealing the secret itself.

The essence of a ZKP can be illustrated through the allegory of the "Cave of Secrets." Imagine a cave in a circular shape with an entrance and a secret door that opens only with a secret word. If a person (the prover) knows the secret word, they can prove this to an observer (the verifier) by entering the cave and emerging from the secret door without revealing the word itself [22].

ZKPs can be broadly categorized into two main types: Interactive and Non-Interactive.

In Interactive ZKPs the prover and verifier engage in a back-and-forth communication. The verifier poses challenges, and the prover responds to these challenges to establish the truth of the statement [22, 23]. On the other hand, Non-Interactive ZKPs do not require such back-and-forth communication. Here the proof is constructed by the prover and sent to the verifier without any further interaction, making them more suitable for scenarios where constant interaction is not feasible [24].

1) Range proof and Bulletproof: Among the various forms of ZKPs, range proofs stand out for their ability to prove that a value falls within a specific range defined as $[0, 2^n - 1]$ without revealing the value itself[11]. An optimized version of range proofs is known as Bulletproofs. Bulletproofs streamline the efficiency of proving statements, particularly in terms of proof size and verification time [11].

The structure of Bulletproofs can be understood in three distinct phases: the commitment phase, the proof generation phase, and the verification phase [11]. In the following paragraphs, we will explain each of these phases in detail using finite field arithmetic, as defined in the original article [11], but our PoC is based on the available library, implemented on elliptic curve arithmetic [25].

The commitment phase is characterized by the creation of Pedersen Commitments, which serve to hide and bind values, ensuring their integrity and confidentiality. In algorithm 5 it creates the commitments A and S which are based on the secret vectors a_L, a_R and the random vectors s_L, s_R. The vectors a_L and a_R are related to the secret value v in the sense that a_L is a binary vector representation of v and a_R is the flipped representation of a_L also known as the two's complement. The commitment A is a cryptographic construct that locks in the secret vectors $\mathbf{a}_L, \mathbf{a}_R$ without revealing them. It uses the group generators $\mathbf{g}, \mathbf{h}$ and the blinding factor γ to ensure the values are kept secret. The same is done with regards to S and s_L, s_R [11].

The proof generation phase involves creating the proof π based on the commitments and various cryptographic operations. Here it generates challenges y and z based on a statement st and the commitments A and S. The prover then computes two vectors $\mathbf{l}(x)$ and $\mathbf{r}(x)$ based on the initial vectors, the challenges, and the random vectors. The inner product of these vectors forms a polynomial $t(x)$, from which the coefficients t_1 and t_2 are extracted. These coefficients encode information about the original vectors in relation to the challenges and are used to create commitments T_1 and T_2 locking in the values while keeping them hidden. A new challenge x_0 is generated from the hash of st, T_1 and T_2, setting the stage for the inner product argument which produces vectors L and R and updates the challenge x, compressing the proof and maintaining its integrity [11]. Finally, the proof π is compiled, containing all necessary elements for a verifier to check the validity of the prover's statement without revealing any sensitive information[11].

Algorithm 5: Bulletproof Proof Generation

Input: Statement $\mathbf{st}$, Secret value v, Range n, Group generators $\mathbf{g}, \mathbf{h} \in \mathbb{G}^n$, Generator $H \in \mathbb{G}$

Output: Bulletproof π

$\alpha, \rho, \tau_1, \tau_2 \xleftarrow{\$} \mathbb{Z}_p$

$\mathbf{a}_L \leftarrow$ binary vector representation of v such that $\langle \mathbf{a}_L, 2^n \rangle = v$

$\mathbf{a}_R \leftarrow \mathbf{a}_L - \mathbf{1}^n$

$A \leftarrow h^\alpha \prod_{i=1}^n g_i^{a_{L_i}} h_i^{a_{R_i}}$

$\mathbf{s}_L, \mathbf{s}_R \xleftarrow{\$} \mathbb{Z}_p^n$

$S \leftarrow h^\rho \prod_{i=1}^n g_i^{s_{L_i}} h_i^{s_{R_i}}$

$y \leftarrow \text{Hash}(st, A, S)$

$z \leftarrow \text{Hash}(A, S, y)$

$\mathbf{l}(x) \leftarrow \mathbf{a}_L - z \cdot \mathbf{1}^n + x \cdot \mathbf{s}_L$

