

# Comparison of the Paillier and ElGamal Cryptosystems for Smart Grid Aggregation Protocols

Fabian Knirsch<sup>ORCID</sup><sup>a</sup>, Andreas Unterweger, Maximilian Unterrainer and Dominik Engel

Center for Secure Energy Informatics, Salzburg University of Applied Sciences, Urstein Süd 1, 5412 Puch/Hallein, Austria

Keywords: Secure Aggregation, Paillier Cryptosystem, ElGamal Cryptosystem, Privacy.

**Abstract:** Many smart grid applications require the collection of fine-grained load data from customers. In order to protect customer privacy, secure aggregation protocols have been proposed that aggregate data spatially without allowing the aggregator to learn individual load data. Many of these protocols build on the Paillier cryptosystem and its additively homomorphic property. Existing works provide little or no justification for the choice of this cryptosystem and there is no direct performance comparison to other schemes that allow for an additively homomorphic property. In this paper, we compare the ElGamal cryptosystem with the established Paillier cryptosystem, both, conceptually and in terms of runtime, specifically for the use in privacy-preserving aggregation protocols. We find that, in the ElGamal cryptosystem, when made additively homomorphic, the runtime for encryption and decryption is distributed more asymmetrically between the smart meter and the aggregator than it is in the Paillier cryptosystem. This better reflects the setup typically found in smart grid environments, where encryption is performed on low-powered smart meters and decryption is usually performed on powerful machines. Thus, the ElGamal cryptosystem is a better, albeit overlooked, choice for secure aggregation protocols.

## 1 INTRODUCTION

Collecting fine-grained load data from smart meters installed in the customer premises has shown to pose severe privacy risks (Wicker and Thomas, 2011; McKenna et al., 2012; Burkhart et al., 2018). To mitigate them, secure aggregation protocols have been proposed by many authors for privacy-preserving data aggregation in the smart grid, e.g., (Li et al., 2010; Erkin and Tsudik, 2012; Knirsch et al., 2017). These protocols protect customer privacy by only providing the sum of load data from a number of households at one point in time (Buescher et al., 2017).

One approach for secure aggregation protocols is to employ an additively homomorphic cryptosystem and an entity that acts as a (semi-trusted) aggregator. Each smart meter encrypts its individual measurement and sends the encrypted value to the aggregator. The aggregator uses the additively homomorphic property of the underlying cryptosystem to calculate the *encrypted* sum and forwards this sum to the energy provider, who decrypts it. This way, the aggregator does not learn individual meter readings, but

needs to be trusted for performing the correct aggregation (Unterweger et al., 2019). Figure 1 shows the principal setup and actors of such a smart grid aggregation protocol.

Existing works, e.g., (Li et al., 2010; Erkin and Tsudik, 2012; Erkin, 2015; Rane et al., 2015) provide little or no justification for the choice of the cryptosystem and most commonly employ the Paillier scheme (Paillier, 1999). Only very few publications make use of the ElGamal cryptosystem in the context of smart grids, e.g., (Busom et al., 2016). To the best of our knowledge, there exists no performance comparison between the Paillier cryptosystem and the ElGamal cryptosystem (ElGamal, 1985), which are the two most commonly used homomorphic cryptosystems (Armknecht et al., 2013) and also those with the highest security guarantees (Fontaine and Galand, 2007).

While ElGamal is more lightweight in terms of encryption complexity, it is multiplicative homomorphic, but can be made additively homomorphic (Cramer et al., 1997). This can be advantageous for low-powered devices such as smart meters. For this reason, the ElGamal cryptosystem is already widely employed in e-voting applications, as presented in

<sup>a</sup>  <https://orcid.org/0000-0002-6346-5759>

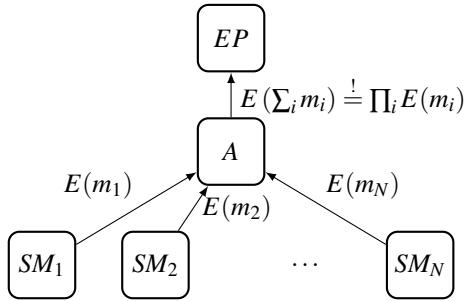


Figure 1: Aggregation protocol with homomorphic encryption: Each smart meter ( $SM_i$ ) sends its encrypted value  $E(m_i)$  to the aggregator ( $A$ ). The aggregator calculates the sum of the values in the ciphertext domain using the additive homomorphic property of the cryptosystem. The result is sent to the energy provider ( $EP$ ) which decrypts it to obtain the plaintext sum  $\sum_i m_i$  of the readings.

