Side Channel Security of Smart Meter Data Compression Techniques

Marcell Fehér[1], Niloofar Yazdani[1], Diego F. Aranha[1], Daniel E. Lucani[1]
Morten Tranberg Hansen[2], Flemming Enevold Vester[2],
[1]Agile Cloud Lab, Department of Engineering, DIGIT, Aarhus University, Aarhus, Denmark
[2] Kamstrup A/S, Skanderborg, Denmark
{sw0rdf1sh,n.yazdani,dfaranha,daniel.lucani}@eng.au.dk, {mtr,flev}@kamstrup.com

***Abstract*—Given the large and sustained growth in the number of smart meters for different applications, e.g., electricity, water, heat, effective data compression has become increasingly important. Although smart meters tend to encrypt payloads using state-of-the-art solutions, the packet length variability introduced by compression of the data can be exploited in a side channel attack to gain knowledge about the consumption of individual meters. In a nutshell, a meter reporting zero (or constant) consumption can be compressed more than one reporting more erratic consumption. An attacker may gain knowledge of behavioral patterns of a household, e.g., when is no one home, or company, e.g., active periods of production. This paper analyzes the correlation between packet length and reported consumption of several signals and practical reporting periods for the DLMS standard using real (anonymized) smart meter measurements. We consider various built-in compressors and also propose new techniques that can both increase the compression and reduce this correlation. Our proposed schemes are particularly well suited for the increasingly popular case of high frequency reporting, e.g., reporting each measurement as it becomes available.**

***Index Terms*—compression, DLMS, IoT, generalized deduplication, smart meters, side channel, information leak**

I. Introduction

In the last decade most utility providers have changed the way they collect consumption data from infrequent manual readings to automatic upload using Smart Meters. Installing these connected devices to residential and industrial customers allows the provider to monitor usage at a much finer granularity and receive frequent readings. The opportunities enabled by this high detailed information are game-changing: providers can get feedback on the quality of their service including near real time alerts on anomalies and power outages, as well as drastically improved fraud detection and demand forecasting, just to name a few.

These new business advantages come at a price. Providers now have to manage thousands to millions of Smart Meters, provide adequate upload bandwidth and collect, store and analyze the received data. To reduce the size of readings, the protocol supports data compression. Since the meters are usually connected via a low-bitrate technology like 2G cellular network, savings add up to a tremendous amount for the whole fleet, and providers can rarely afford to turn compression off. Although the protocol includes adequate encryption, some of the compressors are found to leak information via a side channel. When an external actor is able to observe the size of uploaded data packets of readings, they might be able to infer information about the consumption.

In this paper we analyze the correlation of measured consumption and the size of compressed readings data packets under different conditions. We later propose three new compression techniques that provide a better trade-off between privacy and bandwidth savings than the built-in DLMS/COSEM Null and Delta compression schemes. The rest of the paper is organized as follows. After giving an overview of recent efforts to preserve privacy of Smart Meter data in Section II, we introduce the readings transfer protocol, its implications on uncompressed packet size, and discuss the built-in Null and Delta compression schemes in Section III. Section IV describes the novel compression techniques we are proposing to mitigate, and in some cases eliminate the information leak. Section V shows the analysis of all discussed compressors, and Section VI draws the conclusions.

II. Related Work

Data collected by smart meters can be used to violate user privacy, e.g., by revealing sensitive energy consumption patterns that can indicate household occupancy at different times in the day or even economic status of its inhabitants. A recent survey [1] explores legal issues, notions of privacy and requirements for privacy-preserving smart meter data collection, management and processing. Two main threat models are considered: (i) the *trusted operator model*, in which data is collected and stored in a centralized information infrastructure (Meter Data Management System – MDMS) managed by the operator, who is trusted; (ii) the *non-trusted operator model*, in which the service provider and collaborators with access to the data can behave adversarially. The first model is fundamentally weaker, and forces adversaries to infer private information from external sources such as observable network traffic, tampered smart meters or compromised infrastructure. However, it captures currently deployed infrastructures and simplifies billing, since the operator has access to consumption information in plaintext. The latter model captures the complexity of modern energy ecosystems, but complicates operational aspects. The authors also make a distinction between *cryptographic privacy*, where privacy mechanisms are enforced by cryptographic techniques that limit what can be learned from smart meter data; and *statistical privacy* which limits what can be inferred about an individual who contributes sensitive information to a larger data set.

