Title: **Secure Store and Forward Proxy for Dynamic IoT Applications over M2M Networks**

Authors: Daniel Díaz-Sánchez, *Senior Member*, IEEE
Telematic Eng. Department, Carlos III Univ., 28911, Leganés, Madrid, SPAIN
(e-mail: dds@it.uc3m.es)

R. Simon Sherratt, *Fellow, IEEE*
Department of Biomedical Engineering, the University of Reading, RG6 6AY, UK
(e-mail: sherratt@ieee.org)

Florina Almenarez, *Member*, IEEE
Telematic Eng. Department, Carlos III Univ., 28911, Leganés, Madrid, SPAIN
(e-mail: florina@it.uc3m.es)

Patricia Arias, *Member*, IEEE
Telematic Eng. Department, Carlos III Univ., 28911, Leganés, Madrid, SPAIN
(e-mail: ariasp@it.uc3m.es)

Andrés Marín, *Member*, IEEE
Telematic Eng. Department, Carlos III Univ., 28911, Leganés, Madrid, SPAIN
(e-mail: amarin@it.uc3m.es)

This work has been partially supported by INRISCO TEC2014-54335-C4-2-R and Jose Castillejo Mobility Grant CAS14/00364.

Publication: **IEEE Transactions on Consumer Electronics**
Volume: 62
Issue: 4
pp.: 389 - 397
Date: November 2016
DOI: 10.1109/TCE.2016.7838091
Date of Publication: 02 February 2017

Abstract

Internet of Things (IoT) applications are expected to generate a huge unforeseen amount of traffic flowing from Consumer Electronics devices to the network. In order to overcome existing interoperability problems, several standardization bodies have joined to bring a new generation of Machine to Machine (M2M) networks as a result of the evolution of wireless sensor/actor networks and mobile cellular networks to converged networks. M2M is expected to enable IoT paradigms and related concepts into a reality at a reasonable cost. As part of the convergence, several technologies preventing new IoT services to interfere with existing Internet services are flourishing. Responsive, message-driven, resilient and elastic architectures are becoming essential parts of the system. These architectures will control the entire data flow for an IoT system requiring sometimes to store, shape and forward data among nodes of a M2M network to improve network performance. However, IoT generated data have an important personal component since it is generated in personal devices or are the result of the observation of the physical world, so rises significant security concerns. This article proposes a novel opportunistic flexible secure store and forward proxy for M2M networks and its mapping to asynchronous protocols that guarantees data confidentiality.

Index Terms

Machine to Machine, Internet of Things, IoT, privacy, encryption.

I. INTRODUCTION

The Internet of Things (IoT) [1] encompasses the related concepts of Machine to Machine (M2M) [2], Smart Cities [3] and Crowd Sensing [4]. These concepts and IoT are sometimes interchangeably used [5]. Smart Cities, despite its unclear definition [6], strives for transforming life and working environments within a region by embedding a wide range of electronic, digital and data technologies in government systems [7]. Crowd sensing pursues a fusion of human and machine intelligence [8]. It is a distributed problem-solving model where a crowd is engaged in solving a complex problem through an open call [4] by means of their personal devices.

IoT considers the overall dataflow for a number of different concepts, as Smart Cities or Crowd Sensing, involving how the information is fetched from connected 'things,' then combined and presented to help efficiently use physical infrastructures [9], achieve local governance with improved e-participation [10], learn, adapt and innovate with environments effectively [11]. Beyond enabling the aforementioned concepts, the rise of IoT is proportional to the need to enable technologies at the right cost [12] and that affects IoT from the design of the connected thing to the final data processing. When it comes to data processing, elastic computing frameworks in the cloud [13] assist on the process of transforming the unprecedented amounts of data [14], [15] that will be generated in these scenarios into usable metrics, statistics and predictions. Moreover, due to the variety of different applications, flexible network architectures are also needed to transport data from devices to the backend. These infrastructures should, at least, enable interoperability, provide resilience and preserve confidentiality and privacy. For that reason, several standardization bodies have joined to develop M2M architectures [16]. M2M has its origin in Supervisory Control And Data Acquisition (SCADA) systems, but also encompasses the evolution of Mobile Cellular Networks and heterogeneous Wireless Sensor Networks (WSN) and Actor Networks to converged networks. Unlike traditional WSN, that usually serve a single application/organization, M2M networks are designed to improve interoperability among them, pursuing M2M networks to be shared among different applications and organizations. Due to that, M2M networks are becoming important connections into bigger systems as IoT. They consist of a large number of heterogeneous devices with different capabilities and functionalities that can inter-operate among them and with different back-ends.