(Adida, 2008; Chaum et al., 2008; Culnane et al., 2015).

## 1.1 Related Work

The properties of the Paillier cryptosystem have been investigated in detail by, e.g., (Catalano et al., 2001; Damgård and Jurik, 2001; Damgård et al., 2010). The same is true for a number of variations of the system, e.g., (Fouque et al., 2001; Galbraith, 2002; Hazay et al., 2012). Similarly, investigations for the ElGamal cryptosystem and its variations exist, e.g., (Cramer and Shoup, 2003; Kumar and Madrai, 2012; Armknecht et al., 2013).

A general runtime comparison between the ElGamal and the multiplicative homomorphic RSA cryptosystem has been conducted by (Maqsood et al., 2017). However, the Paillier cryptosystem has not been considered in their analysis, as opposed to our work.

A non-peer-reviewed publication titled “An experimental study on Performance Evaluation of Asymmetric Encryption Algorithms” by Farah et al. exists which describes runtime results for both, the Paillier and ElGamal cryptosystems. However, their reported results do not increase with increasing plaintext size and even drop to zero for some plaintexts, casting doubts on their numbers and thus their conclusions.

To the best of our knowledge there is currently no comparison of the Paillier and ElGamal cryptosystem in the context of smart grid aggregation protocols and no analysis of the suitability of the latter in practical setups exists. Thus, this aspect is investigated in this paper.

## 1.2 Contribution

This paper briefly presents both, the Paillier and the ElGamal cryptosystems with extensions from the literature to make them comparable for additive homomorphic operations. The main contribution of this paper is the runtime analysis and comparison of the two cryptosystems for aggregated smart meter data. The detailed analysis of encryption and decryption times as well as of each relevant algorithmic step allows for conclusions on the suitability and practicability of both cryptosystems for secure aggregation. It also allows for recommendations on which system to prefer for this smart grid aggregation use case. In this paper, we find that the ElGamal cryptosystem is overlooked for many proposed protocols. The encryption is more lightweight compared to Paillier cryptosystem and the additional overhead at decryption can be mitigated by powerful devices in the EP’s premises or is negligible for practical applications.

## 1.3 Structure

The paper is structured as follows: Section 2 describes the Paillier and ElGamal cryptosystems, as well as additional algorithms to make them comparable. Section 3 compares both cryptosystems and their application for privacy-preserving aggregation. Section 4 summarizes the findings and gives an outlook to future work.

## 2 BACKGROUND

In this section, the Paillier and ElGamal cryptosystems, which are compared in this paper, are explained briefly, together with their relevant properties. In addition, the Cramer transformation is described which allows using the ElGamal cryptosystem in such a way that it becomes comparable to the Paillier cryptosystem. Furthermore, multiple algorithms for calculating the aggregate after ElGamal decryption are described.

### 2.1 Paillier Cryptosystem

The Paillier cryptosystem is an additively homomorphic, semantically secure public-private key cryptosystem (Paillier, 1999; Catalano et al., 2001). An additively homomorphic cryptosystem fulfills the equation

$$D(E(m_1) \cdot E(m_2)) = m_1 + m_2$$

for two plaintexts  $m_1$  and  $m_2$ , where  $E$  and  $D$  denote the encryption and decryption functions, respectively.

Given two large prime number  $p$  and  $q$  of the same length, the public key  $(n, g)$  is calculated by

$$n = pq, g = n + 1$$

and the private key  $\lambda$  is calculated by

$$\lambda = \varphi(n), \mu = \varphi(n)^{-1} \pmod{n},$$

where  $\varphi(n) = (p - 1)(q - 1)$ .

Encryption of a plaintext  $m \in \{0, 1, \dots, n - 1\}$  to a ciphertext  $c$  is performed by

$$c = g^m r^n \pmod{n^2},$$

with a random number  $r \in \{1, 2, \dots, n - 1\}$ .

Given two ciphertexts  $c_1$  and  $c_2$ , the additive homomorphic property can be shown by

$$c_1 \cdot c_2 = g^{m_1+m_2} (r_1 \cdot r_2)^n \pmod{n^2}.$$

Decryption of the above expression will result in the sum of  $m_1$  and  $m_2$ .

