A privacy-friendly aggregation algorithm for demand side management of residential loads

Emilio J. Palacios-Garcia*, Joppe W. Bos†, Xavier Carpent‡, and Geert Deconinck*
*Dept. of Electrical Engineering (ESAT-ELECTA), KU Leuven, Leuven, Belgium
†NXP Semiconductors, Leuven, Belgium
‡Dept. of Electrical Engineering (ESAT-COSIC), KU Leuven, Leuven, Belgium
Emails: emiliojose.palaciosgarcia@kuleuven.be, joppe.bos@nxp.com, xavier.carpent@esat.kuleuven.be, geert.deconinck@kuleuven.be

***Abstract*—One of the main challenges of residential demand-side management (DSM) is achieving enough aggregated flexibility to participate in the different energy markets. Aggregators enable such operation by managing a large portfolio of individual customers. However, this also involves a continuous flow of information from the individual users to the Aggregator. Due to the sensitive nature of energy data, from which behaviour and usage patterns can be inferred easily, not every customer is willing to share this information. What is more, considering the current efforts to protect personal data, the limitation of data collection and exposure should be a priority of any modern system. To achieve such functionality, this work presents a novel privacy-friendly aggregation algorithm that combines market-based control (MBC), public-key cryptography and secret sharing, with the goal to provide the same energy aggregation services but without revealing the individual user profiles.**

***Index Terms*—data privacy, demand-side management, energy flexibility, public-key cryptography**

I. Introduction

With increasing production volumes coming from renewable sources, and bidirectional power flows in the network, demand side management (DSM) is an effective tool that transmission/distribution system operators (TSO/DSO) can employ to balance the network. Operators interact with end consumers, so they increase, decrease, or reschedule certain loads [1].

Since the energy volumes that residential consumers can provide are relatively low, users must rely on an intermediary party, generally known as the Aggregator. Its task is to manage the aggregated flexibility of a group of households [2].

The problem lies in the amount of sensitive information contained in residential energy data. As non-intrusive load monitoring (NILM) techniques have demonstrated, usage patterns can be inferred from household measurements [3], [4]. Hence, not every costumer is willing to share their energy data, which results in a low acceptance of DSM strategies in the residential sector [5]. Besides, there are also international regulations, such as the GDPR in Europe [6], concerning the amount of data that can be collected from individual users.

This work was supported by Flanders Innovation and Entrepreneurship (VLAIO) within the ICON project, "PrivateFlex: Privacy-friendly Power Flexibility Trading", grant number HBC.2019.0073.

Privacy concerns can be mitigated by performing meaningful computations on *encrypted data*. The concept was introduced in the 1970s [7] but it was not instantiated concretely until 2009 in the form of fully homomorphic encryption (FHE) [8]. This technique allows an untrusted party to carry out any arbitrary computation on *encrypted data* without learning anything. However, for the application considered in this work, where only aggregation is needed, privacy-concerns can be tackled much more efficiently by simply making use of standardised public-key cryptography schemes (such as RSA [7] or elliptic curve cryptography [9], [10]), and computation on shares. This last technique relies on the idea of splitting up the data so, when every share is combined again, the original information is revealed [11]. This work combines such cryptographic schemes with market-based control (MBC) algorithms to unlock a privacy-friendly energy trading.

II. Background:Three-step approach

The privacy-friendly aggregation is implemented as an extension of [12], which proposed a MBC three-step approach (TSA) for controlling a fleet of electrical vehicles (EVs).

A. Step 1: Constraints aggregation

The first step is aggregating the local constraints of each home energy management system (HEMS) i in the set of HEMSs, $\mathcal{H} = \{1, \dots, H\}$. This step represents the core issue, as we want to prevent the Aggregator from learning the individual flexibility representations.