The wide range of IoT applications using M2M networks will generate a tremendous amount of traffic that will affect network performance. Moreover, unlike the traffic generated by traditional Internet services, which flows from the network to user devices, M2M traffic will flow from devices to the network. Thus, M2M networks should accommodate, among others, urgent, burst and regular traffic patterns without interfering with existing Internet services. To accomplish that, M2M networks provide intermediate storage, proxies and other network elements that allow traffic shaping and store and forward services preventing the network from collapsing.

Managing networks in real time is now a reality with technologies including Software Defined Networking (SDN) [17] but may require redirecting data through untrusted networks or storing data opportunistically in intermediate untrusted proxies. Moreover, it may also require redirecting or forking traffic to different sinks in a dynamic way to overcome network congestion and cope with applications demands. However, the data handled by M2M networks in behalf of IoT applications contains huge amounts of personal data, either generated by personal devices (as it may happen with connected personal devices or crowd sensing) but also those generated as a result of observing the physical world (as it may happen in Smart City applications), rising reasonable concerns about confidentiality and privacy if data traverses untrusted networks.

This article considers different IoT/M2M applications and the supported traffic patterns to propose an novel opportunistic and flexible secure store and forward proxy for M2M systems. Traditional public key encryption allows a user Alice to encrypt a message *m* to user Bob using Bob's public key so only Bob can decrypt because only Bob has his private key. The algorithm used by this proposed proxy allows a message *m* to be encrypted by Alice (using her private key) and delivered to a proxy without knowing who will be the recipient. If later on, Bob is designated as the recipient, Alice can provide a re-encryption key to the proxy that allows it to transform the message encrypted by Alice into a message encrypted to Bob without actually accessing the plaintext. In such a way, a proxy can aggregate and store encrypted data from Alice until instructed to deliver that data to a recipient transforming the data appropriately so the recipient can decrypt it. The article also explores its viability with common Consumer Electronics (CE) devices as those acting as end devices or middle backhaul network elements and the mapping of the protocol to asynchronous protocols including CoAP [18].

This article is based on a previous work [19]. The remaining of the article is structured as follows. Sections II introduces technologies related to this research and also explains other security approaches. The design of the secure proxy is elaborated in section III. Section IV goes into detail about the architecture and shows the proxy operation for a concrete case. Section V provides implementation details and some metrics. Finally conclusions are presented in section VI.

II. BACKGROUND AND RELATED WORK

A. *IoT/M2M Architecture*

A plausible IoT system using M2M networks, according to current standardization efforts [2], is depicted in Fig. 1. It shows three different domains, the M2M device domain, the network domain and the application domain.