## 2.2 ElGamal Cryptosystem

The ElGamal cryptosystem is a public-private key cryptosystem with a multiplicatively homomorphic property (ElGamal, 1985). A multiplicatively homomorphic cryptosystem fulfills the equation

$$D(E(m_1) \cdot E(m_2)) = m_1 \cdot m_2.$$

Given a publicly known cyclic group  $G$  of order  $q$  with a publicly known generator  $g$ , the public key is

$$h = g^r,$$

with  $r \in \{1, 2, \dots, q - 1\}$  being randomly chosen.  $r$  is the private key.

Encryption of a plaintext  $m$  to a ciphertext  $c$  is performed by

$$c_1 = g^s, c_2 = m \cdot h^s,$$

with a randomly chosen  $s \in \{1, 2, \dots, q - 1\}$ .

Calculating the product of two ciphertexts yields the product of the corresponding plaintexts after decryption.

## 2.3 Cramer Transformation

The ElGamal cryptosystem with its multiplicatively homomorphic property can be transformed into an additively homomorphic cryptosystem (like the Pailier) scheme with the Cramer transformation (Cramer et al., 1997).

The plaintext value to be encrypted must be in the exponent. In practice,  $m$  can be transformed to  $m'$  by

$$m' = g^m \pmod{q}.$$

The encryption is then applied to  $m'$ . Note, however, that, after decryption, this yields  $g^{\sum_i m_i}$  as a plaintext. To solve the discrete logarithm and to retrieve the actual value  $\sum_i m_i$ , one of three recovery algorithms can be applied: (i) Brute Force; (ii) Pollard's Lambda; or (iii) Baby Step Giant Step.

## 2.4 Brute Force Algorithm

To recover  $m$  from  $m' = g^m \pmod{q}$  with known  $g$  and  $q$ , the discrete logarithm  $\log_g(m') = m \pmod{q}$  can be solved by brute forcing all possible values of  $m$ , until a solution is found. The complexity of this algorithm is  $O(q)$ . If it is known that  $m$  is within a given interval  $[0, b]$ , the complexity is reduced to  $O(b)$ .

Linear speedup can be achieved by parallelizing the brute-force search. While not reducing the complexity itself, the constant factor of the runtime is reduced proportional to the number of parallel searches.

## 2.5 Pollard's Lambda Algorithm

Pollard's Lambda algorithm is designed to solve the discrete logarithm and achieves a runtime complexity of  $O(\sqrt{b} - a)$  for a plaintext in the interval  $m \in [a, b]$  (Pollard, 1978). This assumes that  $m$  is with certainty within the defined interval.

Pollard's Lambda algorithm rewrites  $m = b + d - d_k \pmod{q}$  for a  $k \in [0, I - 1]$ .  $I$  can be chosen according to (Pollard, 1978). First, a sequence  $x_{0..I}$  is computed where  $x_0 = g^b$  and

$$x_{i+1} = x_i g^{f(x_i)},$$

with  $f$  being a parameterizable pseudorandom function.  $d$  is calculated as

$$d = \sum_{i=0}^{I-1} f(x_i).$$

Then, a sequence  $y_{0..k}$  is computed where  $y_0 = m'$ ,

$$y_{i+1} = y_i g^{f(y_i)},$$

and additionally, a sequence  $d_{0..k}$  is computed by

$$d_j = \sum_{i=0}^{j-1} f(y_i).$$

If  $y_k = x_I \pmod{q}$ , then  $m' g^{d_k} = g^{b+d} \pmod{q}$ , which equals  $g^{m+d_k} = g^{b+d} \pmod{q}$  and thus  $m = b + d - d_k \pmod{q}$ . If  $k > I$  or  $d_i > b - a + d \pmod{q}$ , the algorithm fails and the choice of  $f$  needs to be adapted.

## 2.6 Baby Step Giant Step Algorithm

The Baby Step Giant Step algorithm is designed to solve the discrete logarithm and achieves a runtime complexity of  $O(\sqrt{q})$ , where  $q$  is the order of the group (Shoup, 1997; Schnorr and Jakobsson, 2000).

$m' = g^m \pmod{q}$  (see previous section) can be rewritten as  $m' = g^{kx+l} \pmod{q}$  where  $x = \lceil \sqrt{q} \rceil$ .

The algorithm first computes  $g^l \pmod{q}$  for  $0 \leq l < x$  and then tries the values  $0 \leq k < x$  until

$$g^l = m'(g^{-x})^k \pmod{q}$$

holds. This comparison is realized efficiently by having a look-up table with the precomputed values of  $g^l$ .

### 3 COMPARISON

In this section, we compare the complexity and runtime properties of the Paillier and ElGamal cryptosystems in a setup typical for smart grid secure aggregation use cases.