In terms of cryptographic privacy, most of the solutions for smart metering privacy in the trusted operator model are concentrated in applications of *homomorphic encryption* [2]–[4]. In such cryptosystems, operations performed over data encrypted under the same public key correspond to closely-related operations performed over the underlying plaintexts. The biggest advantage of such solutions is that data utility is preserved, with the downside of a non-trivial performance penalty due to computationally intensive public-key cryptography operations. Privacy in the non-trusted operator model can be obtained by aggregating data among meters before delivery, which may create scalability issues and resilience problems in case of failure [5]. Moreover, a third party is necessary for distributing secrets beforehand.

Statistically privacy techniques attempt to hide data patterns, compromising some of the data utility. The simplest proposal in this class is to employ rechargeable batteries [6], [7] which physically introduce noise as an increase or decrease in consumption, and limit the possibility of load monitoring at a non-trivial (environmental) cost. Sankar et al. [8] proposes a theoretical framework based on mutual information as a privacy metric, where utility is estimated by a mean-square error distance metric. A major shortcoming shared by these works is their lack of formal guarantees in terms of a proper privacy-accuracy tradeoff, relying on heuristic analysis instead. The *differential privacy* definition solves this concern by providing quantitative methods to formally assess the privacy loss of contributing to a data set. Differentially private solutions represent the state of the art in the non-trusted operator model [9], but incur increases in bandwidth consumption to hide periods of low demand, and might marginally penalize consumers due to lower accuracy.

We further note that the cryptographic and statistical privacy notions are complementary, and must be addressed jointly. Cryptographic mechanisms can still leak side-channel information [10], and statistical methods might reveal too much information with consecutive queries. In that sense, data confidentiality becomes a challenge, as pointed out in [1]: "*An interesting related issue is whether data access patterns at the MDMS or communication patterns between a meter and the MDMS could reveal private information about a consumer even if the data are encrypted. We are not aware of a reported privacy breach of this kind (...)*". Our paper discusses exactly this problem in the context of smart meter data compression methods assuming encrypted communication, and appears to be the first to address it in a realistic setting.

III. Motivation

The de-facto protocol for information exchange between Smart Meters and the Head End System (HES, the back-end system that manages the meters) is called Device Language Message Specification (DLMS). The utility provider can create so-called *Load Profiles*, which define what quantities should be measured by the meters at what time intervals (typically 15 minutes), and the trigger condition(s) when the collected readings should be uploaded. Usually there are multiple profiles active at the same time, allowing the provider to monitor different qualities of the service at different rates, or instruct the meter to upload a very detailed timeline of events that happened just before an anomaly.

The DLMS protocol also includes a message format which meters must use when compiling a number of readings into a data packet called an Application Data Unit (APDU). The vendor of the Smart Meter system or the utility provider using it is allowed to customize the message format, and add or remove components of APDUs as they wish. The real life dataset we analyze in this paper has been provided by Kamstrup A/S, a major Smart Meter manufacturer, who has extended the standard APDU format with a header that describes the Load Profile which the APDU belongs to. This makes the packets self describing, simplifying the process in the HES to identify and categorize them quickly.

When an APDU is constructed by a meter, the DLMS protocol dictates that every reading is encoded to a fixed length segment in the binary data packet. Since the header is also the same for each message of the same Load Profile, the total size of uncompressed APDUs is constant. This is excellent for user privacy, since the APDU size does not reveal anything about the values reported in it, but requires a lot of bandwidth. To help this problem, DLMS includes a number of data compression options.