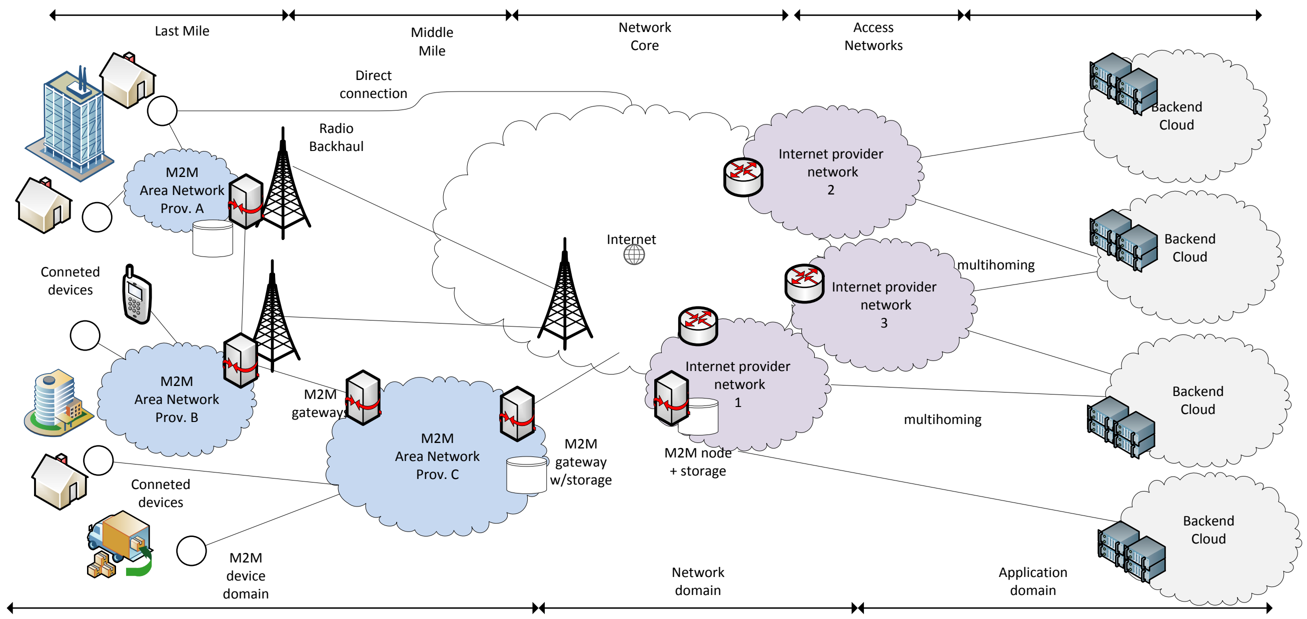

Fig. 1. IoT connected devices using M2M networks to reach Internet. Backends are connected through several providers.

Depending on the application, a M2M area network may contain a huge number of devices including sensors, actuators, mobile devices, or industrial control devices that provide data as environmental, utility, fleet, energy control, etc. It also contains smart devices as gateways that collect, aggregate, store or filter data packets that are eventually transmitted through the network domain to the backend servers in the application domain. Devices using a given M2M network can communicate with others using gateways and gateways can cooperate with each other. Gateways access the network domain using long-range terrestrial links or radio backhauls.

Finally, backend servers in the application domain process the data from the connected devices for a given purpose.

B. *M2M Communication Technology*

Devices in the ‘last mile’ within the device domain are frequently limited, either in memory, processing power, battery or both. Devices may need to rely on gateways to reach the application domain, may need to suspend to save battery or cooperate with near devices to accomplish their tasks. Thus, network technologies employed in this domain should also consider these constraints. Bluetooth was the first affordable Personal Area Network (PAN) technology available and is nowadays supported by the vast majority of commercial devices. However, it was designed for a limited number of use cases, not to address current IoT/M2M needs due to its power consumption.

Zigbee [20], since its conception, was designed to support larger low-rate Personal Area Networks (LR-WPAN) with lower data rates that required much less power to operate, and due to that, covered several use cases not considered by Bluetooth. That circumstance has partially changed with the development of Bluetooth Low Energy (BTLE) that, keeping BT data rate, offers several modes that suit some of the IoT/M2M use cases. Zigbee is the foundation for recent designs [21] as 802.15.4e that improves the support for some industrial applications; 802.15.4f supporting bidirectional RFID; or 802.14.4g specially designed for Smart Utility Networks [22]. BTLE is an interesting alternative in heavy industrial environments since its modulation is more robust to RF jamming than Zigbee's.

Other transmission technology as Z-Wave (short-range, low-rate with similar Zigbee consumption), DASH7 or Wireless M-Bus, have been proposed to deal with similar environments but lacks of a significant community adoption [23], [24]. WiFi, supporting short range communications, provides higher data rates making it useful in some cases to link areas within the last mile or middle mile (limited to special cases). WiMAX (IEEE 802.16) due to its long-range is especially interesting as a middle-mile backhaul to the network domain.

C. M2M Communication Protocols

IoT and M2M have inherited the use of web services or web APIs on top of the Representational State Transfer (REST) [25] architecture. RESTful environments have traditionally relied on HTTP, and thus TCP, to provide a scalable and reliable transport for the creation of resources, the update and retrieval of their representation. Together with the use of a comprehensive naming space based on URLs, the RESTful architecture is present in the majority of Internet services.

However, HTTP and TCP are not adequate for constrained devices. TCP is a loss-less end-to-end connection oriented transport protocol that provides reliability, congestion control and flow control. Despite TCP simplifying the application when it comes to data transmission by placing intelligence in protocol, it requires to keep state at both endpoints during the entire life of an interaction, thus compromising valuable resources, particularly battery power. Moreover, HTTP is a text based request response protocol that has improved system interoperability but presents a significant overhead and encoding/parsing complexity; its point to point communication model is not suitable for some interesting IoT/M2M traffic patterns as notifications, while the adoption of HTTP Multicast has been very scarce.