For the comparison, we use a setup that reflects typical capabilities that can be found in the field. Smart meters are lightweight devices with limited computational resources, whereas the EP can have more powerful devices to decrypt and calculate the aggregate, respectively. The setup for the comparison is based on the aggregation protocol depicted in Figure 1, which reflects a setup commonly found in literature, see, e.g., (Li et al., 2010; Erkin, 2015; Rane et al., 2015). Each smart meter encrypts its measurement value using either the Paillier cryptosystem or the ElGamal cryptosystem and submits the encrypted values to an aggregator. The aggregator calculates the product of these values and forwards the result to the energy provider. The EP then decrypts the aggregate.

#### 3.1 Evaluation Setup

Both cryptosystems, the Cramer transformation and the recovery algorithm, are implemented in Java 8. Measurements are performed on Windows 10 using an Intel Core i7-6500U Processor<sup>1</sup> capable of running 4 threads. The following implementations are used

- *Paillier Encryption Threshold Toolbox*<sup>2</sup>. This toolbox implements the Paillier cryptosystem and also provides support for the thresholded variant.
- The ElGamal cryptosystem and the Cramer transformation are implemented according to (Cramer et al., 1997) and (ElGamal, 1985).
- Pollard’s Lambda algorithm is implemented according to (Pollard, 1978).

<sup>1</sup><https://ark.intel.com/products/88194/>  
Intel-Core-i7-6500U-Processor-4MCache-up-to-3\_10-GHz

<sup>2</sup><http://www.cs.utdallas.edu/dspl/cgi-bin/pailliertoolbox/index.php>

- The Baby Step Giant Step algorithm is a modified version of an implementation from the University of Maryland<sup>3</sup>, which can be extended in order to support multiple threads.

The implementations for Paillier, ElGamal, Cramer and the recovery algorithm only use the formulas as in Section 2 without additional overhead. All of them rely on the Java BigInteger class<sup>4</sup> and its performance, so that the time required for the calculations is comparable.

The current recommended key length for cryptosystems based on the discrete logarithm problem to be secure for the year 2030 and beyond is 3072 bits according to (Barker, 2016). As this applies to both, the Paillier cryptosystem (Paillier, 1999) and the ElGamal cryptosystem (ElGamal, 1985), a key length of 3072 bits is used for the evaluation.

All measurements are repeated 110 times in order to minimize external factors on the mean value. The first ten measurement values are discarded to mitigate cache-warming effects of the Java Virtual Machine.

#### 3.2 Paillier and ElGamal Runtime

Figure 2 depicts the runtimes for encryption and decryption of the Paillier and ElGamal cryptosystems. It is clear that the encryption time for the ElGamal cryptosystem is more than one order of magnitude smaller (approximately one 15<sup>th</sup>) than that of the Paillier cryptosystem. The reason for this discrepancy is the much smaller range in the size of the exponent for which the calculations for the ElGamal encryption are performed (see Section 2).

However, in order to allow for an additively homomorphic property for the ElGamal cryptosystem, the Cramer transformation as described in Section 2 needs to be applied to the plaintext value. This is already included in the values depicted in Figure 2. Figure 3 shows how large the runtime portion for this transformation (0.042 ms) is compared to the total runtime (2.709 ms). This portion is negligible in practice.

Similar to the encryption runtime, the decryption time for the ElGamal cryptosystem is lower than that of the Paillier cryptosystem. The runtime is about one fourth. Note that the result of the decryption is, however, not the sum of the plaintext values  $\sum_i m_i$ , but rather  $g^{\sum_i m_i}$ . Thus, one of the recovery algorithms mentioned in Section 2 (e.g., Brute Force algorithm) needs to be applied, which incurs additional runtime.

<sup>3</sup><https://www.csee.umbc.edu/~stephens/crypto/CIPHERS/BigIntegerMath.java>

<sup>4</sup><https://docs.oracle.com/javase/8/docs/api/java/math/BigInteger.html>

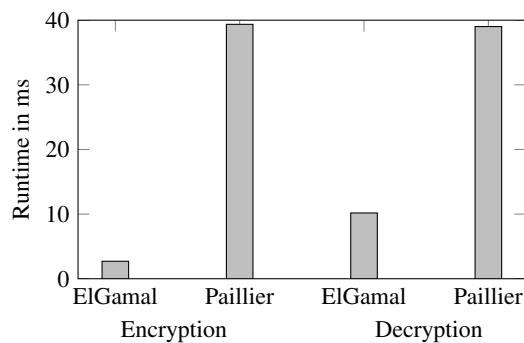


Figure 2: Comparison of encryption and decryption runtimes for the ElGamal cryptosystem and the Paillier cryptosystem.