$\mathbf{r}(x) \leftarrow \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n + x \cdot \mathbf{s}_R) + z^2 \cdot \mathbf{2}^n$

$t(x) = \langle \mathbf{l}(x), \mathbf{r}(x) \rangle$

$t_1 =$ coefficient of x in $t(x)$

$t_2 =$ coefficient of x^2 in $t(x)$

$T_1 \leftarrow g^{t_1} h^{\tau_1}$

$T_2 \leftarrow g^{t_2} h^{\tau_2}$

$x_0 \leftarrow \text{Hash}(st, T_1, T_2)$

$L, R, x, \langle \mathbf{l}', \mathbf{r}' \rangle \leftarrow$ $\mathbf{IP}(st, l(x), r(x), g, h, H, x_0)$

$\pi \leftarrow (A, S, T_1, T_2, x, y, z, L, R, \langle \mathbf{l}', \mathbf{r}' \rangle)$

return π

Algorithm 6 is the part of the proof generation phase where the inner product is calculated. It starts with preparing the inputs for the calculations to be made, the most notable here being that of n' which gets halved at the start of each round. The core of the algorithm is an iterative process that involves halving the vectors and updating the challenge x [11]. The algorithm splits the vectors $\mathbf{l}'$ and $\mathbf{r}'$ into left and right halves in each iteration, effectively halving their size. New vectors L and R are computed using the halves of $\mathbf{g}'$, $\mathbf{h}'$, and the inner product of the corresponding halves of $\mathbf{l}'$ and $\mathbf{r}'$, along with the distinct generator H [11].

A new challenge x is generated in each iteration by hashing the vector polynomials L and R together with the statement st (st is explained later on and its making is depicted in 8). This challenge is then

used to update the vectors $\mathbf{g}'$, $\mathbf{h}'$, $\mathbf{l}'$, and $\mathbf{r}'$ for the subsequent round[11].

Upon completion of the iterative process, the algorithm outputs the final vectors L and R, the updated challenge x, and the final inner product $\langle \mathbf{l}', \mathbf{r}' \rangle$[11].

Algorithm 6: Inner Product Argument in Bulletproofs

Input: Statement st, Vectors $\mathbf{l}(x), \mathbf{r}(x)$, Group generators $\mathbf{g}, \mathbf{h} \in \mathbb{G}^n$, Generator $H \in \mathbb{G}$, Challenge x_0
Output: Vectors L, R, updated challenge x and Final Inner Product $\langle \mathbf{l}', \mathbf{r}' \rangle$

$n' \leftarrow n$
$\mathbf{g}' \leftarrow \mathbf{g}$
$\mathbf{h}' \leftarrow \mathbf{h}$
$\mathbf{l}' \leftarrow \mathbf{l}(x)$
$\mathbf{r}' \leftarrow \mathbf{r}(x)$
while $n' > 1$ **do**
  $n' \leftarrow n'/2$
  $\mathbf{l}_{\text{left}} \leftarrow \mathbf{l}'[1 : n']$
  $\mathbf{l}_{\text{right}} \leftarrow \mathbf{l}'[n' + 1 : 2n']$
  $\mathbf{r}_{\text{left}} \leftarrow \mathbf{r}'[1 : n']$
  $\mathbf{r}_{\text{right}} \leftarrow \mathbf{r}'[n' + 1 : 2n']$
  $L \leftarrow \prod_{i=1}^{n'} g_i'^{l_{\text{right},i}} \cdot h_i'^{r_{\text{left},i}} \cdot H^{\langle \mathbf{l}_{\text{right}}, \mathbf{r}_{\text{left}} \rangle}$
  $R \leftarrow \prod_{i=1}^{n'} g_i'^{l_{\text{left},i}} \cdot h_i'^{r_{\text{right},i}} \cdot H^{\langle \mathbf{l}_{\text{left}}, \mathbf{r}_{\text{right}} \rangle}$
  $x \leftarrow \text{Hash}(st, L, R)$
  $\mathbf{g}' \leftarrow \mathbf{g}'[1 : n'] \cdot x + \mathbf{g}'[n' + 1 : 2n'] \cdot x^{-1}$
  $\mathbf{h}' \leftarrow \mathbf{h}'[1 : n'] \cdot x^{-1} + \mathbf{h}'[n' + 1 : 2n'] \cdot x$
  $\mathbf{l}' \leftarrow \mathbf{l}_{\text{left}} \cdot x + \mathbf{l}_{\text{right}} \cdot x^{-1}$
  $\mathbf{r}' \leftarrow \mathbf{r}_{\text{left}} \cdot x^{-1} + \mathbf{r}_{\text{right}} \cdot x$
end
return $L, R, x, \langle \mathbf{l}', \mathbf{r}' \rangle$