In the most generic case, each HEMS i provides the Aggregator with a set of vectors representing the maximum $\mathbf{E}^i_{\text{max}}$ and minimum $\mathbf{E}^i_{\text{min}}$ energy boundaries, as well as, optionally, the maximum $\mathbf{P}^i_{\text{max}}$ and minimum $\mathbf{P}^i_{\text{min}}$ power constraints. The length of these vectors will depend on the step interval Δt over the samples in the time horizon $t \in \mathcal{T} = \{1, \dots, T\}$.

In addition, a power demand constraints vector $\mathbf{P}^i_{\text{dem}}$ is used to represent the control range over a set of priority values. In previous studies [12], this priority was expressed as the piece-wise linear bidding function $f^i(p, p^i_*)$ in (1), which depends on the priority range $p \in [0, 1]$ and the so-called corner priority of the load p^i_*. The number of samples in this vector is conditioned by the selected priority resolution Δp.

$$f^i(p, p_*^i) = \begin{cases} P_{\max}^i - p\left(\frac{P_{\max}^i - P_{\min}^i}{p_*^i}\right) & 0 \le p < p_*^i \\ P_{\min}^i & p_*^i \le p \le 1 \end{cases} \tag{1}$$

$$\mathbf{P}_{\text{dem}}^i = \left\{ P_{\text{dem}}^i | P_{\text{dem}}^i = f^i(p, p_*^i) \quad \forall p \in [0,1] \right\}$$

Collecting all the vectors mentioned above, the generic set of constraints $\mathcal{L}^i$ that a given HEMS i sends to the Aggregator will then be expressed as in (2).

$$\mathcal{L}^i = \left\{ \mathbf{E}_{\max}^i, \mathbf{E}_{\min}^i, \mathbf{P}_{\max}^i, \mathbf{P}_{\min}^i, \mathbf{P}_{\text{dem}}^i \right\} \tag{2}$$

The goal of the Aggregator is to obtain an aggregated representation of these constraints as:

$$\mathcal{L} = \left\{ \begin{array}{c} \mathbf{E}_{\max} = \sum_{i=1}^{H} \mathbf{E}_{\max}^i, \quad \mathbf{E}_{\min} = \sum_{i=1}^{H} \mathbf{E}_{\min}^i, \\ \mathbf{P}_{\max} = \sum_{i=1}^{H} \mathbf{P}_{\max}^i, \quad \mathbf{P}_{\min} = \sum_{i=1}^{H} \mathbf{P}_{\min}^i, \\ \mathbf{P}_{\text{dem}} = \sum_{i=1}^{H} \mathbf{P}_{\text{dem}}^i \end{array} \right\} \tag{3}$$

B. Step 2: Aggregated optimisation

Using the aggregated constraints in Step 1, the Aggregator is capable of optimising the flexibility use. The objective function might vary slightly depending on the target market (e.g. day-ahead, intra-day, imbalance, ancillary services, etc.). However, the problem is generically expressed as:

$$\begin{array}{ll} \min_{P(t)} \; C\left(P(t)\right), \text{ such that} & \\ \mathbf{P}_{\min}(t) \le P(t) \le \mathbf{P}_{\max}(t), & \forall t \in \{0, ..., T\}, \\ \mathbf{E}_{\min}(t) \le E(t) \le \mathbf{E}_{\max}(t), & \forall t \in \{0, ..., T\}, \\ E(t) = E(t-1) + P(t) \cdot \Delta t, & \forall t \in \{1, ..., T\} \end{array} \tag{4}$$

where $C(P(t))$ represents an arbitrary cost function for each aggregated power set point $P(t)$. This power is constrained by the aggregated $\mathbf{P}_{\max}$ and $\mathbf{P}_{\min}$ boundaries calculated in step 1. Likewise, the total energy $E(t)$ is also constrained by $\mathbf{E}_{\max}$ and $\mathbf{E}_{\min}$. An ideal battery model is assumed (4).

At this stage, only aggregated power and energy constraints are used. Thus, limited amount of sensitive information is revealed, as long as the number of aggregated units is significant enough, and provided reasonable values for individual inputs.