The *Null Data* compression scheme replaces values with a null marker byte if it is identical or predictable from the previous record. For example, if the timestamp is increased at a constant rate between data records, only the first reading will include the full 14-byte value. Every other will only have a single-byte null marker. Similarly, when the power consumption does not change between measurements, Null Data replaces the full values with one byte. This makes the compressed APDU size highly dependent on the underlying compression readings, leading to compromised user privacy.

A different method, *Delta Array compression* aims to reduce APDU size by decreasing the number of bytes for values that can be represented shorter than the data type would prescribe. For example, the default data type for active energy takes 4 bytes. If the actual value in a particular measurement fits into a single byte, the Delta Array compressor changes the type code preceding the value in order to use only one byte to encode the measurement, thus, saving three bytes. While Null Data causes the compressed APDU size to change every time consecutive readings of a quantity are identical, Delta Array signals more significant changes in the consumption, when the latter value cannot be represented at the same length.

IV. Proposed Compression Schemes

As either alternatives or additives to the built-in methods, we propose three new compression schemes that strike a good balance between size reduction and privacy protection. All of them can be applied to the transferred APDUs transparently for the existing meter and HES software components, since the compressor works on the output APDUs of the meter and

Fig. 1: An example APDU that consists of four readings of four quantities. Total size: 527B, header (blue): 331B (63%), type codes (orange): 44B (8%), data (black): 152B (29%).

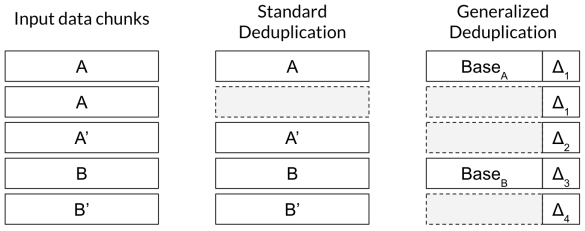

Fig. 2: Standard and Generalized Deduplication

the decompressor losslessly reconstructs the original APDU before the HES receives it.

A. Header Hash Replacement

The header that makes APDUs self describing and the HES simpler is relatively large if there's a small number of readings being transmitted in a message. For a Load Profile that measures four quantities produces a 527-byte uncompressed APDU when only four readings are encoded together, out of which only 152 bytes contain the actual measured values, the rest is the header (331B) and the type information (44B), as illustrated by Fig 1.

Since the header is constant for each Load Profile, it is not necessary to send it with every message. Instead, we propose replacing it with its hash digest. The decompressor would simply read the hash value, look up the corresponding Load Profile from a dictionary and inject it to the APDU before forwarding it to the Head End System. The size of this lookup table stays under 360 kB even if we assume a relatively large number of 1000 different Load Profiles, and SHA-256, a secure hash function. When used with uncompressed APDUs this technique preserves user privacy completely as it produces constant-size APDUs. This is due to the fact that the fixed size header is replaced by a fixed-size hash, which does not reveal any information about consumption to an observer. The achieved compression varies widely, e.g., from 86 to 11%, depending on the number of readings in an APDU (min: 1, max: 48) and checksum used (min: CRC32, max: SHA-256).

B. Generalized Deduplication-based Schemes

Deduplication is a well-known compression technique, which eliminates identical chunks in the input data. Recently a new, more general variant of the traditonal method has been introduced [11], which is able to compress not only identical but also similar data chunks. The key idea of the

Fig. 3: Block compressor using Generalized Deduplication

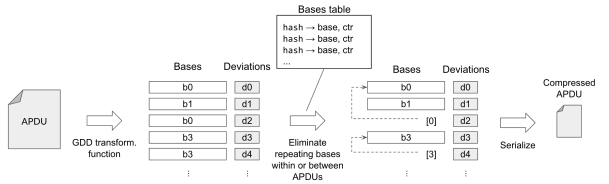

Fig. 4: Stream compressor using Generalized Deduplication

new technique called Generalized Deduplication (GD) is to decompose each data chunk to a pair of *basis* and a small *deviation* in a way that 1) the original data chunk can be reconstructed perfectly, and 2) similar chunks are mapped to the same basis (See Fig 2). The core of this technique is the transformation function that converts a data chunk to a basis-deviation pair during the compression phase, and reconstructs the original chunk from the pair in the decompressor. Error correcting codes like Hamming and Reed-Solomon have shown promising results with IoT data, and new algorithms are also being designed specifically for GD. Based on the concept of Generalized Deduplication, we propose two concrete compression schemes for DLMS APDUs.