Algorithm 7 depicts how a proof is verified. The verification of a proof is done by first reconstructing/recalculating the commitments T_1, T_2, the challenges x, y, z and the inner product l', r'. Thereafter it checks if its calculations matchup with what it got from the prover, if so then it returns $TRUE$ otherwise $FALSE$ [11].

For the sake of rendering Bulletproofs non-interactive, a method known as the Fiat-Shamir heuristic is employed. This technique involves creating a statement st as depicted in algorithm 8, which is a tuple consisting of a range n and a Pedersen Commitment V to the secret value v. Afterwards then the statement is used when hashing as otherwise an adversary can prove invalid statements [11]. The non-interactive approach is feasible because, in the Fiat-Shamir transformation, the "honest verifier's

Algorithm 7: Bulletproof Verification

Input: Proof $\pi = (A, S, T_1, T_2, x, y, z, L, R, \langle \mathbf{l}', \mathbf{r}' \rangle)$, Statement st, Group generators $\mathbf{g}, \mathbf{h} \in \mathbb{G}^n$, Generator $H \in \mathbb{G}$
Output: Validity of the proof: True or False

$x' \leftarrow \text{Hash}(st, L, R)$
$y' \leftarrow \text{Hash}(st, A, S)$
$z' \leftarrow \text{Hash}(A, S, y')$
if $y \neq y'$ ***or*** $z \neq z'$ ***or*** $x \neq x'$ **then**
  return *False*
end
The verifier uses y, z, A, S to deduce values l_1, l_2, r_1, r_2 such that:
$T_1' = g^{l_1} h^{r_1}$
$T_2' = g^{l_2} h^{r_2}$
if $T_1 \neq T_1'$ ***or*** $T_2 \neq T_2'$ **then**
  return *False*
end
Compute $\langle \mathbf{l}', \mathbf{r}' \rangle$ using L, R, x
if *Computed* $\langle \mathbf{l}', \mathbf{r}' \rangle \neq$ *provided* $\langle \mathbf{l}', \mathbf{r}' \rangle$ **then**
  return *False*
end
return *True*

messages" – which are typically random challenges in an interactive setting – are replaced with deterministic elements derived from a hash function, using elements from $\mathbb{Z}_p^*$ [11].

Algorithm 8: Formation of Statement st in Bulletproofs

Input: Secret value v, Blinding factor r, Range n, Group generators $g, h \in \mathbb{G}$
Output: Statement st, Commitment V

$V \leftarrow g^v h^r$
$st \leftarrow \{V, n\}$
return st, V

2) Proof of Equivalence: A Zero-Knowledge Proof of Equivalence (ZKPe) is a cryptographic protocol used to demonstrate that two separate pieces of data, although different in form or representation, are equivalent in value [26]. This type of proof is particularly useful for verifying the equality of information without revealing the actual data, which is crucial in maintaining privacy [10]. In our application, ZKPe is employed to uphold data integrity by showing that the amount of energy consumption encrypted in an EC ElGamal ciphertext [8] is equivalent to the value

committed in an EC Pedersen Commitment [27], thereby ensuring the integrity of the Bulletproof [28].

An implementation of a ZKPe is a Non-Interactive Zero-Knowledge Proof of Equivalence (NIZKPe). Unlike interactive Zero-Knowledge Proofs (ZKPs) [10], NIZKPe requires only a single message exchange, significantly enhancing efficiency in communication-limited scenarios.