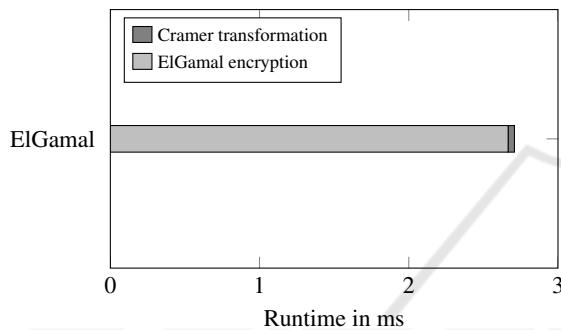


Figure 3: Breakdown of runtime for the steps of ElGamal encryption.

### 3.3 Recovery Runtime

After decryption, the EP has to apply a recovery algorithm for retrieving the actual sum of the values. In order to assess the required runtime for this step, an aggregation group size (number of smart meters  $N$ ) and a value range (size of  $m_i$ ) that is representative for real-world applications (Erkin, 2015) (and is commonly used in related work (Knirsch et al., 2018; Buescher et al., 2017)) are used: Each measurement  $m_i$  is 16 bits in size and the aggregation group size  $N$  (number of smart meters) is varied between one and  $2^7 = 128$ . Aggregating 16-bit values with these group sizes results in measurement sums  $\sum_i m_i$  between  $2^{16}$  and  $2^{23}$ .

Note that the EP can have powerful devices to perform the calculations and thus any recovery algorithm can be implemented efficiently using multiple threads. For our analysis, the value range is split into  $T \in \{1, 2, 4\}$  intervals of equal length and one thread is used per interval. The number of threads is limited by the properties of the processor.

Figure 4 shows runtimes for the Brute Force algorithm with a varying number of threads. The vertical bars denote the standard deviation for each point.

Each thread starts at a defined value, e.g.,  $m = 2^{16}$  for the first thread, calculates the discrete logarithm and then increases this value by one in each step until  $m'$  is found. For  $T = 1$ , the entire space of the measurement sum is thus searched linearly. The runtime increases nearly linearly with the size of  $\sum_i m_i$  (note that both axes are logarithmic).

Increasing the number of threads incurs a near constant overhead for starting the additional threads. This leads to higher runtimes for small values of  $\sum_i m_i$ . However, this allows partitioning the search space and thus running the recovery algorithm in parallel.

Running the algorithm in parallel leads to earlier recoveries. Once  $\sum_i m_i$  has been successfully recovered, all threads can be stopped. It can be seen that a higher number of threads in general improves runtime for larger values of  $\sum_i m_i$ . For example, the worst case runtime for one thread is 104.5 s, while it is 49.9 s for four threads. It can also be seen from the results that the choice of the starting value for a thread significantly impacts the time needed for recovery. Since for practical applications an interval for  $\sum_i m_i$  can be guessed, this will improve overall performance.

While the Brute Force algorithm only provides a baseline which can be outperformed by more efficient algorithms, such as Baby Step Giant Step and Pollard's Lambda, it can be seen that the time needed for recovering the sum of the measurements is below 50 s even for the largest value of  $\sum_i m_i$  using four threads.

Other algorithms for recovery as described in Section 2 can be used. This will further reduce the complexity, but not necessarily the runtime for this step. For the Baby Step Giant Step algorithm and Pollard's Lambda algorithm in the practical evaluation we could not find suitable algorithm parameters for which any value bit size could be processed faster than Brute Force.

Alternatively, due to the comparably small range of  $\sum_i m_i$  (i.e., a small plaintext space), a lookup table can be constructed that stores all possible values for  $g^{\sum_i m_i}$  and the corresponding exponent. This allows trading additional memory consumption against access time (Croman et al., 2016). In particular, an access complexity of  $O(1)$  can be achieved with 3095 MiB of additional memory with  $2^{23}$  entries mapping one 3072 bit ciphertext to one 23 bit plaintext each.

However, as the results for the Brute Force algorithm are already practically feasible, the use of a lookup table is not necessary and the fine-tuning of the parameters of the Baby Step Giant Step algorithm and Pollard's Lambda algorithm remain future work.