C. Step 3: Dispatch

From the optimal aggregated power profile $P(t)$ calculated in step 2, the Aggregator must dispatch the control signals for the individual assets. Following the MBC methodology, this is carried out using an equilibrium signal p_{eq}, which each HEMS can use for steering its local loads. The equilibrium set point is derived as in (5) using the aggregated power demand vector $\mathbf{P}_{\text{dem}}$ and the first aggregated dispatch power $P(t=1)$.

$$p_{eq} = \underset{p \in [0,1]}{\operatorname{argmin}} \left| \mathbf{P}_{\text{dem}}(p) - P(t=1) \right| \tag{5}$$

The inverse operation is performed locally by each HEMS, which maps p_{eq} into their individual demand constraints $\mathbf{P}_{\text{dem}}^i$. Since the aggregated equilibrium set point is shared with all HEMSs, no privacy issues exist. Thus, our goal is to ensure that Step 1 is carried out in a privacy-friendly fashion while maintaining the end-to-end functionality of the TSA.

III. Privacy-friendly aggregation algorithm

The TSA aggregation algorithm has been expanded with public-key cryptography and secret sharing in the form of additive random shares. In this way, the Aggregator has only access to the aggregated constraints $\mathcal{L}$ but does not learn anything about the individual data $\mathcal{L}^i$.

The solution relies on at least one intermediate 3rd party to perform a privacy-friendly aggregation. This role can be played by a non-trusted entity, as it would not learn anything from the data. Since the only tasks of these entities is to calculate partial additions, the name Adder will be used.

In a setup-phase, the Aggregator generates a public/private key-pair using the standardised RSA scheme [13]. The public-key is provisioned securely to each individual HEMS. Additionally, an important requirement of this solution is that the 3rd party/parties must not collude with the Aggregator.

A. Single 3rd party

The simplest approach is to use a single Adder. To guarantee privacy, every HEMS split its data into two parts. Since each HEMS represents its data as a set of constraints $\mathcal{L}^i$, this operation implies splitting every element of each vector. The notation $m^i[n]$ is used to indicate any element of a constraints vector for the i^{th} HEMS. The index $[n]$ indicates that an element-wise operation is performed. The random sampling and addition/subtraction operations on shares are done in the field $\bmod \left(2^{64}\right)$ for 64-bit inputs, with values that are scaled in an appropriate integer-range for the target use-case.

Since only the computation of the sum is needed one can efficiently achieve a privacy-friendly solution. Start by sampling a uniform random number $r^i[n]$. Then the element $m^i[n]$ is additively split into $(p_1^i[n], p_2^i[n])$ as

$$p_1^i[n] = r^i[n], \quad p_2^i[n] = m^i[n] - r^i[n] \tag{6}$$

In this way, two shares p_1^i and p_2^i are generated for each element m^i such that $p_1^i[n] + p_2^i[n] = m^i[n]$. Note, however, that a single of those shares holds no information on $m^i[n]$ (information-theoretical security). Next, one of these shares (e.g. p_2^i) is encrypted using the pre-provisioned public key from the Aggregator.

$$c^i[n] = \text{Enc}(p_2^i[n]) \tag{7}$$

Applying (6) and (7), each vector of constraints will be represented as two vectors, (i) a vector of plaintexts $p_1^i[n]$, and (ii) a vector of ciphertexts $c^i[n]$. Each of these vectors has the