In the Non-Interactive Chaum-Pedersen Protocol for Elliptic Curves (Algorithm 9), the prover begins by selecting a random scalar c from the set $\mathbb{Z}_q$. The prover then computes two points on the elliptic curve: $U = cP$ and $V = cB$, where P is the base point and B is the prover's public key point. The challenge e is derived using a cryptographic hash function that incorporates U, V, the elliptic curve E, the finite field $\mathbb{F}_q$, the transaction identifier TxID, and the timestamp TS. The prover calculates the response $s = c + xe \mod q$ and sends the tuple (U, V, e, s) to the verifier. Upon receipt, the verifier checks if e is correctly computed and verifies two elliptic curve point equations to validate the proof. This protocol, utilizing the Fiat-Shamir heuristic, ensures secure and private verification of equivalence on elliptic curves.

Algorithm 9: Non-Interactive Chaum-Pedersen Protocol for Elliptic Curves

Input: Elliptic Curve E over finite field $\mathbb{F}_q$, Base Point P, Prover's public key point B, secret scalar x, Transaction Identifier TxID, Timestamp TS

Output: True or False

Prover:

$c \xleftarrow{\$} \mathbb{Z}_q$

$U = cP,\ V = cB$

$e = H(E, \mathbb{F}_q, P, B, U, V, \text{TxID}, TS)$

$s = c + xe \mod q$

Send (U, V, e, s)

Verifier:

Receive (U, V, e, s)

Verify

$e \stackrel{?}{=} H(E, \mathbb{F}_q, P, B, U, V, \text{TxID}, TS)$

Check

$(sP \equiv U + eB) \wedge (sB \equiv V + exB)$

Return

$((sP \equiv U + eB) \wedge (sB \equiv V + exB))$

The EC Pedersen Commitment is a cryptographic protocol that allows one to commit to a chosen value while keeping it hidden, with the ability to reveal the committed value later. This algorithm takes as input a value v, which is the piece of information you want to commit to, and a random scalar r, which serves as a blinding factor to keep the commitment secret. It also requires two predefined base points G and H on an elliptic curve E, which are public parameters of the system [27].

The Elliptic Curve Pedersen Commitment, C, in algorithm 10, is calculated by performing elliptic curve scalar multiplication of the value v with the base point G, and the random scalar r with the base point H, then adding the results together on the elliptic curve. The resulting point C is the commitment. It effectively 'locks in' the value v in such a way that it cannot be changed without altering the commitment, but also does not reveal v or r. The commitment can later be opened by revealing v and r, allowing others to verify that C was constructed correctly. This process ensures both the hiding and binding properties required for secure commitments.

Algorithm 10: Elliptic Curve Pedersen Commitment

Input: Value v, Random scalar r, Base Points G, H on Elliptic Curve E

Output: Commitment C

$C \leftarrow vG + rH$

return C

V. Proposed solution

The system architecture of our proposed solution, as explained in section III-A, includes three main entities: the Trusted Authority, the Smart Meter, and the Control Center, and this section will explain how these interact to construct a distributed protocol for privacy-preserving billing in smart meters. This interaction is visualized in the sequence diagram presented in figure 2.

On the Trusted Authority: The process begins with the trusted authority distributing the cryptographic keys, having verified the identities of the entities.

On the Smart Meter:

1) *Measurement*: The smart meter starts by making a plaintext measurement of energy consumption.

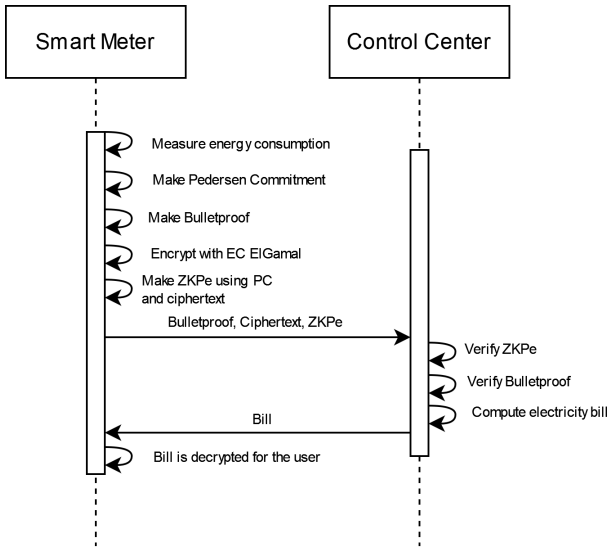

Fig. 2: Sequence diagram of the proposed distributed protocol.

