Compromised through Compression
Privacy Implications of Smart Meter Traffic Analysis

Pol Van Aubel and Erik Poll

Digital Security group, Institute for Computing & Information Sciences
Radboud University, the Netherlands
pol.vanaubel@cs.ru.nl erikpoll@cs.ru.nl

Abstract. Smart metering comes with risks to privacy. One concern is the possibility of an attacker seeing the traffic that reports the energy use of a household and deriving private information from that. Encryption helps to mask the actual energy measurements, but is not sufficient to cover all risks. One aspect which has yet gone unexplored – and where encryption does not help – is traffic analysis, i.e. whether the length of messages communicating energy measurements can leak privacy-sensitive information to an observer. In this paper we examine whether using encodings or compression for smart metering data could potentially leak information about household energy use. Our analysis is based on the real-world energy use data of ±80 Dutch households.
We find that traffic analysis could reveal information about the energy use of individual households if compression is used. As a result, when messages are sent daily, an attacker performing traffic analysis would be able to determine when all the members of a household are away or not using electricity for an entire day. We demonstrate this issue by recognizing when households from our dataset were on holiday. If messages are sent more often, more granular living patterns could likely be determined.
We propose a method of encoding the data that is nearly as effective as compression at reducing message size, but does not leak the information that compression leaks. By not requiring compression to achieve the best possible data savings, the risk of traffic analysis is eliminated.

Keywords: Privacy · Compression · Traffic analysis · Smart meter

1 Introduction

Privacy risks of smart metering have been analysed by looking at what information can be deduced from energy measurements on household granularity [18,7,6,16]. This shows that smart metering measurements are privacy sensitive.

The smart meter can send, among other things, a daily report of meter values to the system operator. For an individual meter this report in its plainest form can grow to several kilobytes. For several reasons, amongst which is simply the monetary cost of data, system operators want to limit the bandwidth used by this communication. The standards used for this communication are IEC 62056, more commonly referred to as DLMS/COSEM. They allow for encoding and compression to be applied to the meter readings, before encrypting them and sending them to a central system [8,9,10].

Because these messages are encrypted, someone who can eavesdrop on the communication does not have access to the actual meter readings. However, traffic analysis may still be possible. In traffic analysis, we analyse the metadata of network communication: who communicates, when, how much, to whom, without regard to the contents of the communication. Encryption does not necessarily reduce the risk of traffic analysis, especially if, as is the case for DLMS/COSEM, the length of the messages is still known to the outside observer. The latest version of DLMS/COSEM, not yet standardized by the IEC, defines a new encoding method in addition to the existing encoding and compression options. This paper explores how these options can influence the length of typical messages and what information this may leak to an attacker.

In addition, we propose a method of encoding the data that is nearly as effective as compression at reducing message size, but is not vulnerable to traffic analysis by itself. This allows for data savings without introducing the risk of traffic analysis.

Attacker model The question we are concerned with in this paper is whether an attacker observing DLMS/COSEM traffic can learn privacy-sensitive information solely from the length of the messages when the encoding and compression options in DLMS/COSEM are used. We assume a passive attacker capable of capturing all DLMS/COSEM traffic, but not injecting or manipulating messages.

In the Dutch smart metering infrastructure, measurements are currently taken every 15 minutes, and sent in daily batches after midnight. We perform our analysis on the messages communicating these daily batches to the grid operator. The only source of information for the attacker is the length of these messages. For this research, we do not consider other types of messages like reports on power quality, because the link between them and potential privacy impact is unclear.

In Section 2 we relate this paper to existing research into the privacy of smart metering. In Section 3 we explain the relevant parts of the DLMS/COSEM communication standards: the encodings and compression. We also introduce our proposed alternative encoding that should prevent the problems we identify. In Section 4 we explain the setup for our analysis, and in Section 5 we show the results and discuss our findings. Finally, we suggest some avenues for future work in Section 6, and we discuss our findings and give some recommendations in Section 7.

2 Background

The encryption used in DLMS/COSEM does not hide the plaintext message length from the attacker, it only adds a constant overhead to every encrypted message. This means that an attacker may be able to derive privacy-sensitive information by analysing the length of messages, partially circumventing the protection that the encryption is supposed to provide.

We cover some related research on privacy aspects of smart metering and the situation in the Netherlands in Section 2.1. In Section 2.2 we explain why the length of encrypted messages may leak information about the data they contain. In Section 2.3 we relate this to existing work that analyses correlations between power use and compression, and explain our contribution.

2.1 Smart Metering Privacy

Privacy risks of smart metering have been analysed by looking at what information can be deduced from energy measurements of an entire household. Molina-Markham et al. show that with one measurement every second, very detailed household living patterns can be deduced [18]. Greveler et al. show that from similar data they can recognize household appliances like refrigerators, kettles, and coffee machines. Worryingly, they can even distinguish different television broadcasts being watched [7,6]. Liao, Stankovic, and Stankovic also show that detecting refrigerators, boilers, kettles, toasters, etc. is possible, and by doing so, can distinguish household activities [16].

All these focus on what can be learned from frequent unencrypted readings, on the order of a measurement per second. Such frequent measurements are not (currently) transmitted by Dutch smart meters – they send measurements taken every 15 minutes [20]. However, that does not mean that there are no privacy concerns for the data that *is* sent by the smart meters. The law introducing the smart meter in the Netherlands was initially blocked by the First Chamber of Parliament, because it did not adequately take consumer privacy into account. An important issue was that it mandated 15-minute readings, and made the smart meter itself mandatory. The law was only passed after being rewritten to make the smart meter optional and the 15-minute readings opt-in [1]. The Distribution System Operators (DSOs) themselves also consider some of the metering data – in particular the measurements of energy use – privacy-sensitive [4].