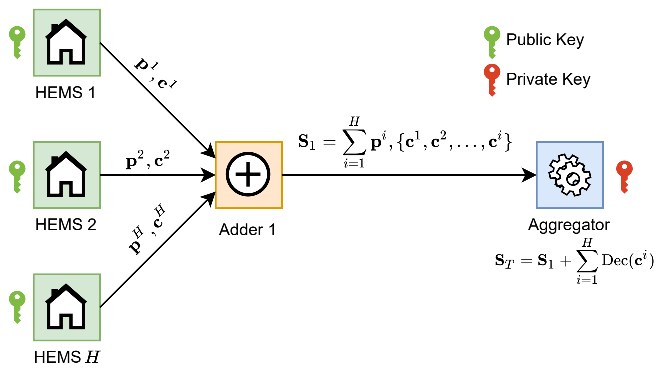

Public Key
Private Key
HEMS 1
HEMS 2
HEMS H
$\mathbf{p}^1, \mathbf{c}^1$
$\mathbf{p}^2, \mathbf{c}^2$
$\mathbf{p}^H, \mathbf{c}^H$
Adder 1
$\mathbf{S}_1 = \sum_{i=1}^{H} \mathbf{p}^i, \{\mathbf{c}^1, \mathbf{c}^2, \ldots, \mathbf{c}^i\}$
Aggregator
$\mathbf{S}_T = \mathbf{S}_1 + \sum_{i=1}^{H} \mathrm{Dec}(\mathbf{c}^i)$

Fig. 1. Secret sharing-based architecture with one 3rd party (Adder).

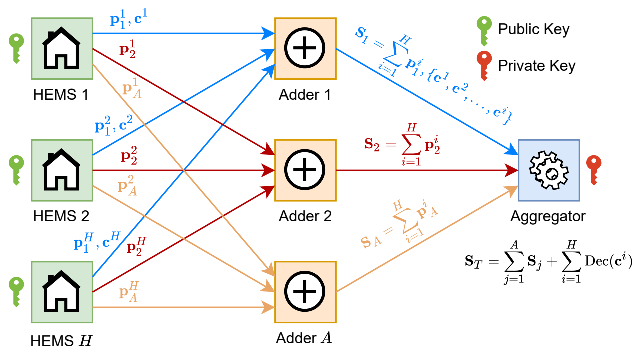

Public Key
Private Key
HEMS 1
HEMS 2
HEMS H
$\mathbf{p}_1^1, \mathbf{c}^1$
$\mathbf{p}_2^1$
$\mathbf{p}_A^1$
$\mathbf{p}_1^2, \mathbf{c}^2$
$\mathbf{p}_2^2$
$\mathbf{p}_A^2$
$\mathbf{p}_1^H, \mathbf{c}^H$
$\mathbf{p}_2^H$
$\mathbf{p}_A^H$
Adder 1
Adder 2
Adder A
$\mathbf{S}_1 = \sum_{i=1}^{H} \mathbf{p}_1^i, \{\mathbf{c}^1, \mathbf{c}^2, \ldots, \mathbf{c}^i\}$
$\mathbf{S}_2 = \sum_{i=1}^{H} \mathbf{p}_2^i$
$\mathbf{S}_A = \sum_{i=1}^{H} \mathbf{p}_A^i$
Aggregator
$\mathbf{S}_T = \sum_{j=1}^{A} \mathbf{S}_j + \sum_{i=1}^{H} \mathrm{Dec}(\mathbf{c}^i)$

Fig. 2. Secret sharing-based architecture generalised to A 3rd parties (Adders).

same dimensions as the original. Once this operation is carried out for every vector in the set $\mathcal{L}^i$, the plaintexts/ciphertexts vectors pairs are forwarded to the Adder as shown in Fig. 1.

Since the Adder does not have access to the private key of the Aggregator it cannot decrypt the ciphertext $c^i[n]$. Moreover, the Adder cannot infer any useful information from the individual $p_1^i[n]$ since these are just uniformly-distributed random values. The role of the Adder is to compute the sum $S[n]$ of the plaintext shares of all HEMS vectors as in (8) and forward the ciphertexts and this sum $S[n]$ to the Aggregator.

$$S[n] = \sum_{i=1}^{H} p_1^i[n] \tag{8}$$

Finally, the Aggregator receives for each constraints vector, the sum of the plaintext shares $S[n]$ and the set of ciphertexts $\{c^1[n], c^2[n], \ldots, c^H[n]\}$. Using its private key, the Aggregator then decrypts the ciphertexts and adds the corresponding plaintexts and eventually calculates the total sum $S_T[n]$ for each constraints vector.

$$S_T[n] = S[n] + \sum_{i=1}^{H} \mathrm{Dec}(c^i[n])$$

This aggregated constraints set is equivalent to the original $\mathcal{L}$, so Steps 2 and 3 of the TSA can be performed in the same way. Moreover, assuming the Aggregator did not collude with the Adder, neither will learn anything about the individual $\mathcal{L}^i$.