Block Compression with GD: First, we propose using Generalized Deduplication to compress each APDU independently. This simple algorithm converts chunks of an APDU to a series of base-deviation pairs, and replaces the repeating bases by the index of the first chunk where they appeared (see Fig 3) while preserving the ordering of chunks. The decompressor finds the referenced base by chunk index, and passes each base-deviation pair to the inverse transformation function, which reconstructs the original data chunk. Due to the simplicity of this technique, both the compressor in the Smart Meter and decompressor in the HES are fast and require very low amount of memory.

Stream Compression with GD: Since there is a lot of similiarity between readings across APDUs, we can reduce transfer size even further by leveraging information that has been sent previously. The GD-based stream compressor does exactly that, as an extension to the block mode. When it encounters a base that is new in the current APDU but has already been sent in a previous one, it replaces the base with its hash digest (see Fig 4). Using hash to identify bases instead a numeric index serves two purposes. First, since Smart Meters do not stop sending readings for a very long time, the number of unique bases in the total lifetime of the stream may grow very high, requiring a long data type for these cross-APDU base indexes from the very beginning. Second, since hashes identify APDUs across multiple Meters, the decompressor at the Head End System can use a single

lookup table of hashes and bases for all meters. If we would be using numeric indexing, the decompressor would have to keep track of the indexes separately for each meter.

V. Evaluation

A. Evaluation metrics

The aim of our paper is to analyze different APDU compression schemes in terms of compression ratio and amount of information leaked by the packet size. We consider a passive attacker who can observe compressed APDUs when they are uploaded from Smart Meters to the Head End System, but cannot decrypt and decompress them, or alter the contents of the messages. The attacker is assumed to know that the APDUs contain measurements of additive units like kilowatt-hour (kWh), where the consumption of a time period can be obtained by subtracting the measured value at the beginning from the value at the end of the time window.

Compression ratio: The compression ratio is calculated by dividing the compressed APDU size by the original size. Accordingly, compression ratio ranges from 0 to 1 where higher values are better.

Information leakage: The purpose is to measure how much information about electricity consumption could be revealed by observing the compressed APDU sizes. To this end, we calculate the multiple correlation coefficient between the measured consumption (4 predictor variables) and the changes between consecutive APDU sizes (1 dependent variable). Higher correlation means that more information about consumption can be inferred from the observed APDU sizes. When an APDU is constructed, the meter encodes the accumulated value of the measured units every 15 minutes. For example, if 0.4 kWh was consumed evenly during one hour and reading at the beginning of the hour was 40.0 kWh, then the subsequent readings contain 40.1, 40.2, 40.3 and 40.4 kWh values. We derive our predictor variables from these signals by calculating the difference of subsequent readings, e.g: 0.1, 0.1, 0.1, 0.1 kWh. The dependent variable is also the derived from the list of compressed APDU sizes by taking the difference between the current and previous APDU.

Since the APDUs in our test dataset measure four signals at once, we use *Multiple correlation coefficient* (R) [12], [13] to quantify the connection between the observed APDU sizes and the underlying electricity consumption. This statistical method generalizes the standard coefficient correlation to having more than one independent variable. R ranges from 0 to 1, where higher values mean stronger relationship.

It is common to refer to this coefficient as the squared multiple correlation coefficient (R^2). Since value of R^2 is dependent on the number of independent variables, it sometimes overestimate the population correlation. *Adjusted* R^2, denoted by R^2_{adj} was introduced to correct for this overestimation. We report both R and R^2_{adj} values for every compression scheme that we evaluated.