2.2 Traffic Analysis: Length as a Side-Channel

The research mentioned in Section 2.1 [18,7,6,16] uses the actual energy use data from the meter, which can be hidden by encryption. It may be possible, however, to use message size as a side-channel to gain information about the encrypted data, especially if encrypted messages directly leak the size of their plaintexts. E.g. consider the case where we have two batches of an equal number of meter readings: one containing a batch of 8-bit integers, and the other a batch of 32-bit integers. Even if we encrypt these before sending them to a client, an attacker can still trivially distinguish the two, just by seeing that the *size* of one message is much larger than the other.

Even when the messages being encrypted are the same length, using compression to save bandwidth may introduce possibilities for traffic analysis. E.g.

now consider the case where we have two batches of meter readings, where one batch has measured “0” fifty times, and the other batch has fifty different measurements. These measurements take an equal amount of space in the message. If we only encrypt them and send them to a recipient, an attacker listening in would not be able to tell, based on size alone, which one we have sent. But if we compress them before encrypting, one message may compress down to say “50 times 0”, whereas the other needs the space for all its individual measurements. The attacker looking at message length can deduce that one of these messages has a lot of repeating values, whereas the other does not.

Already in 2002 the existence of such a compression-induced length side-channel in encrypted messages was highlighted by Kelsey [12]. Langley hypothesized in 2011 that compression before encryption could be used in an attack to retrieve information being transmitted over the then-newly-developed SPDY protocol, without actually breaking the underlying cryptographic protocols [13]. This was put in practice within a year by Rizzo and Duong in the CRIME attack against HTTPS, SPDY, and TLS [19]. In 2013 Prado, Harris, and Gluck followed this with the BREACH attack against the compression in HTTP [5].

It is important to note that these attacks do not “break” the cryptography as such. Instead, the traffic analysis reveals the contents of parts of the communicated data, even though the messages remain encrypted.

2.3 Traffic Analysis to Learn Power Consumption

Encryption in the DLMS/COSEM standard primarily uses AES in Galois Counter Mode, which adds a constant overhead to each encrypted message and does not hide the plaintext message length from the attacker [8]. So from Section 2.2 we can conclude that an attacker who sees the message length may be able to learn information about the power consumption of a household by linking message length to the power consumption.

In the case of smart metering, all the messages containing energy measurements could in principle be the same size: they contain the same number of measurements encoded the same way, as we explain in Section 3. However, if compression is used, the messages can have different lengths.

Fehér et al. [2] show for a limited dataset that compression of smart metering data can result in correlation between message length and power consumption. They do not assess the practical implications of this correlation: the fact that one exists does not necessarily mean it can be used by observers to infer interesting information in real-life scenarios.

Our contribution is that we have analysed the effects that encoding and compression have on the energy use data of real Dutch households. We investigate whether an attacker performing traffic analysis could actually find a link between message length and power consumption and derive privacy-sensitive information from it. We confirm there exists such a correlation on a much larger dataset and show concrete risks to privacy stemming from these correlations. Furthermore, we propose an encoding that approaches the effectiveness of compression, but does not exhibit the same correlation as compression.

3 Encoding and Compression Options in DLMS/COSEM

As mentioned in Section 1, system operators want to limit the bandwidth used by the communication needed for smart metering. In this paper we deal with two orthogonal concepts: encoding and compression. Both of these accomplish a reduction in size. Encoding defines how individual data elements are represented in a message. It can transform individual data elements in the message to achieve a more efficient representation of the same information, e.g. by using smaller data types or by converting absolute to relative measurements. Compression, on the other hand, applies a compression algorithm to an entire message without regard to the data contained within. Importantly, compression and encoding can be applied *together*, first transforming the data into a smaller encoded version, then applying compression to it. We therefore consistently distinguish between encoding and compression.

In Section 3.1 we briefly introduce the DLMS/COSEM standards. In Section 3.2 we introduce the different encoding mechanisms used in our analysis, and in Section 3.3 we explain the compression mechanism used.

3.1 DLMS/COSEM

DLMS/COSEM (Device Language Message Specification / Companion Specification for Energy Metering) is a set of IEC standards that define

1. a COSEM object model that gives structure to the available information in the form of COSEM objects [10,9], and
2. a DLMS/COSEM communication stack that defines the messages and underlying communication layers [8] used to communicate the objects.

As part of 2, elements are encoded using ASN.1, with a tag-length-value structure: every element is tagged with its type and, if the type does not have a predefined length, its length, followed by the encoded value of the element.

We can put all the 15-minute energy consumption measurements of an entire day in a batched measurement object. The structure of such an object and a possible corresponding encoding as a message is given in Table 1. Each measurement is combined with a status code and timestamp stating when the measurement happened in a `Struct`, and all 96 `Struct`s are wrapped in an `Array`. This example uses a straightforward encoding not designed to reduce the size, which we deduced from documentation and test cases available in DLMS/COSEM.

In this construction, every element is tagged with its type. The overhead of this can be significant: an object containing only 8-bit integers encoded this way would result in half of the message being spent on the type-tags of these integers.

Object Element	Value	Encoded value (hex)	Length (bytes)
Header		C4010000	4
Array (96 elements)		0160	2
Struct (3 elements)		0203	2
Date	2013-01-01 00:00:00	090C07DD0101 0500000000800000	14
Status	0	1100	2
Measurement	65530	060000FFFA	5
Struct (3)		0203	2
Date	2013-01-01 00:15:00	090C07DD0101 05000F0000800000	14
Status	0	1100	2
Measurement	65816	0600010118	5
⋮	⋮	⋮	⋮
Struct (3)		0203	2
Date	2013-01-01 18:30:00	090C07DD0101 05121E0000800000	14
Status	0	1100	2
Measurement	78286	06000131CE	5
⋮	⋮	⋮	⋮
Struct (3)		0203	2
Date	2013-01-01 23:45:00	090C07DD0101 05172D0000800000	14
Status	0	1100	2
Measurement	82362	06000141BA	5
Complete message		C401000001600203090C07DD01 010500000000800000110006000 00FFFA0203090C07DD01010500 0F000080000011000600010118 ⋮ 0203090C07DD010105121E0000 800000110006000131CE ⋮ 0203090C07DD010105172D0000 800000110006000141BA	2214

Table 1. Structure of a batched measurement object (first two columns, based on DLMS/COSEM test cases), the encoding in the corresponding message (third column), and length of each encoded field. Most measurements have been omitted for brevity. The tags and lengths of the elements are included in the encoded values, and therefore accounted for in their length. E.g. the encoding of the measurement itself as a 4-byte integer needs 5 bytes due to the tag.