Absorbing huge amounts of traffic from devices scattered among a big area is a challenge involving every part of the system architecture. A coherent approach to the problem requires responsive, message-driven, resilient and elastic architectures. For that reason, the work on Constrained RESTful Environments (CoRE) concentrates on providing an adequate RESTful architecture for the most constrained devices and networks proposing 6LoWPAN [26] and Constrained Application Protocol (CoAP) [27], [28]. 6LoWPAN provides open systems based interoperability among low-power devices using constrained radio transmission technology, especially 802.15.4. In such a way, devices can access IP networks and communicate directly with existing IP devices using standard IP routing techniques at a reasonable cost. The major contribution of 6LoWPAN is enabling IPv6 simplifying the requirements thus improving adoption and enabling a huge address space.

CoAP defines a generic web protocol for those constrained devices and networks fulfilling M2M requirements, unicast and multicast delivery over UDP, asynchronous and message driven exchanges with low overhead together with proxy and caching capabilities (that is consistent with M2M networks). CoAP defines a messaging model over UDP containing a short header, binary options and an optional payload. Every message contains a 16bit long Message ID field that can be used for reliability. If a message should be confirmed it should contain a confirmable (CON) flag and the sender will wait for an ACK with the same Message ID. If a timeout expires without the corresponding ACK, the sender will retransmit the message following. Messages requiring no reliability will be sent without the CON header. Request and responses in CoAP are carried over this CoAP reliable messaging layer. Requests and responses carry a Method Code and a Response Code respectively, together with optional information as URI or media type. Methods and Response codes are consistent with RESTful systems. A Token is used to match responses to requests independently from the underlying messages; the token size can vary from 0 to 8 bytes.

D. Traffic Characterization

As discussed by Verma *et al.* [2], the existence of differentiated traffic patterns in M2M gives network operators an extra degree of freedom to prioritize M2M operation reducing congestion, latency and packet loss in the network core that is shared with existing internet services.

A thorough discussion on the different M2M applications and their traffic characterization can be found in the literature [2], [29]-[31]. For the purpose of the research described in this article, two general application groups are differentiated. Rigid applications are those requiring immediate and/or constrained delivery. Emergency applications, critical infrastructure monitoring or alerts triggered by monitoring systems, are some examples of applications that need immediate delivery. Others as surveillance, live video, or media distribution, are examples of applications that need bounded network delay or rate to work.

Adaptive applications are those tolerant to either delay or rate. Smart grid, environmental monitoring, vending and any other M2M application data, excluding alarms and urgent messages, are worth to be processed, but not immediately. For instance, environmental monitoring may generate a large amount of traffic that, once processed, helps to evaluate interesting variables including air quality. However, whenever the measured values delivered are below a given alarm threshold, that data, that does not need to be processed immediately, may be stored conveniently at intermediate network nodes until network conditions improve in case of congestion. In such a way, data loss and network congestion can be minimized.

E. Related Work

Many aspects of security in M2M networks have been already covered by underlying protocols. Datagram Transport Layer Security (DTLS) [32] describes and adaptation of Transport Layer Security that works over UDP, and thus, can be used as a transport for CoAP [33]. Alternatively, 6LoWPAN has been extended to support IPSEC [34]. These approaches tackle end to end security but require pre-existing trust relationships among nodes. Others deal with message authentication [35] or explore novel approaches [36] including identity [37] and privacy.

M2M has its origin in Industrial Control System (ICS) as SCADA, but incorporates protocols that have been proven resistant to many different attacks on the Internet. However, since the worm *Stuxnet* revealed that Industrial Control System (ICS) can be compromised, several efforts has been proposed to amend existing ICS systems that are worth mentioning. ModBus [38] is an application layer protocol used to communicate supervisory data to SCADA systems that lacks authentication and is vulnerable to various security attacks. Profibus [39] is a master/slave protocol that supports isolated communication based on token possession but also lacks of authentication allowing spoofed nodes to impersonate the master. Other solutions [40] propose mechanisms to verify node integrity in a distributed Industrial Control System (ICS) requiring a previous trust relation. Unlike the aforementioned ICS security solutions, the presented solution from this work does not only target industrial control system but also general IoT/M2M applications. It is not constrained by controller/node architecture, and it allows for unidirectional asynchronous communication between devices and backends so devices can send data asynchronously over virtual isolated channels. Moreover, unlike [40], IPSEC or TLS/DTLS, the presented solution from this work does not require pre-existing relations among entities since recipients can be determined after data is transmitted and proxies can be untrusted. In such a way this work provides a mechanism for secure opportunistic network storage based on proxy re-encryption that helps managing M2M traffic by relying in any available network storage even if it is located within an untrusted domain. End to end security as provided by DTLS or IPSEC otherwise, requires data sinks to be trusted so their certificates or raw public keys are known to the sender (or can be verified) in advance.