Let us assume $\{X_1, X_2, ..., X_j, ..., X_J\}$ and y is a set of J independent variables and the dependent variable, respectively. X and y are defined as the augmented matrix collecting the data for the independent variables and the vector of observations for the dependent variables, respectively, as:

$$X = \begin{bmatrix} 1 & x_{1,1} & \dots & x_{1,j} & \dots & x_{1,J} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & \dots & x_{n,j} & \dots & x_{n,J} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & \dots & x_{N,j} & \dots & x_{N,J} \end{bmatrix}, y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \\ \vdots \\ y_N \end{bmatrix}, \quad (1)$$

where N denotes the number of samples. The vector $\hat{y} = Xb$ with $b = \left(X^T X\right)^{-1} X^T y$ contains the predicted values of the dependent variable. The regression sum of squares is

$$SS_{regression} = b^T X^T y - \frac{1}{N}(1^T y)^2, \quad (2)$$

where 1^T is a row vector of $1'$s conformable with y. The total sum of squares is

$$SS_{total} = y^T y = \frac{1}{N}(1^T y)^2. \quad (3)$$

Accordingly, the squared multiple correlation coefficient and the adjusted squared multiple correlation coefficient are

$$R^2 = \frac{SS_{regression}}{SS_{total}}, \quad (4)$$

$$R^2_{adj} = 1 - [(1 - R^2)\big(\frac{N-1}{N-J-1}\big)]. \quad (5)$$

Although several different formulas for R^2_{adj} exist, this is one most commonly used. Rules-of-thumb [14] determine the minimum number of samples to conduct multiple and partial correlation. In particular, the minimum number of samples to have an accurate multiple correlation coefficient is $50 + 8 \cdot J$.

B. Numerical results

We evaluate the performance of several compression methods using the smart meter readings provided by Kamstrup A/S. We use a CSV file containing 8400 measurements of four signals with the resolution of 15 minutes. Hence, we will have 4 independent variables. We evaluate the performance considering six reporting time periods of $1, 2, 4, 12, 24$ and 48 readings per APDU. For instance, since the meter is recording once every 15 minutes, reporting time period of 2 means having 2 measurements in an APDU and can be interpreted as converting and uploading the buffer of a meter once every half an hour. Currently most utility providers collect readings once or twice a day, and wish to increase this granulariy.

We compare our techniques, i.e., header hash replacement, GD block and stream compression and their combinations, with the built-in compression methods Null, Delta and both of them applied together Furthermore, we compare with uncompressed scheme as well as the standard compression scheme LZW [15]. We apply LZW to each APDU, separately.

For GD, we apply Hamming and Reed-Solomon codes to create the mapping from a chunk to the basis-deviation pair. In particular, we use a range of Hamming codes with 5 to 9 parity bits. The number of parity bits determines the chunk size. We use Reed-Solomon codes over the finite field of 2^8 with the error correction capability of 1 byte and chunk sizes