3.2 Possible Encodings in DLMS/COSEM

A new iteration of the DLMS/COSEM IEC standards is currently in development, currently forecast to be published in June 2022.

In the DLMS/COSEM standards there are two encoding options with the explicit purpose to save bandwidth:

1. NULL Coding (already standardized in [9,10,8])
2. Delta Coding (proposed as part of the next iteration of the standards)

In this section we explain both of these encodings and how they apply to our problem.

NULL Coding In the current (2017) COSEM object model [9,10], a value may be replaced by a short NULL-value if it can be unambiguously derived from the previous instance of that object. For meter readings, this may happen when the meter reading is the same as the previous one. For timestamps, this may happen if an initial timestamp is transmitted and the periods between timestamps are known. We refer to this mechanism as NULL Coding.

It is important to note that NULL Coding can only work because the `Array` type is heterogeneous: looking at Table 1, not every `Struct` in the `Array` needs to have an identical layout, so the integer types they contain can change if the encoding allows for that. However, in DLMS/COSEM it is also possible to use a so-called `Compact-Array`. This is a homogeneous array type that specifies the type-tags of all its elements only once at the start, and requires every element to have an identical structure. This prevents use of NULL Coding for everything but timestamps in `Compact-Array`.

Delta Coding A second option, which we refer to as Delta Coding, is proposed for the next (2022) version of the DLMS/COSEM IEC standards. Similar to NULL Coding, Delta Coding seeks to save bandwidth by only transmitting changes to the previous value. E.g. on a system where a meter reading is a 4-byte integer, but power is consumed in amounts that fit in a single byte, Delta Coding would save three bytes per message, or 75%.

Because DLMS/COSEM uses tag-length-value encoding, new integer types (delta types) are introduced so that a distinction can be made between Delta Coded values and absolute measurements. The proposed delta types are signed and unsigned integers of 8, 16, and 32 bits.

The proposed standard does not (yet) prescribe an exact way in which these delta types should be used. To still be able to perform a usable analysis, we have considered what we believe are the most straightforward ways to use delta types for our batched measurement messages, and have come up with several different encodings which are all included in our analysis.

Since messages are considered independent, the first measurement in a message will still need to be an absolute measurement regardless of which encoding is used. All following measurements in that message can be encoded as delta types. The ways in which this could be accomplished that we analysed are:

1. **Minimum-Length Delta Coding**: Since the encoded `Array` allows mixing of types, the most obvious way is to simply use the smallest possible encoding for each individual measurement. Since most Dutch household connections provide 3 x 25A @ 230V connections, the theoretical maximum consumption in 15 minutes is 4313Wh, a value that easily fits in a 16-bit delta type. However, we expect that a lot of interval measurements in periods with low energy use will fit in 8-bit deltas. This encoding will therefore result in a *variable* message length saving between 2 and 3 bytes per measurement when compared to the encoding shown in Table 1.
2. **`N`-bit Delta Coding**: Another option is to pick the smallest delta type in which all measurements of an entire batch fit, and encode all measurements except the first one using that type. This results in larger messages than option 1, but the message length would not depend (as much) on consumption. We explore this option for 8-bit, 16-bit, and 32-bit delta types, and we refer to these as `N`-bit Delta Coding.

Both options 1 and 2 are possible using the normal `Array` type, and we believe they are both possible interpretations of the proposed addition of delta types. However, choosing option 2 with a 16-bit delta type could result in more savings than the minimum-length Delta Coding, if a homogeneous `Compact-Array` is used. This does not appear to have been considered, so we propose this as an additional option and include it in our analysis:

3. **16-bit Compact Delta Coding**: We propose the structure laid out in Table 2: a `Compact-Array` using 16-bit delta types to encode all except the first value. For this to work, the first measurement must be encoded outside of the array, because it must have a non-delta type to base the deltas on. The overhead needed for this is dwarfed by the savings that a `Compact-Array` provides over an `Array`. We refer to this option as 16-bit Compact Delta Coding.
   As we have already explained, the theoretical maximum consumption of a Dutch household in 15 minutes is much smaller than 65535Wh, so we do not really need to consider this case with a 32-bit delta type. Conversely, it would not work with an 8-bit delta type because some measurements do exceed 255Wh, which would re-introduce the need for a variable length.

3.3 Compression Used in DLMS/COSEM

Both in the current and proposed versions of DLMS/COSEM, the packet compression mode of ITU-T V.44 [11] may be applied to messages. This mode uses a data compression method in the Lempel-Ziv (LZ) family of compression algorithms: Lempel-Ziv-Jeff-Heath (LZJH) compression [11, Annex B.1].

4 Experimental Setup

We want to determine whether applying the encoding and compression options of DLMS/COSEM could enable traffic analysis, and whether that traffic anal-

Object Element	Value	Encoded value (hex)	Length (bytes)
Header		C4010000	4
Struct (2 elements)		0202	2
Struct (3 elements)		0203	2
Date	2013-01-01 00:00:00	090C07DD0101 0500000000800000	14
Status	0	1100	2
Measurement	65530	060000FFFA	5
Compact Array		13	1
Type tags			
Struct (3)		0203	2
Date Status 16-bit delta		09 11 20	1 1 1
Length (380 bytes)		8182017C	4
Entry			
Date	2013-01-01 00:15:00	00	1
Status	0	00	1
Measurement	65816	011E	2
⋮	⋮	⋮	⋮
Entry			
Date	2013-01-01 18:30:00	00	1
Status	0	00	1
Measurement	78286	0398	2
⋮	⋮	⋮	⋮
Entry			
Date	2013-01-01 23:45:00	00	1
Status	0	00	1
Measurement	82362	213	2
Complete message		C40100000202020309 0C07 DD01010500000000800000 1100060000FFFA13020309 11208182017C0000011E ⋮ 00000398 ⋮ 00000213	419

Table 2. Structure of the batched measurement object from Table 1, encoded according to our proposed 16-bit Compact Delta Coding. Omitting most type tags allows for massive data savings.

ysis could result in leaking private information. Our experiment consists of the following steps:

1. take publicly available real-world measurement data,
2. encode and compress them in the different possible combinations,
3. for each combination, attempt to find a relation between message length and energy consumption, and
4. use that relation to try to derive private information.