2) *EC Pedersen Commitment*: It then creates an EC Pedersen Commitment based on this plaintext energy consumption measurement.
3) *Bulletproof Creation*: Following this, the smart meter generates a Bulletproof based on the plaintext measurement and the Pedersen Commitment, using a range approximating their consumption. These ranges can be constructed by knowing national average consumption data and normalizing the upper boundary of the range and the measured value, to fit the pre-defined Bulletproof range, specifically $[0; 2^n - 1]$, as defined in section IV-B1.
4) *EC ElGamal Encryption*: The plaintext measurement is then encrypted using EC ElGamal encryption, converting it into a ciphertext.
5) *Zero-Knowledge Proof of Equivalence (ZKPe)*: A ZKPe is constructed using both the EC Pedersen Commitment and the EC ElGamal ciphertext.
6) *Transmission to Control Center*: The Smart Meter sends the Bulletproof, the encrypted ciphertext, and the ZKPe to the control center for further processing.

On the Control Center:

1) *Verification of ZKPe*: Upon receiving the data, the control center first verifies the ZKPe to confirm that the committed and encrypted measurements correspond to the same value.
2) *Bulletproof Verification*: It then verifies the Bulletproof to ensure that the measurement lies in an expected range for that consumer segment. If it does not, the measurement is flagged for manual inspection on-premise at the client's.
3) *Bill Computation*: Once these verifications are complete, the Control Center computes the energy consumption bill based on the verified data using the homomorphic additive property of the EC ElGamal scheme. It does so by calculating the multiplicand, M, as the current price of 1 kwH by 100, thus allowing for 2 decimal precision, and then iteratively add the ciphertext to itself M-number of times.
4) *Bill Transmission*: This computed bill is then transmitted back to the smart meter.

Final Step on the Smart Meter:

1) *Bill Display*: The Smart Meter, upon receiving the bill, decrypts it, divides it by 100, and displays the bill to the user.

A. Proof-of-Concept

Throughout the creation of this proposed solution, we have been working on a PoC implementation. The PoC has been realized through the development of two key cryptographic modules: one for performing EC ElGamal key generation, encryption, and decryption, and one for generating and verifying Bulletproofs. These modules are implemented in C, leveraging the capabilities of the RELIC toolkit [29] for cryptographic primitives and the secp256k1 library [25], which has a PoC-implementation of Bulletproofs. In the PoC, we did not implement the abnormal measurement filtering, but the procedure is described in section V. About the ZKPe, a PoC is in the making, but not yet finished at the time of writing this report. To find and test the PoC, we refer to our data availability statement in section IX.

VI. Performance Evaluation

This section presents the performance evaluation of selected functionality in the PoC.

The testing was executed on an ASUS TUF GAMING FX504GM_FX80GM. This device has a Processor Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz, equipped with 4 Cores and 8 Logical Processors, and supported by 12 GB of RAM.

For the EC ElGamal-module, we measured the completion time of the following functions:

- The encrypt function that encrypts a plaintext message to a ciphertext.
- The decrypt function that decrypts a ciphertext to a plaintext message.
- The multiply function that multiplies a ciphertext.

In table 1 we visualize these results.

	Encrypt	**Multiply**	**Decrypt**
Int 1	0.000318	0.000004	0.000264
Int 10	0.000338	0.000005	0.000954
Int 100	0.000343	0.000008	0.008674
Int 1000	0.000338	0.000010	0.095879
Int 10000	0.000340	0.000014	1.059926
Int 100000	0.000346	0.000015	11.692359

TABLE I: Average time taken (in seconds) for the functions Encrypt, Multiply and Decrypt in the EC ElGamal implementation, with message sizes ranging from $1 - 100000$.

To measure the performance of the Bulletproof module, we also measured the completion time of the following functions:

- The Setup function, which initializes the environment variables needed to perform bulletproof range proofs.
- The Commit function, which creates the Pedersen Commitments for the associated Bulletproof.
- The Prove function that creates the Bulletproof for a set of values.
- The Verify function that checks the validity of the Bulletproof.

	Base
Setup	4.333424
Commit	0.000147
Prove	0.006413
Verify	0.001139

TABLE II: Average time taken (in seconds) for the functions Setup, Commit, Prove, Verify and Teardown. The tests involved 1 commitment and proof at a range of 2^{16}.