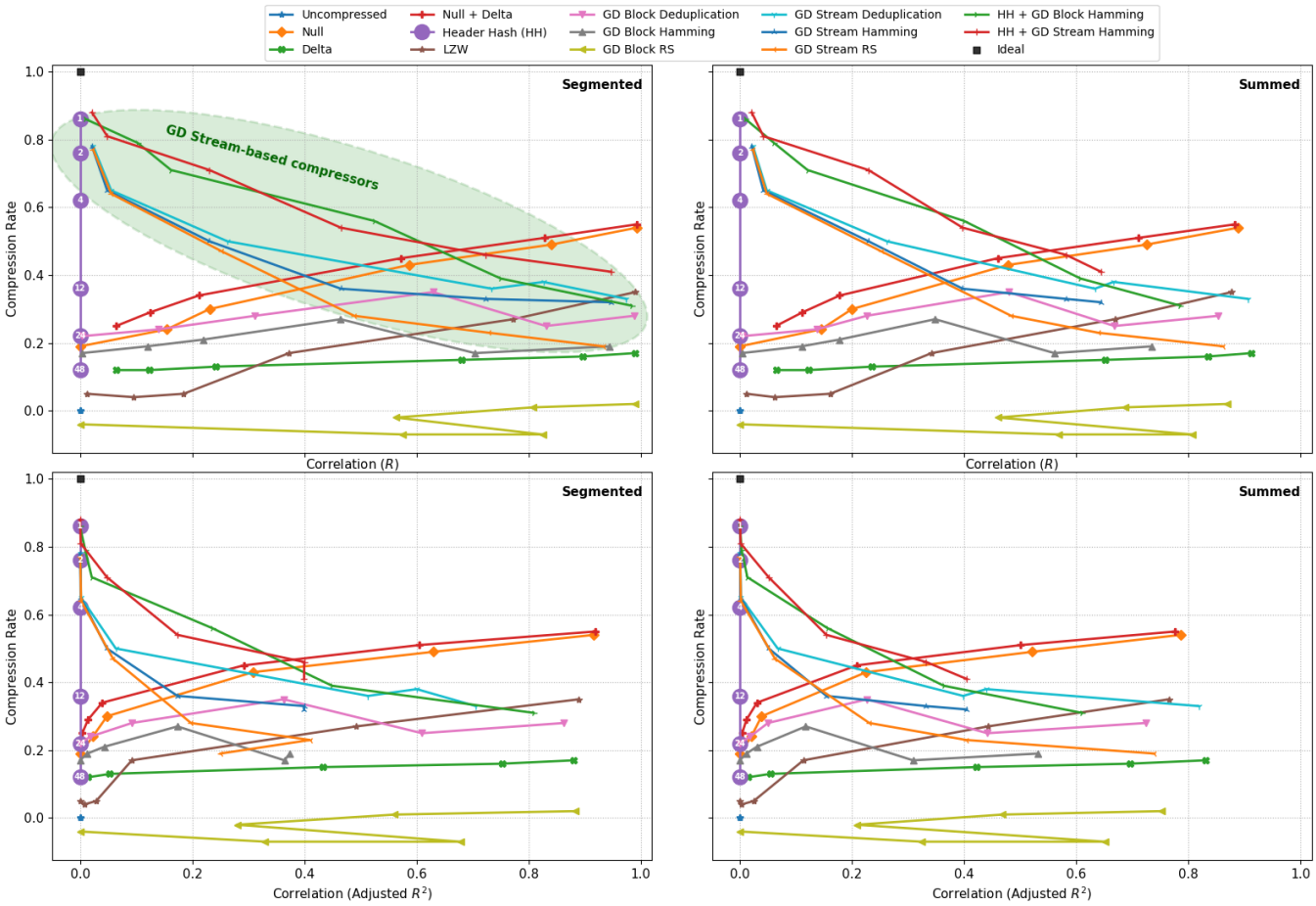

Fig. 5: Compression ratio and correlation of different compression schemes over different reporting time periods.

of $8, 16, 32, 50, 64$, and 100 bytes. For measuring compression ratio with a GD-based compression scheme, we measure the compression ratio for all different DD, Hamming and Reed-Solomon configurations and report the best result of each.

Correlation coefficient cannot be calculated in case of having a constant dependent or independent variable. If the APDU sizes are constant such as uncompressed scheme or header hash, no information can be inferred from the APDU sizes. Thus, we assign the correlation coefficient of 0 for the case of having a constant APDU size. If a measurement signal is constant, i.e., the differential samples are constant and equal to 0, the changes in this signal has no effect in the APDU size changes and we can safely remove the signal. In our data set, one of the four signals is constant, which we do not consider it in multiple correlation coefficient calculation.