In Section 4.1 we introduce the dataset, and in Section 4.2 we explain how we generate messages from that dataset. In Section 5 we will cover steps 3 and 4.

4.1 The Zonnedael Dataset

For our analysis we use a publicly available dataset from the Dutch DSO Liander [15]. This dataset contains the real energy use data of 80 Dutch households, consensually collected for research purposes in 2013. Liander does not specify whether these households are within one neighbourhood. For our purposes this does not matter – all that matters is that these are real measurements from real households. Some filenames refer to this as the "Zonnedael" dataset, a fictitious name probably intended to better shield the data from deanonymization efforts. We therefore also refer to this data as the Zonnedael data.

The dataset contains relative energy measurements on a 15-minute interval, at Watt-hour resolution. We use this dataset mainly because it is readily available, contains real-world data, and has the measurement frequency we need for our analysis.

4.2 Converting Metering Data to DLMS/COSEM Messages

We take the relative energy measurements from the Zonnedael dataset introduced in 4.1. Using Python, we transform the dataset into absolute measurements like they would be taken by a smart meter. We then generate batch messages covering 24-hour periods starting at midnight, similar to how the Dutch infrastructure batches daily meter readings.

We construct messages:

1. without NULL Coding,
2. with NULL Coding applied only to dates, and
3. with NULL Coding applied to dates and Delta Coding applied to measurements.

For option 3, we implement the variants of Delta Coding mentioned in Section 3.2:

1. Minimum-Length Delta Coding,
2. fixed-length 32-bit, 16-bit, and 8-bit Delta Coding, and
3. our proposal of 16-bit Compact Delta Coding.

If a message cannot be encoded in a chosen encoding, that encoding is ignored for that message. This is e.g. the case when using 8-bit Delta Coding with measurements greater than 255Wh, or when particular measurements are missing from the dataset.

We compress each of these messages individually. Unfortunately, LZJH is a patented algorithm, with no open-source implementation available. Rather than attempt to write our own implementation, we decided to analyse the effects of compression using another member of the LZ family, Lempel-Ziv-Markov-chain (LZMA). Both algorithms are based on the concept of Lempel-Ziv complexity [14]. The basic operation of these algorithms is the same: repeated sequences of data in a stream are replaced by references to the earlier occurrences. Fehér et al. use the same rationale we do for their choice of using Lempel-Ziv-Welch as an approximation of the behaviour of LZJH in [2]. We therefore assume that our findings for LZMA will hold for LZJH as well, though it would be interesting to see our results reproduced by a meter manufacturer with access to an implementation of LZJH.

To actually perform the compression we use the routines available in Python's standard library [17].

We store the compressed and uncompressed version of each encoded message. We can then explore the relation between the length of the resulting messages and the energy measurements that they were generated from.

Note that we do *not* implement encryption. The AES-GCM encryption used in DLMS/COSEM only adds a constant amount of overhead to all messages, so the plaintext message length is known to the attacker. Since we do not look at the contents of the messages anyway, encrypting the messages would only add computational complexity without altering our findings. Even simulating encryption by adding a constant factor to the lengths is superfluous: our analysis would look the same regardless of whether we add a constant factor to all message lengths, because we are not interested in the absolute lengths, but in how they correlate with energy use.

5 Experimental Results

Our analysis using the real-world Zonnedael dataset introduced in Section 4.1 answers three questions:

1. Given the *un*compressed messages (for all encodings), can we find correlations between (daily) household energy use and message length?
2. Given the compressed messages (for all encodings), can we find correlations between (daily) household energy use and message length?
3. If this correlation exists, can we use message length to impact user privacy?

As a reminder, we can use all encodings with or without compression, which results in a total of 14 different options to analyse.

As we explain in Section 5.2, the answer to the first question is "no" — with one exception — whereas the answer to the second question is "yes". In Section 5.3 answer the third question by showing how we can use that correlation to impact user privacy, and we discuss the implications of our findings in Section 5.4. First, however, we look at how effective the encodings and compression actually are at saving data in Section 5.1.

5.1 Effectiveness of Encodings & Compression

The effectiveness of the encodings without compression applied is given in Table 3. NULL and Delta Codings are by themselves already very effective at reducing message length. NULL Coding shrinks the messages by a factor of 2.26, simply by eliminating the need to transmit every timestamp as a full 13-byte sequence. When Delta Coding is used, its effectiveness depends on the type of Delta Coding and the actual meter value, but it ranges from 2.26 up to 5.28.

An overview of compression effectiveness when applied to these encodings is given in Table 4. In addition to the tables, figures 1 and 2 give a visual indication of how effective the encodings and compression are at saving data.

Encoding	(Avg.) size	Reduction ratio
None	2214	—
NULL Coding	979	2.26
Minimum-Length Delta Coding	697	3.18
32-bit Delta Coding	979	2.26
16-bit Delta Coding	789	2.81
8-bit Delta Coding	694	3.19
16-bit Compact Delta Coding	419	5.28

Table 3. Message size & size reduction ratio for uncompressed messages. Sizes for the minimum-length Delta Coding are given as average. Size reduction ratio is given relative to the unencoded message.

Encoding (compressed)	Avg. size	Reduction ratio
None	348	6.36
NULL Coding	208	10.64
Minimum-Length Delta Coding	196	11.30
32-bit Delta Coding	206	10.75
16-bit Delta Coding	181	12.23
8-bit Delta Coding	180	12.30
16-bit Compact Delta Coding	179	12.37

Table 4. Message size & size reduction ratio for compressed messages. Sizes are given as average. Size reduction ratio is given relative to the uncompressed, unencoded message from Table 3.