The proxy re-encryption algorithms this work relies on the notion of 'atomic proxy cryptography,' first introduced by Mambo and Okamotom [41], using a semi-trusted proxy that transforms a cipher text for Alice into a cipher text for Bob without actually accessing the plaintext. Popular well-known approaches to proxy re-encryption are BBS (Blaze, Bleumer & Strauss) [42] and AFGH (Ateniese, Fu, Green & Hohenberger) [43], [44]. BBS is based on ElGamal. In BBS, the proxy, knowing the key for Alice to Bob direction, can derive the key for Bob to Alice direction as the multiplicative inverse. This is not a feature but a problem since the proxy, which can be untrusted, does not need additional information to derive the inverse. Moreover, to accomplish bidirectional communication, two keys (direct and inverse) should be kept since deriving the inverse every time is computationally expensive. AFGH is based on Bilinear maps [45] and has the property a proxy knowing the transformation key for one direction cannot derive the opposite direction key from it. Moreover, a message that has been encrypted for proxy re-encryption with AFGH can be recovered by the origin before and after proxy re-encryption permitting the source to recover, forward or re-encrypt a message for another destination if network conditions change again as it may happen in dynamic scenarios.

III. SECURE PROXY DESIGN

The purpose of proxy re-encryption algorithms is to provide confidentiality among data sources and sinks even if data should traverse, be stored or managed by, an untrusted third party.

It should be noted that an untrusted entity is considered an entity owner of the data that has no preexisting trust relation with. This can happen because the entity is actually unknown to the owner, is a competitor etc., so should not have access to unencrypted data. Thus, it does not mean the entity is rogue or malicious. In the context of M2M, those networks or entities, despite unknown, thus untrusted, can be identified using the mechanisms M2M provides for that purpose. Malicious entities or networks are otherwise considered distrusted and should not be used since reliability cannot be guaranteed.

Proxy re-encryption algorithms are very useful over existing end to end security mechanisms when some of these circumstances occur: 1) potential data sinks are not known in advance either because network errors force to send data to alternative data sinks, or new data sinks willing to collect data from connected devices are added dynamically; 2) there is no control or trust relation with all the entities the data will flow through; 3) intermediate unknown or untrusted entities as proxies or storage, are needed to avoid data loss or circumvent congested networks.

A. Algorithm Selection and Key Generation

As discussed in the previous section, the biggest traffic component in M2M flows from connected devices to the network and not otherwise. A big portion of traffic is expected to follow a fire-and-forget pattern avoiding limited devices to wait for confirmation which reduces device storage requirements and allows devices to sleep in order to save power. Reliability is in many cases provided by the M2M network. Thus, a huge number of sensors sending data asynchronously with no orchestration to several sinks may have an important impact on congestion leading to data loss. Management messages, flowing to connected devices, are expected to be less frequent and numerous than sensor data and may follow a send-and-receive pattern, so do not need to be proxied.

AFGH has been chosen over BBS since it is designed in such a way that re-encryption with a given key can happen just in one direction such that the proxy is unable to derive the inverse direction key from the first. In the following paragraphs AFGH set up is described.

A map $e: \mathbb{G}_1 x \mathbb{G}_1 \rightarrow \mathbb{G}2$ is a bilinear map if $\forall a,b \in \mathbb{Z}_q$, $x,y \in \mathbb{G}_1$ then $e(x^a,y^b) = e(x,y)^{ab}$ is efficiently computable. AFGH chooses two parameters: g and Z, where $g \in \mathbb{G}_1$and $Z=e(g,g) \in \mathbb{G}_2$. For endpoints, both data source and data sink have to generate a public (PK_a) and private (SK_a) key, for instance a node α1 will have $SK_{\alpha 1}=(\alpha 11,\alpha 12)$ and $PK_{\alpha 1}=(Z^{\alpha 11},g^{\alpha 12})$. The second part of the public key is only send to a partner if α1 accepts to receive proxied reencryptions of messages originally encrypted for the partner.