To calculate the multiple correlation coefficient, the number of samples in the dependent and independent variables must be equal. For reporting time period of 1, each APDU contains a single set of measurements, i.e., 1 measurement of each signal. Thus, the number of samples in the dependent and independent variables are equal. However, for reporting time period of 2, each APDU maintains 2 sets of measurements causing the number of samples in the dependent variable to be half of the number of samples in the independent variables. To solve this issue for reporting time period of 2, we expand each independent variable into 2 independent variables, each containing half of the samples. One of them contains the odd samples and the other one contains the even samples. Then, we will have $2 \cdot 3 = 6$ independent samples for reporting time period of 2 readings/APDU. Similarly, for reporting time period of x, we have $x \cdot 3$ independent variables. We call this scheme, *segmented* approach, hereafter, due to splitting each independent variable into multiple independent variables.

As previously mentioned, our data set contains 8400 measurements per signal. According to the rules-of-thumb, we do not have enough samples per variable for reporting time periods of 24 and 48. For instance, for reporting time period of 24, we have $24 \cdot 3 = 72$ ($J = 72$) independent variables. And, we need a minimum of $50 + 8 \cdot 72 = 626$ samples per variable; while, we only have $\frac{8400}{24} = 200$ samples per variable. This means that the calculated multiple correlation coefficient is not reliable for 24 and 48 reporting time periods.

To overcome this problem, we define another scheme called *summed* approach. Rather than expanding each independent

variable into multiple independent variables, we sum the consecutive differential values in the same time window that is covered by the APDU, for each independent variable. Thus, the number of independent variables is always equal to 3 ($J = 3$) for all the reporting time periods which implies the number of samples is always greater than $50 + 8 \cdot 3 = 74$.

Fig. 5 demonstrates compression ratio and multiple correlation coefficient for different compression schemes over different reporting time periods for segmented and summed approaches. The black square at the top left is the theoretical ideal case where the multiple correlation coefficient is 0 (e.g. no information can be inferred) and the compression ratio is 1 (e.g. every APDU is compressed to zero bytes). The closer to the ideal case translates to a better performance.

Each compression scheme's curve has 6 markers on it. From left to right, these markers belong to reporting time periods of $1, 2, 4, 12, 24$ and 48 except for header hash and uncompressed approaches. Header hash replaces the header with a fixed size hash digest. Thus, the size of the compressed APDU is constant and the multiple correlation coefficient is considered 0, and, its curve has only vertical changes. The numbers on its curve demonstrate the reporting time periods. As in uncompressed scheme, we do not have either compression or correlation, it is always fixed in $(0, 0)$.

The plots of Fig. 5 clearly identify two big trends: one for most algorithms and one for some of our approaches. Our methods with GD stream-based compressors and also our header hash scheme get better in terms of both compression and correlation with more frequent uploads (smaller reporting time periods), while the built-in ones get worse in terms of compression. For instance, header hash plus GD stream (Hamming) has a correlation of 0.02 and compression ratio of 0.88 for reporting time period of 1. Header hash scheme, for reporting time period of 1, corresponds to correlation of 0 and compression ratio of 0.86. While for higher reporting time periods, our schemes provide a little bit less compression compared to Null or Null plus Delta, but, they provide less correlation and hence higher security. For reporting time period of 48, summed scheme, Null plus Delta has R^2_{adj} of 0.77 while GD stream (Hamming) and header hash plus GD stream (Hamming) correspond to R^2_{adj} of as low as 0.40.

While GD-based compressors provide slightly lower compression with Hamming compared to standard deduplication, but, GD-based compressors with Hamming gives a significant higher security. Considering summed technique and reporting time period of 48, R^2_{adj} is equal to 0.82 and 0.40 for GD stream Dedup and GD stream Hamming, respectively. This shows that the Header Hash Replacement and GD stream-based techniques correspond to significantly higher compression and privacy compared to the built-in DLMS methods, when the reporting time period is reduced to an hour or less.