The most important conclusions to draw from these results are:

- Uncompressed, our proposal of 16-bit Compact Delta Coding is overwhelmingly the best option, being much smaller than even the smallest messages of Minimum-length Delta Coding.
- Delta encoding using only 32-bit Deltas is equivalent to normal encoding with NULL Coding for dates, both achieving only a factor 2.26 improvement.

Thus, using only 32-bit deltas is not an improvement on already existing options.
– 8-bit Delta Coding is 3.19 times better than no encoding. However, 8-bit Delta Coding is not very useful because it turns out that on average only 18.5% of messages can be encoded using *only* 8-bit Deltas. The per-household median for this is even lower, at 12%.
– Minimum-length Delta Coding is as effective as 8-bit Delta Coding for those messages that can be expressed in only 8-bit Deltas, and then shows a slight increase in space required as energy use increases. This is visible as a *very slight* upward slope in the green plot on the left-hand side of Figure 1. However, this results in a correlation between energy used and message length, which is a problem, as we explain in Section 5.2.
– Compression is very effective in all cases. Something not apparent from Table 4, but which can be seen in Figure 1, is that compression on the Delta Codings has a large spread in the lower ranges of energy use. This spread narrows as the total use increases.

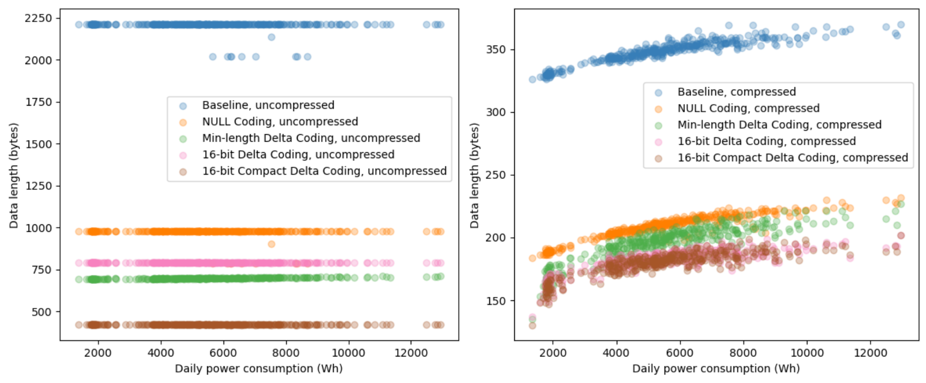

Fig. 1. Scatterplots of the relation between different encodings and data lengths of compressed & uncompressed messages of a single household. 8-bit Delta Coding removed because of its lack of usefulness and overlap with Minimum-length Delta Coding. Uncompressed encodings on the left, compressed encodings on the right.

5.2 Correlations of Encodings & Compression with Energy Use

We see strong correlations induced by compression on all encodings. Therefore, we first discuss the *uncompressed* versions of these encodings, and then discuss the effects of compression separately.

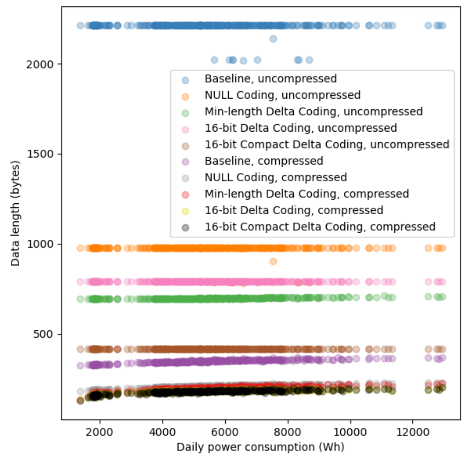

Fig. 2. Scatterplots from Figure 1 combined into one graph. Notice that uncompressed 16-bit Compact Delta Coding is very close to the compressed baseline.

Uncompressed messages We can find no correlation between the size of the uncompressed versions of messages encoded with *most* encodings and the energy use of a household – which follows from these being flat lines in Figure 1.

All the Delta Codings *except* the Minimum-Length Delta Coding are encodings where message size does not vary, so for them this is as expected.

Uncompressed versions of both NULL Coding and the baseline messages do show some variation, but this variation is not correlated with power use. This effect can be seen in the left graph of Figure 1, in the form of minor outliers below the majority of message lengths. These messages are all one of three sizes. The reason for this is simple: at the start of its life, a smart meter will be able to transmit the meter values in 8-bit integers, but this only holds until it passes 255 Watt-hours. Then, it will be able to use 16-bit integers until it passes 65 kWh. From that point, until it hits 4,294,967 kWh, the meter will be able to use 32-bit integers. This is expected to be sufficient well beyond its lifetime. We discuss the (negligible) privacy implications of this in Section 5.3.

However, we *do* find a strong correlation between power use and the length of uncompressed Minimum-Length Delta-Coded messages. We did expect to see some correlation here: 15-minute household consumption for the households in our dataset seems to mostly fit in 8-bit deltas, but as consumption increases more measurements in a message will need 16-bit deltas. This increases total message size by a single byte each time it happens, inducing *some* correlation between higher energy use and message length. However, we had not expected this correlation to be as strong as it is, around or above 0.8 for most households. Since the data-saving of Minimum-Length Delta Coding is inferior to our proposal of 16-bit Compact Delta Coding — which does not show any correlation — the safe option is to just use 16-bit Compact Delta Coding.

Compressed messages When compression is applied, results from our dataset show a strong correlation between the length of the messages and energy use, *regardless of the encoding used*. For the majority of customers, both the Pearson and Spearman correlations for all compressed messages with the power use are high, being at least 0.8 in most cases and 0.9 or higher in many. This is clearly visible in the upward slopes on the right-hand side of Figure 1. The actual correlation looks to be more logarithmic than linear in nature, but that is not a problem for determining that the relation exists in the first place.