B. Multiple 3rd parties

The approach from Section III-A can be generalised to multiple Adders. Considering A as the number of available Adders, each HEMS i will split each element $m^i[n]$ into $A+1$ additive shares as indicated in (9). One of these shares is encrypted as illustrated in (10). The set of shares, $\{p_1^i, p_2^i, \ldots, p_{A+1}^i\}$, is exchanged as illustrated in Fig. 2.

$$\begin{aligned} p_j^i[n] &= r_j^i[n], \ \forall\, j = 1, \ldots, A, \\ p_{A+1}^i[n] &= m^i[n] - \sum_{j=1}^{A} r_j^i[n] \end{aligned} \tag{9}$$

$$c^i[n] = \mathrm{Enc}(p_{A+1}^i[n]) \tag{10}$$

This has two advantages. First, A independent Adders means the Aggregator would have to collude with all them to gain access to individual data. Furthermore, the additional encrypted share guarantees that even if Adders collude among them, they will not learn any individual constraints vectors.

The equations for the plaintext sum $S_a[n]$ in each Adder and the Aggregator sum $S_T[n]$ are given as (11) and (12).

$$S_a[n] = \sum_{i=1}^{H} p_a^i[n] \tag{11}$$

$$S_T[n] = \sum_{a=1}^{A} S_a[n] + \sum_{i=1}^{H} \mathrm{Dec}(c^i[n]) \tag{12}$$

It should be highlighted that the encryption can be applied to any of the $A+1$ shares. Moreover, the ciphertexts vectors can be forwarded directly to the Aggregator or to any Adder.

Finally, an interesting result of this generic formulation is its applicability to a scenario where instead of using A intermediate 3rd parties, their role is taken over by the HEMSs which implement a multi-party computation (MPC). This simply means that $A = H$. However, the challege here is to deploy the necessary communication channels among HEMSs.

IV. Evaluation

To assess the feasibility and performance of the privacy-friendly aggregation algorithm, a proof-of-concept (PoC) implementation was developed. The PoC is comprised of the three actors, namely, HEMSs, Adders and Aggregator.

The architecture is shown in Fig. 3. The entities communicate using the MQTT protocol secured with TLS. The topics on which each entity publishes are denoted by solid arrows, while subscriptions are dashed arrows. The client certificates limit the topics to which each entity has access.

Each entity was containerised using Docker. Subsequently, the entire solution was deployed on a Kubernetes cluster[1], where the algorithm parameters, as well as the number of entities (i.e. replicas) could be modified to simulate multiple HEMSs or Adders. Additionally, a time series database

[1]The Kubernetes cluster (v1.17.7) runs on Azure and uses the type *Standard_D2_v3* with 2 vCPU (Intel Xeon) and 8 GiB memory per VM.

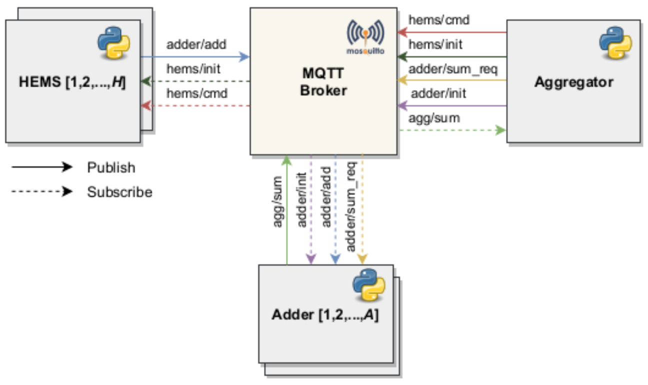


Fig. 3. Communication architecture between the elements in the PoC.