VII. Conclusive remarks

To conclude the study, it is useful to review each of the requirements of the protocol in the proposed solution, as defined in section I-A, and make some remarks on each objective.

A. Elliptic Curve ElGamal scheme for privacy-preserving billing

In section IV-A, we defined a scheme for EC ElGamal, based on Washington's book [9], which, natively, supports additive operations made on ciphertexts. In section V we integrated the scheme into a protocol and implemented the billing procedure to calculate the total price of energy consumption. To demonstrate the feasibility of the proposed protocol, we made a PoC implementation, in section V-A, and then tested it in section VI. Our preliminary results showed an average encryption speed, irrespective of measured value, of approximately 0.0003 seconds, equating to 0.3 milliseconds. For decryption, we saw that a value between 1000 and 10000, which is the most common household electricity consumption interval [30], took between 0.1 and 1.1 seconds. For smart meter computations, these execution speeds are assumed feasible. Finally, we observed a multiplication speed of 0.000004 and 0.000015 seconds, which is also assumed feasible, considering we do not have significant computation or memory constraints on the side of the control center.

In summary, we successfully proposed a distributed module for an EC ElGamal scheme that ensures secure data storage, transmission, processing, and privacy-preserving billing.

B. Bulletproof for abnormal filtering

In section IV-B1 we presented the underlying algorithms for making the Bulletproof, based on the original article named "Bulletproofs: Short Proofs for Confidential Transactions and More" [11]. In section V we integrated the scheme into a protocol, and, similarly to the ElGamal module, we made a PoC implementation, in section V-A. We then tested it in section VI, which showed that setting up the cryptographic environment was the most cumbersome function, taking 4.333424 seconds. However, as this process is only executed once, on the setup of the smart meter, it is assumed feasible. For the other functions, we measured similarly assumed feasible results, specifically 0.000147 and 0.006413 for making the EC Pedersen Commitment and generating the Bulletproof, respectively. Finally, on the side of the control center, we measured a verification time of 0.001139 seconds, which is also assumed feasible.

However, as declared in section V-A, we did not implement the abnormal measurement filtering in the PoC, therefore we cannot comment on this aspect.

In summary, we successfully proposed a distributed Bulletproof module, which allows for range-based filtering, thus allowing the utility provider to filter abnormal measurements.

C. Zero-Knowledge Proof of Equality for integrity

Similarly, in section IV-B we presented the protocol for generating the ZKPe, and we proposed a protocol integrating the scheme in section V. However, we did not make a PoC of it, but one is being made.

In summary, we successfully proposed a ZKPe module, which allows for verifying the integrity of the ciphertext.

VIII. Future work

Future development will go in three directions: finishing the PoC, realistic testing of the PoC and distributed key generation, each explained in dedicated subsections below.

A. Finishing the Proof-of-Concept

We plan to finish our PoC of the system, by implementing range-based filtering into the Bulletproof module. The difficulty in doing this is to calculate the range. However, this could be solved by knowing the average electricity consumption of an adult in a country over a specified period. Subsequently, one would need to identify the maximum expected consumption level, which acts as the upper limit in the normalization range. The next step involves scaling the measured consumption values to fit within the predefined range of $[0, 2^n - 1]$. This scaling could be executed by dividing the actual consumption values by the maximum consumption and then multiplying the result by $2^n - 1$. Once normalized, these values become suitable for constructing a Bulletproof range proof. We furthermore plan to finish the ZKPe module by using the EC Pedersen Commitment generated in the Bulletproof and the EC ElGamal ciphertext in the algorithm provided in section IV-B.

B. Smart meter testing

Once our last PoC module is finished, our efforts will be directed at validating our protocol on actual smart meters, moving beyond the preliminary testing conducted on a consumer-grade laptop. This shift is important, as laptops, with their significantly higher processing power and enhanced memory capabilities, do not accurately represent the constrained hardware environment of smart meters. Testing on laptops fails to capture the realities of limited computational resources, stringent energy constraints, and simpler operating systems that are characteristic of smart meter hardware. By conducting empirical tests on smart meters, we aim to gather data that reflects the protocol's performance in terms of computational load, energy efficiency, and operational reliability under these constrained conditions. Such targeted testing is essential for design space exploration and fine-tuning of our protocol.