One possible explanation for this correlation is that the lowest energy use of each single household happens when the residents are away from home for extended periods of time, possibly entire days, and the load of a household in this situation is likely to be repetitive, allowing for more compression. To clarify, when the residents are away from home, only the "base load" of a household is being measured. The base load consists mostly of duty-cycling equipment such as freezers, or always-on equipment such as clocks. The base load will therefore be both fairly low and repetitive. As mentioned in Section 4.2, the LZMA compression algorithm we use is based on the concept of Lempel-Ziv complexity. Lempel-Ziv complexity is a measurement of how "repetitive" a sequence is, and the lower the Lempel-Ziv complexity of a sequence, the better it is compressed by an LZ algorithm. Because the lowest energy use of a household is that where only the base load is present, it makes sense that the best results of compression correlate with the lowest energy use, with the correlation being caused by the *repetitive nature* of the energy use. This explains why the correlation does not hold as well across households: the actual consumption of the most repetitive load may differ significantly from one household to the next.

This is only conjecture, however, which can be subject of future research.

5.3 Deriving Private Information from the Correlations

Using the insights from sections 5.1 and 5.2, we can now show the we can derive privacy-sensitive information using the link between power consumption and message length. We first discuss the issue where a *new* meter leaks that fact. Then we show that we can determine when a household went on holiday.

New energy meter As mentioned in Section 5.2, uncompressed versions of both NULL Coding and the baseline messages do show some variation, because the meter starts counting from 0 and the initial messages can therefore use smaller integers to encode the measurements. However, the yearly use of an average Dutch household is between 1500 and 5000kWh [21], depending on household composition and building type, so after only five days most households will already have passed the point where 32-bit integers are being used. The length of uncompressed messages therefore *does* leak that a new meter is installed, but the privacy impact of this data is questionable and the leak is only present for a few days at most. After this, the length is stable, and no further information can be recovered from these two *un*compressed encodings.

Determining holidays and other absences We now show that the compressed versions of all encodings definitely leak privacy-sensitive data in a real-world setting. The graphs in Figure 3 show the power use over the course of the entire year for two different customers. They overlay the message lengths of compressed versions of a few different encodings, including our proposed 16-bit Compact Delta Coding.

We can assume that the lowest energy use indicates that the residents are absent from the house, and we know that message length strongly correlates with energy use. This does not allow us to make statements about these customers on an hour-to-hour basis, but it does allow us to recognize longer periods of absence because they break the somewhat irregular pattern of normal life.

As seen in Figure 3, customer 17 has clearly gone on a longer vacation twice: once in June, and once in August. We also think there may have been a short period of absence in October, and we believe they went away for Christmas. On the other hand, customer 46 probably went on a short holiday in February, a long summer holiday in June and July, and a third holiday in October, but celebrated Christmas at home.

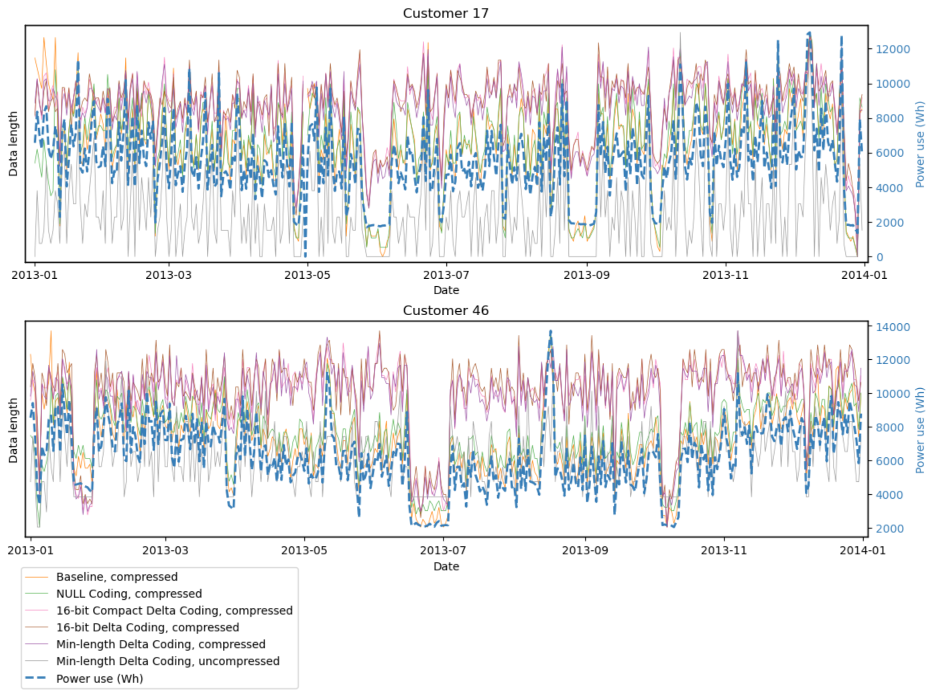

Fig. 3. The year in message lengths: household energy consumption of customer 17 (top) and 46 (bottom), plotted as actual power use (dashed blue) and the *normalized* message lengths (solid lines). The different message length plots in each graph are all the different encodings from which we can derive absences, all based on the same data for each customer. All except one are compressed: for Minimum-length Delta Coding the uncompressed version is also included. In all plots, it is very easy to recognize the valleys that indicate prolonged absence. The same plots for the other uncompressed encodings would just be horizontal lines across the year, providing no information, and are therefore omitted. The actual power use is only shown for validation; the effect is so striking it is clear we can derive the holiday periods from message length alone.

5.4 Discussion

In Section 5.3 we showed that we can derive private information from the correlations we found. In this section we discuss the implications of these findings. We speculate on other patterns that could be uncovered using this kind of analysis, paying particular attention to the influence we expect frequency of transmission and measurement to have on the capabilities of an attacker. We also reflect on how big of a privacy risk our findings actually are.