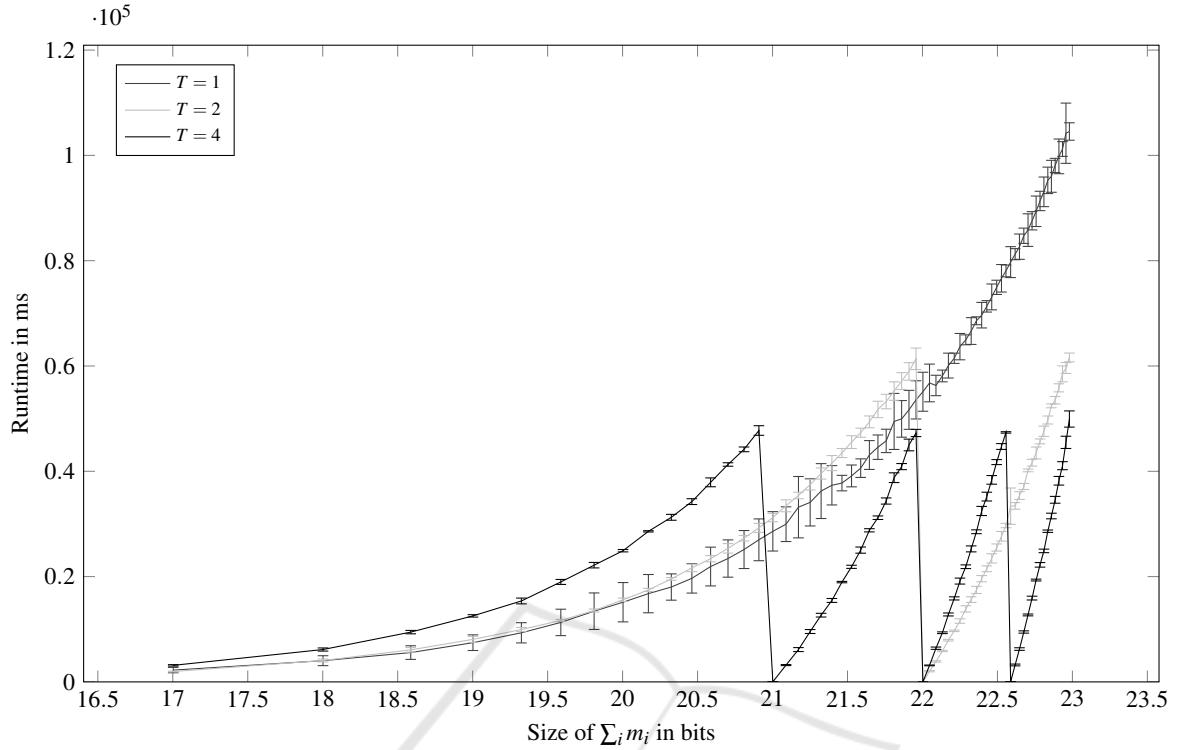


Figure 4: Runtime for the Brute Force algorithm after ElGamal decryption with a variable number of threads  $T$ . The vertical bars denote the standard deviation.

### 3.4 Combined Runtime

The combined runtime is the time needed for decryption in case of the Paillier cryptosystem and the ElGamal cryptosystem as well as the additional time needed for recovery when using the latter.

Table 1 summarizes the runtimes for SM and EP for both cryptosystems. It can be seen that the Paillier cryptosystem requires about 15 times the runtime for encryption compared to the ElGamal cryptosystem, but performs equally for encryption and decryption. On low-powered smart meters, this gives the ElGamal cryptosystem a significant advantage over the Paillier cryptosystem.

In Section 3.2 the ElGamal decryption runtime is found to be 10.179 ms. The time needed for recovery is at most 49930 ms (from Section 3.3). Thus, the combined runtime is always below 50 s. In most European countries, measurements of one day are submitted only once per day (see e.g., (Nationalrat, 2010; Bundestag, 2016)). Hence, the aggregator has a time frame of 24 hours to recover the sum before the values of the next day need to be processed. Even for real-time network stability monitoring with 15 minute intervals, the combined runtime is comparably small.

Note that the measurements above have been per-

formed on consumer-grade hardware and in a practical setup much more powerful hardware is used, further reducing the runtime.