C. Distributed key generation

Our last focus is on transitioning from a Trusted Authority-based key distribution model to a distributed key generation (DKG) mechanism [31]. Distributed key generation allows for the creation of cryptographic keys in a decentralized manner, eliminating single points of failure and enhancing the system's overall security and robustness.

D. Elliptic Curve Digital Signature Algorithm

As part of our future work, we also aim to integrate the Elliptic Curve Digital Signature Algorithm (ECDSA) into our protocol. The inclusion of ECDSA will provide robust mechanisms for ensuring non-repudiation and authenticity.

ECDSA, a widely-recognized standard for digital signatures based on ECC, offers enhanced security with smaller key sizes compared to traditional RSA-based systems [32]. By leveraging ECDSA, each smart meter can securely sign the data it transmits, ensuring non-repudiation and authenticity of the information. This feature is particularly important in scenarios where the source of data must be verifiable and indisputable, such as in legal disputes or billing inquiries [33].

IX. Data Availability Statement

To find our PoC and test it, please navigate to the dedicated GitHub-repository found here [34].

References

[1] Hassan Farhangi. "The path of the smart grid". In: *IEEE Power and Energy Magazine* 8.1 (2010), pp. 18–28.

[2] Soma Shekara Sreenadh Reddy Depuru, Lingfeng Wang, and Vijay Devabhaktuni. "Smart meters for power grid: Challenges, issues, advantages and status". In: *Renewable and Sustainable Energy Reviews* 15.6 (2011), pp. 2736–2742.

[3] Clark W Gellings. "The smart grid: enabling energy efficiency and demand response". In: (2009).

[4] Alireza Ipakchi and Farrokh Albuyeh. "Grid of the future". In: *IEEE Power and Energy Magazine* 7.2 (2009), pp. 52–62.

[5] Samet Tonyali et al. "Privacy Implications of Smart Meter Data: In-Depth Analysis". In: *Journal of Network and Computer Applications* 97 (2017), pp. 123–135.

[6] Eoghan McKenna, Ian Richardson, and Murray Thomson. "Smart meter data: Balancing consumer privacy concerns with legitimate applications". In: *Energy Policy* 41 (2012). Modeling Transport (Energy) Demand and Policies, pp. 807–814. ISSN: 0301-4215. DOI: https://doi.org/10.1016/j.enpol.2011.11.049. URL: https://www.sciencedirect.com/science/article/pii/S0301421511009438.

[7] Ann Cavoukian, Jules Polonetsky, and Christopher Wolf. "SmartPrivacy for the Smart Grid: embedding privacy into the design of electricity conservation". In: *Identity in the Information Society* 3 (Aug. 2010), pp. 275–294. DOI: 10.1007/s12394-010-0046-y.

[8] Taher El Gamal. "A public key cryptosystem and a signature scheme based on discrete logarithms". In: *Proceedings of CRYPTO 84 on Advances in cryptology*. Springer-Verlag New York, Inc., 1985, pp. 10–18.

[9] Lawrence C. Washington. *Elliptic Curves, Number Theory, and Cryptography, Second Edition*. College Park, Maryland, U.S.A.: Taylor & Francis Group, 2008.

[10] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The Knowledge Complexity of Interactive Proof Systems". In: *SIAM Journal on Computing* 18.1 (1989), pp. 186–208. URL: http://people.csail.mit.edu/silvio/Selected%20Scientific%20Papers/Proof%20Systems/The_Knowledge_Complexity_Of_Interactive_Proof_Systems.pdf.

[11] Benedikt Bünz et al. *Bulletproofs: Short Proofs for Confidential Transactions and More*. Cryptology ePrint Archive, Paper 2017/1066. https://eprint.iacr.org/2017/1066. 2017. URL: https://eprint.iacr.org/2017/1066.

[12] Z Erkin and G Tsudik. "Private computation of spatial and temporal power consumption with smart meters". In: *Proceedings of the 10th International Conference on Applied Cryptography and Network Security*. Springer. 2012, pp. 561–577.

[13] F Li, B Luo, and P Liu. "Secure Information Aggregation for Smart Grids Using Homomorphic Encryption". In: *Proceedings of First IEEE International Conference on Smart Grid Communications*. IEEE. 2010, pp. 327–332.