Frequencies of transmissions and measurements For the results we presented in Section 5.3 we specifically looked for absences on several consecutive days. However, this kind of analysis can also find patterns of single-day absences, e.g. somebody spending every Saturday away from home. In addition, if meters send smaller batches more frequently, we may be able to start distinguishing between work days and other days. This hypothesis is hard to verify using the Zonnedael dataset, because it does not include any information about what these patterns *actually* are for the households. However, our reasoning is fairly simple to explain. There are two key factors that play a role in this traffic analysis:

– Frequency of transmission
– Frequency of measurement

Changing the frequency of transmission has two effects – one that increases, and one that decreases our abilities:

– More frequent transmission leads to shorter time periods that information pertains to, which should make more detailed patterns emerge. E.g. if a batch of measurements is sent every 6 hours, we may be able to recognize where in the day someone wakes up, whether they went to work, etc.
– More frequent transmission leads to fewer measurements per batch, lowering the correlation because compression has less of an impact. Initial results show that as we approach just a few hours in a period, so on the order of ten measurements in a message, correlation falls sharply.

The second point can then be counteracted by having more frequent measurements. As we approach the order of a message per minute, with measurement frequency of one second, we might in fact be able to approach the capabilities of the research mentioned in Section 2.1, and provide a very detailed view of household activity [18,7,6,16].

But there may also be a negative effect to increasing the measurement frequency. Since the individual measurements are hidden, we have no way of determining *why* a message has a certain length. Since the contents of the measurements actually influence the way compression behaves, it's likely there is a limit to how many measurements can be in a single message before adding more measurements make the analysis *less*, rather than more accurate. In addition, the number of different values for individual measurements may also start playing an important role. 15-minute measurements on a Watt-hour resolution can be anything from 0 to a few thousand. When measured every second, however, they can only be between 0 and 5 for an average Dutch domestic connection.

With the Zonnedael dataset, we cannot really explore the influence of changing the frequency of measurements, because the measurements are fixed on a 15-minute interval. However, in future work we may explore what patterns we are able to deduce from more frequent batches.

Real-world situation Although not really a question for our research, we should discuss the relationship between the *real world* and the kind of analysis that we show in this paper. The traffic analysis assumes that the attacker can eavesdrop on the communication of the meter. If the attacker needs to be physically close to the meter for this, then we should consider that the attacker can also simply observe the house to derive the same information we have shown to be derivable from energy use, and see that someone is away from home. But if the attacker can monitor all the traffic for an entire neighbourhood, or even city, or more, by e.g. examining GPRS traffic, the value of traffic analysis becomes apparent. This is clearly something to take into account in the smart metering infrastructure, even though there are a lot of other pressing privacy issues in this domain.

We do note that we have been assured that the potential problems we identified are not present in the existing Dutch DLMS/COSEM infrastructure. We have also made a lot of assumptions about the format of messages and the desired encodings, based on our interpretation of what the standards *allow*, not on what is actually used in practice. The industry should test whether these assumptions hold in existing DLMS/COSEM implementations.

6 Future Work

We have performed our analysis on the daily batches of 15-minute interval measurements. We are aware of DSOs that are considering using shorter intervals, and reading them live. These scenarios should be explored in future work, as discussed in Section 5.4. The examined encodings and compression might end up influencing these message lengths in a totally different way. A real-world dataset with this granularity would be useful to perform this research, but we are currently unaware of the existence of such a dataset.

We have focused on the actual energy use by a household. The Dutch smart metering infrastructure also communicates other information, such as power quality. This information is treated as privacy-sensitive by Dutch DSOs [4], but the actual relation between power quality measurements and privacy remains largely unexplored. Both this relation and the subsequent impact of potential traffic analysis could be an avenue of future work.

We have suggested an alternative encoding scheme in Section 3.2 and Table 2 that already achieves very good data saving without being vulnerable to the kind of analysis we have done. Whether this solution is truly suitable for

DLMS/COSEM is an open question, and should be answered by the DLMS User Association.

If compression is still deemed necessary, a simple option is to determine an acceptable minimum length for energy use messages, and to pad any compressed messages to that length. This way they also become indistinguishable to an observer, and a good amount of compression can likely still be achieved without sacrificing privacy. The actual implementation and effectiveness of such a padding scheme should be considered by the DLMS User Association, or can be subject of future work.

The work presented by Fehér et al. proposes using the Generalized Deduplication [22] compression scheme, which should lead to lower correlations [2,3]. However, they do not show a complete absence of correlation in GD-compressed messages. So whether this scheme has the desired effect of making traffic analysis useless is an open question. It would be interesting to see if we can reproduce our results using the same dataset and this compression scheme.

7 Conclusions

Several options in the DLMS/COSEM specifications for communicating energy use measured by smart meters can result in variable-length messages and thereby may make traffic analysis possible. Since the AES-GCM encryption used in DLMS/COSEM does not hide the length of messages from an attacker, it has no effect on the possibility of traffic analysis. The options that result in variable length messages are:

1. NULL Coding, where a meter reading may be replaced by a shorter NULL value if it is identical to the previous reading;
2. Minimum-length Delta Coding, where a reading may be encoded in the *smallest* type in which it fits; and
3. compression.

An implementation may use both such an encoding and compression at the same time.

We have found that — in a real-world dataset, using our interpretation of possible DLMS/COSEM encodings — compressing batched energy measurement messages results in a strong correlation between message size and the daily energy use of households in all encodings that we have analysed. Using NULL Coding *without* compression does *not* show this correlation *in our dataset*, because very few measurements are ever identical to the previous one. But this does not rule out such a correlation existing in different datasets. Using Minimum-length Delta Coding without compression results in the same correlation as compression.

An attacker performing traffic analysis could therefore determine when all the members of a household are away. We have shown in Section 5.3 that we can actually use these correlations to identify periods in which households went on vacation, or whether they spent Christmas away from home. We conjecture that if these measurements were sent more often in smaller batches, e.g. four times per day, traffic analysis could reveal more detailed information and distinguish when people wake up, go to work, etc.