(Timescale DB) for capturing the algorithm results, a monitoring utility (Prometheus), and a graphical dashboard client (Grafana) were attached for complementary metrics.

A. Simulation scenario

The first step was to verify the system is operating as expected. A base scenario, consisting of 30 HEMSs that control the charging schedule of local EVs was selected for the test. Each EV was randomly assigned with a maximum capacity $[20-40$ kWh$]$ and a maximum charging rate $[3.3-6.6$ kW$]$. Likewise, for each simulated day an arbitrary target charge level and time of departure was selected. The goal of the optimisation at the Aggregator was charging at the cheapest cost when subjected to variable energy prices.

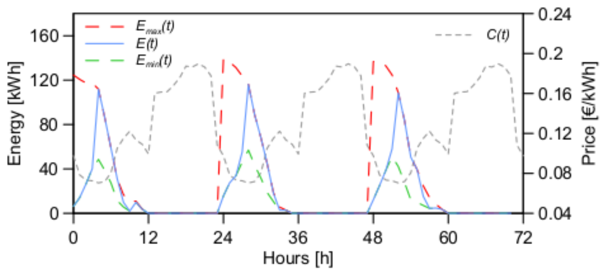
In Fig. 4, the recorded energy limits and trajectories for the Aggregator (a) and sample power profiles of three HEMSs (b) are shown. On the Aggregator side, we can verify that the aggregated maximum (red) and minimum (green) energy constraints are used and enforced during charging scheduled (blue). Moreover, these set points follow the varying energy tariff (grey dashed line), charging when the prices are low.

With regard to the HEMSs, their individual energy and power profiles and constraints can only be accessed locally and are unknown to the Aggregator or Adder. In Fig. 4(b), we selected only three HEMSs to illustrate this behaviour. In addition, the dashed grey line shows their aggregated trend. It can be seen that inferring the individual charging schedule of a particular HEMS is quite challenging even with just 3 units.

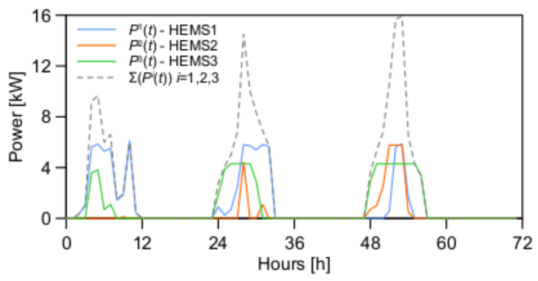
Finally, concerning the Adder, its sole purpose is to assist in the aggregation of the individual HEMS constraints, so the Aggregator can only learn the total sum. Nevertheless, the Adder does not learn anything about the underlying patterns or users' data. This is clearly demonstrated in Fig. 4(c), where the plaintexts sum for the minimum and maximum energy constraints is represented. As expected, the Adder only perceives random numbers from which it is impossible to infer any individual information.

B. Scalability and impact of time resolution

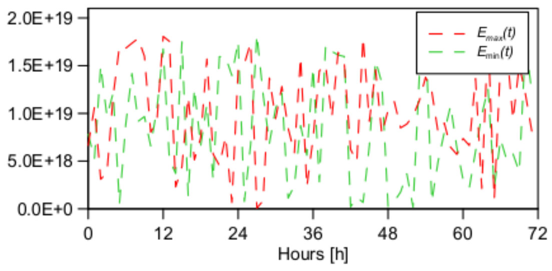
This approach is computationally demanding as it requires encryption and decryption operations along several vectors for each HEMS. Furthermore, the length of those vectors increases



(a) Aggregated energy limits and trajectory at the Aggregator.



(b) Individual charging patterns for three selected HEMSs.



(c) Plaintext sum calculated by the Adder.