## 4 CONCLUSION

Both, the Paillier and the ElGamal cryptosystem are suitable for secure aggregation protocols in the smart grid due to the ability to enable additively homomorphic operations. For encryption, the ElGamal cryptosystem is significantly (more than one order of magnitude) faster than the Paillier cryptosystem. This is highly beneficial for low-powered devices such as smart meters. For decryption, the Paillier cryptosystem is significantly faster overall. However, decryption is performed by energy providers, which typically host large-scale servers for this purpose where runtime is not as crucial as it is on smart meters. Even on significantly less powerful consumer-grade hardware, the additional runtime required for recovery when using the ElGamal cryptosystem is small enough in practical setups. In summary, the ElGamal cryptosystem is to be preferred for such smart grid use cases, despite the Paillier cryptosystem being commonly used.

Table 1: Worst-case runtimes in seconds (rounded) with four significant digits for all operations involving encryption and decryption for the ElGamal and Paillier cryptosystems.

	ElGamal		Paillier	
	SM runtime [s]	EP runtime [s]	SM runtime [s]	EP runtime [s]
Encryption	0.003	–	0.039	–
Decryption	–	0.010	–	0.039
Cramer transformation	0.000	–	–	–
Recovery algorithm	–	49.930	–	–
Total	0.003	49.940	0.039	0.039

## ACKNOWLEDGEMENTS

The financial support by the Federal State of Salzburg is gratefully acknowledged.

## REFERENCES

- Adida, B. (2008). Helios : Web-based Open-Audit Voting. In *USENIX Security Symposium*, volume 17, pages 335–348.
- Armknecht, F., Katzenbeisser, S., and Peter, A. (2013). Group homomorphic encryption: Characterizations, impossibility results, and applications. *Designs, Codes, and Cryptography*, 67:209–232.
- Barker, A. (2016). NIST Special Publication 800-57: Recommendation for Key Management - Part 1: General (Revised).
- Buescher, N., Boukoros, S., Bauregger, S., and Katzenbeisser, S. (2017). Two Is Not Enough: Privacy Assessment of Aggregation Schemes in Smart Metering. *Proceedings on Privacy Enhancing Technologies*, 2017(4):118–134.
- Bundestag (2016). Gesetz zur Digitalisierung der Energiewende. *Bundesgesetzblatt Teil I*, 2016(43):2034–2064.
- Burkhart, S., Unterweger, A., Eibl, G., and Engel, D. (2018). Detecting Swimming Pools in 15-Minute Load Data. In *17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications (TrustCom 2018)*, pages 1651–1655, New York, NY, USA. IEEE.
- Busom, N., Petrlic, R., Seb  , F., Sorge, C., and Valls, M. (2016). Efficient smart metering based on homomorphic encryption. *Computer Communications*, 82:95–101.
- Catalano, D., Howgrave-graham, R. G. N., and Nguyen, P. Q. (2001). Paillier’s Cryptosystem Revisited. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 206–214. ACM.
- Chaum, D., Carback, R., Clark, J., Essex, A., Popoveniuc, S., Rivest, R. L., Ryan, P. Y. A., Shen, E., and Sherman, A. T. (2008). Scantegrity II: End-to-End Verifiability for Optical Scan Election Systems using Invisible Ink Confirmation Codes. In *USENIX Security Symposium*, volume 8, pages 1–13.
- Cramer, R., Gennaro, R., and Schoenmakers, B. (1997). A Secure and Optimally Efficient Multi-Authority Election Scheme. In *Proceedings of the 16th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT’97, pages 103–118, Konstanz, Germany. Springer-Verlag.
- Cramer, R. and Shoup, V. (2003). Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. *SIAM Journal on Computing*, 33:167–226.
- Cromer, K., Decker, C., Eyal, I., Gencer, A. E., Juels, A., Kosba, A., Miller, A., Saxena, P., Shi, E., G  n Sirer, E., Song, D., and Wattenhofer, R. (2016). On Scaling Decentralized Blockchains. In *International Conference on Financial Cryptography and Data Security*, pages 106–125, Christ Church, Barbados. Springer.
- Culnane, C., Ryan, P. Y. A., Schneider, S., and Teague, V. (2015). vVote: a Verifiable Voting System. *ACM Transactions on Information and System Security (TISSEC)*, 18(1).
- Damg  rd, I., Jurik, M., and Buus, J. (2010). A generalization of Paillier’s public-key system with applications to electronic voting. *International Journal of Information Security*, pages 371–385.
- Damg  rd, I. and Jurik, M. J. (2001). A Generalisation, a Simplification and some Applications of Paillier’s Probabilistic Public-Key System. In *PKC 2001: Public Key Cryptography*, pages 119–136.
- ElGamal, T. (1985). A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472.
- Erkin, Z. (2015). Private Data Aggregation with Groups for Smart Grids in a Dynamic Setting using CRT. In *2015 IEEE International Workshop on Information Forensics and Security (WIFS)*, Rome, Italy. IEEE.
- Erkin, Z. and Tsudik, G. (2012). Private Computation of Spatial and Temporal Power Consumption with Smart Meters. In Bao, F., Samarati, P., and Zhou, J., editors, *Proceedings of the 10th international conference on Applied Cryptography and Network Security*, ACNS’12, pages 561–577. Springer, Berlin Heidelberg.
- Fontaine, C. and Galand, F. (2007). A Survey of Homomorphic Encryption for Nonspecialists. *EURASIP Journal on Information Security*.
- Fouque, P. a., Poupart, G., and Stern, J. (2001). Sharing de-