VI. Summary

This paper analyzed the correlation between the packet lengths resulting from data compression and the consumption reported in the payload of those packets. To the best of our knowledge, this is the first analysis of the kind for smart meter data and, particularly, for the DLMS protocol. We analyze this performance of standard compressors in DLMS, e.g., Delta, Null, as well as well-known lightweight compressors, e.g., LZW. These solutions have a common trend: the fewer measurements reported per packet, the less correlation but also the less compression potential. Beyond these state-of-the-art solutions, we propose new families of compressors that provide a better compression-correlation trade-off. In fact, we show that more frequent reporting (fewer measurements per packet) with our schemes actually yields substantially better compression and lower correlation. Thus, our proposals can open the door to higher frequency reporting, which is highly demanded by utility providers, by reducing the transmission costs and reducing the potential for the studied side channel attack. Future work will focus on developing (dynamic) mechanisms that can maintain a similar compression levels, but that reduce even further the correlation between packet length and consumption. We will consider combinations of current proposals as well as techniques to randomize packet lenghts or, alternatively, make the packet lenghts constant.

References

[1] M. R. Asghar, G. Dán, D. Miorandi, and I. Chlamtac, "Smart meter data privacy: A survey," *IEEE Comm. Surveys & Tutorials*, vol. 19, no. 4, pp. 2820–2835, 2017.

[2] F. D. Garcia and B. Jacobs, "Privacy-friendly energy-metering via homomorphic encryption," in *Int. Workshop on Security and Trust Management*. Springer, 2010, pp. 226–238.

[3] Z. Erkin and G. Tsudik, "Private computation of spatial and temporal power consumption with smart meters," in *Int. Conf. on Applied Cryptography and Network Security*. Springer, 2012, pp. 561–577.

[4] N. Busom, R. Petrlic, F. Sebé, C. Sorge, and M. Valls, "Efficient smart metering based on homomorphic encryption," *Comp. Communications*, vol. 82, pp. 95–101, 2016.

[5] N. Buescher, S. Boukoros, S. Bauregger, and S. Katzenbeisser, "Two is not enough: Privacy assessment of aggregation schemes in smart metering," *Privacy Enhancing Tech.*, vol. 2017, no. 4, pp. 198–214, 2017.

[6] S. McLaughlin, P. McDaniel, and W. Aiello, "Protecting consumer privacy from electric load monitoring," in *ACM Conf. on Computer and Communications Security*, 2011, pp. 87–98.

[7] M. Backes and S. Meiser, "Differentially private smart metering with battery recharging," in *Data Privacy Management and Autonomous Spontaneous Security*. Springer, 2013, pp. 194–212.

[8] L. Sankar, S. R. Rajagopalan, S. Mohajer, and H. V. Poor, "Smart meter privacy: A theoretical framework," *IEEE Trans. on Smart Grid*, vol. 4, no. 2, pp. 837–846, 2012.

[9] P. Barbosa, A. Brito, and H. Almeida, "A technique to provide differential privacy for appliance usage in smart metering," *Information Sciences*, vol. 370, pp. 355–367, 2016.

[10] A. M. White, A. R. Matthews, K. Z. Snow, and F. Monrose, "Phonotactic reconstruction of encrypted voip conversations: Hookt on fon-iks," in *IEEE Symposium on Security and Privacy*. IEEE, 2011, pp. 3–18.

[11] R. Vestergaard, D. E. Lucani, and Q. Zhang, "Generalized deduplication: Lossless compression for large amounts of small IoT data," in *European Wireless Conference*. VDE, 2019, pp. 1–5.

[12] H. Abdi, "Multiple correlation coefficient," *The University of Texas at Dallas*, pp. 648–651, 2007.

[13] A. G. Bluman, *Elementary statistics: A step by step approach*. McGraw-Hill Higher Education New York, NY, 2009.

[14] S. B. Green, "How many subjects does it take to do a regression analysis," *Multivariate behavioral research*, vol. 26, no. 3, pp. 499–510, 1991.

[15] T. A. Welch, "A technique for high-performance data compression," *Computer*, vol. 17, no. 6, p. 8–19, Jun. 1984. [Online]. Available: https://doi.org/10.1109/MC.1984.1659158