Fig. 4. TSA for 30 HEMSs using the privacy-preserving scheme.

with the time horizon T_{hor}, step interval Δt, and priority resolution Δp selected for the optimisation. Considering (2), the number of encryption operations $\mathrm{n}\left[\mathrm{Enc}^i(x)\right]$ is given as:

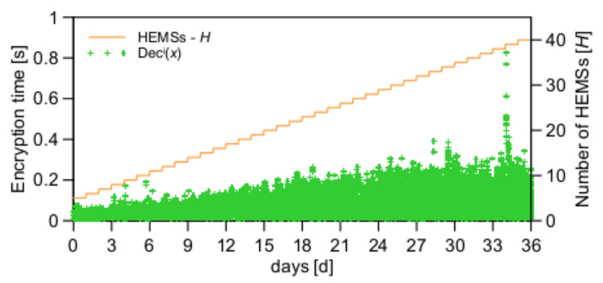
$$\mathrm{n}\left[\mathrm{Enc}^i(x)\right] = \left[\left(n_{\text{vec}} \times \frac{T_{\text{hor}}}{\Delta t}\right) + n_{\text{Evec}}\right] + \frac{1}{\Delta p} \qquad (13)$$

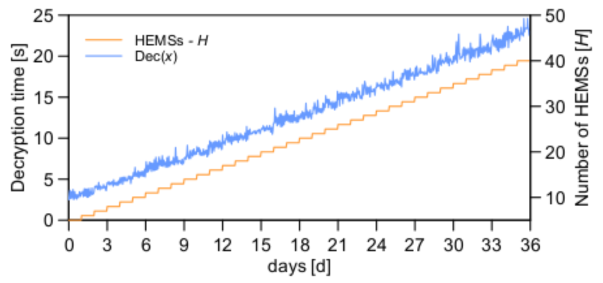
where n_{vec} is the number of energy and power constraints vectors, while n_{Evec} only considers the number of energy constraints vectors. Eq. (13) applies to every HEMS. Hence, the number of decryption operations performed by the Aggregator $\mathrm{n}\left[\mathrm{Dec}(c)\right]$ can be expressed as:

$$\mathrm{n}\left[\mathrm{Dec}(c)\right] = H \times \mathrm{n}\left[\mathrm{Enc}^i(x)\right]$$

Fig. 5 and Fig. 6 show encryption and decryption times when increasing the number of HEMSs and decreasing the step interval Δt. The time horizon (T_{hor}) was fixed to 24 hours, and the priority resolution (Δp) equals 0.01.

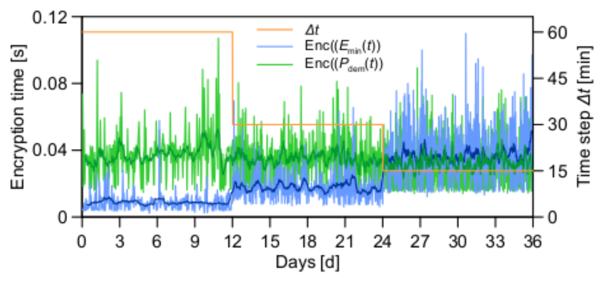
Fig. 5(a) demonstrates that increasing the number of HEMSs (orange) does not affect the individual encryption times at each HEMS since these are independent units. Regarding the decryption at the Aggregator (b), a linear increment in the

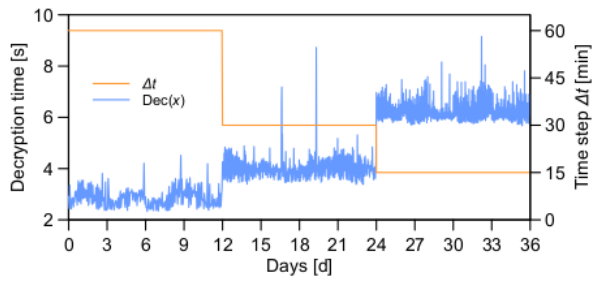




(b) Decryption wall-clock time for the Aggregator.

Fig. 5. En/Decryption operations when increasing the number of HEMSs (one extra HEMS is added daily).





(b) Decryption wall-clock time for the Aggregator.