- ryption in the context of voting or lotteries. *Financial Cryptography*, pages 90–104.
- Galbraith, S. D. (2002). Elliptic curve Paillier schemes. *Journal of Cryptology*, pages 1–10.
- Hazay, C., Mikkelsen, G. L., Rabin, T., and Toft, T. (2012). Efficient RSA key generation and threshold Paillier in the two-party setting. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7178 LNCS:313–331.
- Knirsch, F., Eibl, G., and Engel, D. (2018). Error-resilient Masking Approaches for Privacy Preserving Data Aggregation. *IEEE Transactions on Smart Grid*, 9(4):3351–3361.
- Knirsch, F., Engel, D., and Erkin, Z. (2017). A Fault-tolerant and Efficient Scheme for Data Aggregation Over Groups in the Smart Grid. In *9th IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, Rennes, France. IEEE.
- Kumar, V. and Madrai, S. (2012). Secure Hierarchical Data Aggregation in Wireless Sensor Networks : Performance Evaluation and Analysis. *13th International Conference on Mobile Data Management*, pages 196–201.
- Li, F., Luo, B., and Liu, P. (2010). Secure Information Aggregation for Smart Grids Using Homomorphic Encryption. In *Proceedings of First IEEE International Conference on Smart Grid Communications*, pages 327–332, Gaithersburg, Maryland, USA. IEEE.
- Maqsood, F., Ahmed, M., Ali, M. M., and Shah, M. A. (2017). Cryptography: A Comparative Analysis for Modern Techniques. *International Journal of Advanced Computer Science and Applications*, 8(6):442–448.
- McKenna, E., Richardson, I., and Thomson, M. (2012). Smart meter data: Balancing consumer privacy concerns with legitimate applications. *Energy Policy*, 41:807–814.
- Nationalrat (2010). Bundesgesetz, mit dem die Organisation auf dem Gebiet der Elektrizitätswirtschaft neu geregelt wird (Elektrizitätswirtschafts- und -organisationsgesetz 2010 – EIWOG 2010). BGBI. I Nr. 110/2010 (NR: GP XXIV RV 994 AB 997 S. 86. BR: 8420 AB 8421 S. 791.).
- Paillier, P. (1999). Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Stern, J., editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, Berlin Heidelberg.
- Pollard, B. J. M. (1978). Monte Carlo Methods for Index Computation ( $\text{mod } p$ ). 32(143):918–924.
- Rane, S., Freudiger, J., Brito, A. E., and Uzun, E. (2015). Privacy , Efficiency & Fault Tolerance in Aggregate Computations on Massive Star Networks. In *7th IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, Rome, Italy. IEEE.
- Schnorr, C. P. and Jakobsson, M. (2000). Security of Signed ElGamal Encryption. In Okamoto, T., editor, *Advances in Cryptology - ASIACRYPT 2000*, pages 73–89, Berlin Heidelberg. Springer Berlin Heidelberg.
- Shoup, V. (1997). Lower Bounds for Discrete Logarithms and Related Problems. In Fumy, W., editor, *Advances in Cryptology - EUROCRYPT '97*, pages 256–266, Berlin Heidelberg. Springer Berlin Heidelberg.
- Unterweger, A., Taheri-Boshrooyeh, S., Eibl, G., Knirsch, F., Küpü, A., and Engel, D. (2019). Understanding Game-Based Privacy Proofs for Energy Consumption Aggregation Protocols. *IEEE Transactions on Smart Grid*, 10(5):5514–5523.
- Wicker, S. B. and Thomas, R. (2011). A privacy-aware architecture for demand response systems. In *Proceedings of the Annual Hawaii International Conference on System Sciences*, Koloa, HI, USA. IEEE.