Whether this is actually an (un)acceptable privacy risk is up for debate. Our analysis assumed that meter readings are sent in a big batch, once per day, as is current practice in the Netherlands. However, neither compression nor variable-length encodings are currently in use in the Dutch metering infrastructure. So for now, the risk seems to be purely hypothetical. Also, how easy it is to eavesdrop on communication to then do traffic analysis will depend on the communication medium used and was outside the scope of our research.

Looking towards the future, the risk can easily be eliminated by ensuring there is no variation in the message length, sacrificing *some* data savings to eliminate this risk to user privacy. There are many ways to achieve this. We propose a construction that uses 16-bit Compact Delta Coding *without compression*, described in Section 3.2 and Table 2. This construction results in shorter uncompressed messages than even Minimum-length Delta Coding, and approaches the effectiveness of compression. However, because it does not result in messages that vary in length, it avoids the risk of traffic analysis.

References

1. Cuijpers, C., Koops, B-J.: Smart metering and privacy in Europe: Lessons from the Dutch case. In: European Data Protection: Coming of Age, pp. 269–293. Springer (2013). https://doi.org/10.1007/978-94-007-5170-5_12
2. Fehér, M., Yazdani, N., Aranha, D.F., Lucani Rötter, D.E., Hansen, M.T., Vester, F.E.: Side channel security of smart meter data compression techniques. In: IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm) (Aug 2020). https://doi.org/10.1109/SmartGridComm47815.2020.9302931
3. Fehér, M., Yazdani, N., Hansen, M.T., Vester, F.E., Lucani Rötter, D.E.: Smart meter data compression using generalized deduplication. In: IEEE Global Communications Conference (GLOBECOM) (Dec 2020). https://doi.org/10.1109/GLOBECOM42002.2020.9322393
4. Gedragscode slimme meters voor netbeheerders. Netbeheer NL (2017)
5. Gluck, Y., Harris, N., Prado, A.: BREACH: reviving the CRIME attack (Jul 2013), http://breachattack.com/
6. Greveler, U., Glösekötterz, P., Justus, B., Löhr, D.: Multimedia content identification through smart meter power usage profiles. In: Information and Knowledge Engineering. pp. 383–390. WorldComp (2012), https://worldcomp-proceedings.com/proc/p2012/IKE7720.pdf
7. Greveler, U., Justus, B., Löhr, D.: Identifikation von videoinhalten über granulare stromverbrauchsdaten. In: Sicherheit, Schutz und Zuverlässigkeit. pp. 35–45 (2012), http://subs.emis.de/LNI/Proceedings/Proceedings195/article6606.html
8. Electricity metering data exchange – the DLMS/COSEM suite – part 5-3: DLMS/COSEM application layer. IEC standard 62056-5-3:2017, International Electrotechnical Commission (Aug 2017), https://webstore.iec.ch/publication/27065

9. Electricity metering data exchange – the DLMS/COSEM suite – part 6-1: Object Identification System (OBIS). IEC standard 62056-6-1:2017, International Electrotechnical Commission (Sep 2017), https://webstore.iec.ch/publication/32782
10. Electricity metering data exchange – the DLMS/COSEM suite – part 6-2: COSEM interface classes. IEC standard 62056-6-2:2017, International Electrotechnical Commission (Sep 2017), https://webstore.iec.ch/publication/34317
11. International Telecommunication Union: Recommendation V.44 – Series V: Data Communication over the Telephone Network – Error Control – Data Compression Procedures (2000), https://www.itu.int/rec/T-REC-V.44-200011-I/en
12. Kelsey, J.: Compression and information leakage of plaintext. In: Fast Software Encryption. pp. 263–276. Springer (2002). https://doi.org/10.1007/3-540-45661-9_21
13. Langley, A.: Compression contexts and privacy considerations. spdy-dev mailing list (Aug 2011), https://groups.google.com/g/spdy-dev/c/B_ulCnBjSug/m/rcU-SIFtTKoJ
14. Lempel, A., Ziv, J.: On the complexity of finite sequences. IEEE Transactions on Information Theory **22**(1), 75–81 (1976). https://doi.org/10.1109/TIT.1976.1055501
15. Liander N.V.: Datasets slimme meter, "Zonnedael". https://www.liander.nl/partners/datadiensten/open-data/data, https://www.liander.nl/sites/default/files/Over-Liander-slimme-meter-dataset-2013-levering.zip
16. Liao, J., Stankovic, L., Stankovic, V.: Detecting household activity patterns from smart meter data. In: 2014 International Conference on Intelligent Environments. pp. 71–78 (2014). https://doi.org/10.1109/IE.2014.18
17. lzma – compression using the LZMA algorithm, https://docs.python.org/3.9/library/lzma.html
18. Molina-Markham, A., Shenoy, P., Fu, K., Cecchet, E., Irwin, D.: Private memoirs of a smart meter. In: Workshop on Embedded Sensing Systems for Energy-Efficiency in Building. pp. 61–66. ACM (2010). https://doi.org/10.1145/1878431.1878446
19. Rizzo, J., Duong, T.: The CRIME attack. Ekoparty Security Conference (Sep 2012), https://docs.google.com/presentation/d/11eBmGiHbYcHR9gL5nDyZChu_-lCa2GizeuOfaLU2HOU/
20. Van Aubel, P., Poll, E.: Smart metering in the Netherlands: What, how, and why. International Journal of Electrical Power & Energy Systems **109**, 719–725 (2019). https://doi.org/10.1016/j.ijepes.2019.01.001
21. Vereniging Nederlandse EnergieDataUitwisseling: Profielen elektriciteit (2016 – 2020). https://www.nedu.nl/documenten/verbruiksprofielen/, https://www.nedu.nl/documenten/verbruiksprofielen/
22. Vestergaard, R., Zhang, Q., Lucani Rötter, D.E.: Lossless compression of time series data with generalized deduplication. In: IEEE Global Communications Conference (GLOBECOM) (Dec 2019). https://doi.org/10.1109/GLOBECOM38437.2019.9013957