Fig. 6. En/Decryption operations when decreasing the step interval Δt (simulation of 12 days per interval [60, 30, and 15 minutes]).

wall-clock time (blue) is observed as the number of HEMSs increases (orange).

On the other hand, decreasing the step interval significantly increases the encryption time of the energy and power constraints vectors (blue) as shown in Fig. 6(a). The encryption time of the power demand vector (green) remains constant as its resolution depends on Δp. Considering the decryption time (b) an increment is observed again.

Fig. 6(b) showcases one of the biggest challenges of this algorithm. Reducing the time interval Δt does not only increase the decryption time, but also reduces the available time for performing such operations. For instance, the decryption of 5 HEMSs with 1-hour resolution takes an average of 2.8 seconds, meaning that each hour more than 1200 HEMSs could be aggregated. However, when the time interval is reduced to 15 minutes, the decryption time increases up to 6.3 seconds, which reduces the aggregation possibilities to approximately 140 HEMSs every 15 minutes.

V. Conclusions and Future Work

This privacy-friendly aggregation algorithm based on the TSA offers the same benefits and scalability as the original solution. However, it drastically limits the amount of data that the Aggregator can learn from the individual users.

One or more additional 3rd Adder(s) and public-key operations are needed, which result in additional computational requirements. Nevertheless, well-established public-key algorithms such as RSA are easier to implement and more efficient that other common privacy techniques such as FHE. Moreover, the simplicity and non-trusted nature of the Adders offers a wide range of alternatives for its deployment.

Currently, alternatives without intermediate 3rd parties are also being studied, such as multi-party computation among the participant HEMSs, as introduced at the end of Section III.

References

[1] B. Li, J. Shen, X. Wang, and C. Jiang, "From controllable loads to generalized demand-side resources: A review on developments of demand-side resources," *Renewable and Sustainable Energy Reviews*, vol. 53, pp. 936–944, Jan 2016.

[2] R. D'hulst, W. Labeeuw, B. Beusen, S. Claessens, G. Deconinck, and K. Vanthournout, "Demand response flexibility and flexibility potential of residential smart appliances: Experiences from large pilot test in Belgium," *Applied Energy*, vol. 155, pp. 79–90, Oct 2015.

[3] G. W. Hart, "Nonintrusive appliance load monitoring," *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, 1992.

[4] M. Afzalan and F. Jazizadeh, "A Machine Learning Framework to Infer Time-of-Use of Flexible Loads: Resident Behavior Learning for Demand Response," *IEEE Access*, vol. 8, pp. 111 718–111 730, 2020.

[5] N. O'Connell, P. Pinson, H. Madsen, and M. O'Malley, "Benefits and challenges of electrical demand response: A critical review," *Renewable and Sustainable Energy Reviews*, vol. 39, pp. 686–699, Nov 2014.

[6] "Regulation (EU) 2016/679 of the European Parliament and of the Council," *Official Journal of the European Union*, vol. L119/1, pp. 1–88, 2016. [Online]. Available: http://data.europa.eu/eli/reg/2016/679/oj

[7] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of secure computation*, vol. 4, no. 11, pp. 169–180, 1978.

[8] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *ACM Symposium on Theory of Computing – STOC*, M. Mitzenmacher, Ed. ACM, 2009, pp. 169–178.

[9] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987.

[10] V. S. Miller, "Use of elliptic curves in cryptography," in *Crypto 1985*, ser. LNCS, H. C. Williams, Ed., vol. 218. Springer, Heidelberg, 1986, pp. 417–426.

[11] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks," in *CRYPTO*, ser. LNCS, M. J. Wiener, Ed., vol. 1666. Springer, 1999, pp. 398–412.

[12] S. Vandael, B. Claessens, M. Hommelberg, T. Holvoet, and G. Deconinck, "A scalable three-step approach for demand side management of plug-in hybrid vehicles," *IEEE Transactions on Smart Grid*, vol. 4, no. 2, pp. 720–728, 2013.

[13] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, pp. 120–126, 1978.