[14] Xiaolei Dong et al. "An ElGamal-based efficient and privacy-preserving data aggregation scheme for smart grid". In: *2014 IEEE Global Communications Conference*. 2014, pp. 4720–4725. DOI: 10.1109/GLOCOM.2014.7037553.

[15] Erfaneh Vahedi et al. "A secure ECC-based privacy preserving data aggregation scheme for smart grids". In: *Computer Networks* 129 (2017), pp. 28–36. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2017.08.025. URL: https://www.sciencedirect.com/science/article/pii/S1389128617303353.

[16] Jianbing Ni et al. "Differentially Private Smart Metering With Fault Tolerance and Range-Based Filtering". In: *IEEE Transactions on Smart Grid* 8.5 (2017), pp. 2483–2493. DOI: 10.1109/TSG.2017.2673843.

[17] Elaine Barker et al. "Recommendation for Key Management". In: *NIST Special Publication* 800.57 (2007).

[18] Darrel Hankerson, Alfred Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography*. Springer, 2004.

[19] Daniel J Bernstein. *Curve25519: new Diffie-Hellman speed records*.

URL: https://cr.yp.to/ecdh/curve25519-20060209.pdf. 2006.

[20] Steven D Galbraith. "Mathematics of Public Key Cryptography". In: (2012).

[21] Joseph H Silverman. "The Arithmetic of Elliptic Curves". In: *Graduate Texts in Mathematics*. Vol. 106. Springer. 2006.

[22] Vitali Bestolkau and Ignas Apšega. *Zero knowledge proof - how it works and the Alibaba Cave Experiment*. Sept. 2022. URL: https://www.byont.io/blog/zero-knowledge-proof-how-it-works-and-the-alibaba-cave-experiment.

[23] Oded Goldreich, Silvio Micali, and Avi Wigderson. "Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems". In: *Journal of the ACM*. Vol. 38. 3. ACM. 1991, pp. 690–728. URL: https://dl.acm.org/doi/10.1145/116825.116852.

[24] Manuel Blum, Paul Feldman, and Silvio Micali. "Non-interactive zero-knowledge and its applications". In: *Proceedings of the twentieth annual ACM symposium on Theory of computing* (1988). URL: https://dl.acm.org/doi/10.1145/62212.62222.

[25] Andrew Poelstra. *secp256k1-zkp with Bulletproofs*. GitHub repository. Complement to the paper "Bulletproofs: Short Proofs for Confidential Transactions and More". 2017. URL: https://github.com/apoelstra/secp256k1-zkp/tree/bulletproofs.

[26] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.

[27] Torben Pryds Pedersen. "Non-interactive and information-theoretic secure verifiable secret sharing". In: *Proceedings of CRYPTO 91 on Advances in Cryptology*. Springer. 1991, pp. 129–140.

[28] Benedikt Bünz et al. "Bulletproofs: Short proofs for confidential transactions and more". In: *Proceedings of the IEEE Symposium on Security and Privacy* (2018).

[29] D. F. Aranha et al. *RELIC is an Efficient LIbrary for Cryptography*. https://github.com/relic-toolkit/relic.

[30] Statistics Denmark. *Energy Prices*. https://www.dst.dk/en/Statistik/emner/miljoe-og-energi/energiforbrug-og-energipriser/energipriser. Accessed: 2023-12-21. 2023.

[31] Torben Pryds Pedersen. "A Threshold Cryptosystem without a Trusted Party". In: *Advances in Cryptology — EUROCRYPT '91*. Ed. by Donald W. Davies. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 522–526. ISBN: 978-3-540-46416-7.

[32] Don Johnson, Alfred Menezes, and Scott Vanstone. "The Elliptic Curve Digital Signature Algorithm (ECDSA)". In: *International Journal of Information Security* 1.1 (2001), pp. 36–63.

[33] American National Standards Institute. "Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)". In: *ANSI X9.62* (2005).

[34] Alexander Stæhr Johansen and Henrik Tambo Buhl. *Smart Meter Data Security: Integrating Zero-Knowledge Proofs and ElGamal for Privacy-Preserving Billing*. https://github.com/Alexan123321/R-D_privacy_preserving_smart_meter_billing. GitHub repository. 2023.