B. Single Message Encryption

Consider the data originated by a data source called α1 located in a trusted network should be stored by a proxy and later re-encrypted and delivered to α2. α1, as will be discussed later, can be either an end device powerful enough to perform the encryption on its own, or an M2M gateway in the middle mile acting as backhaul that aggregates data from several end devices. The proxy is located in an untrusted network. Both α1 and α2 have a key pairs. α1 protects the message M containing the data resulting in a tuple $C_{1_{\alpha 1}}$ as:

$$C_{1_{\alpha 1}} = [A_1, B_1] = [g^r, M \cdot Z^{\alpha 1 \cdot r}] \qquad (1)$$

The length of parameter r varies according to the desired encryption strength. $C_{1_{\alpha 1}}$ is delivered to the network and stored in an intermediate entity called P. α2 is designated as the data sink. So α1 fetches the public key of α2 (both parts) and computes the re-encryption key $RK_{\alpha 1 \rightarrow \alpha 2}$ as:

$$RK_{\alpha 1 \rightarrow \alpha 2} = g^{\alpha 11 \alpha 22} \qquad (2)$$

The proxy P is instructed to deliver the message to α2 after re-encrypting $C_{1_{\alpha 1}}$ with $RK_{\alpha 1 \rightarrow \alpha 2}$, to do so, it should compute $C_{2_{\alpha1\to\alpha2}}$ as:

$$C_{2_{\alpha1\to\alpha2}} = [A_2, B_2] = [Z^{\alpha22\cdot\alpha11\cdot r}, M\cdot Z^{\alpha11\cdot r}] = \left[Z^{\alpha22\cdot r'}, M\cdot Z^{r'}\right] \quad (3)$$

Finally, $\alpha2$ can acquire the original message M as:

$$\frac{B_2}{A_2{}^{1/SK_{\alpha2}}} = \frac{M\cdot Z^{r'}}{Z^{r'\cdot\alpha22/\alpha22}} = \frac{M\cdot Z^{r'}}{Z^{r'}} = M \quad (4)$$

The destination can be changed at any time (that would require a new RK to be computed.)

C. Data Stream Encryption

Using the aforementioned algorithm, a connected device (end device or gateway) can deliver a protected message allowing untrusted intermediate entities (proxies) to store and forward it later to a recipient. For that reason gateways located in the middle mile may perform this encryption over the data aggregated from several connected devices.

As it has been shown, no previous trust relation is needed so the recipient can be chosen dynamically before or after the message has been encrypted and delivered to the proxy. Despite this provides a good granularity, the high volume of messages makes individual message encryption using asymmetric encryption very expensive in terms of processing power/throughput and battery life.

End devices or gateways are expected to send data regularly to the network containing data related to, for instance, an area, an asset or a variable type. Despite the delivery is performed in an asynchronous way, during a period of time, the whole sequence of data messages can be considered a flow: a sequence of independent pieces of data with the same origin, destination(s) and interest. The proposed proxy re-encryption mechanism for M2M supports the efficient encryption of data streams by protecting individual messages with symmetric encryption.

Every data stream has an associated key stream, a sequence of randomly generated keys. Every key from the key stream is used during a limited period of time so every message containing a piece of data of the stream is encrypted with a session key that changes regularly. Every time the key changes, it is signaled by a management message that is protected with asymmetric encryption as shown in the previous section. So once the data sinks are known, the data messages are forwarded to them, performing a re-encryption over the management messages only, see Fig. 2.

The set of messages within the sequence protected with the same key is denoted as section. Defining sections allows for revoking authorization after a given time.

D. Application Scenarios

Consider the scenario of Fig. 3. Several connected devices are delivering data to the network through their M2M area network (labeled as M2M Prov. B). The gateway $\alpha1$ protects data messages as shown in (1) (or uses stream protection). Protected messages are decrypted at the destination data sink labelled as Backend Cloud A since the data sink can derive the decryption key. Due to several problems the network connecting the M2M provider B and the Backend Cloud A becomes unavailable.

α1, a limited intermediate node, which cannot store high volumes of data discovers an alternative route that uses a secure proxy located at M2M Area network from provider C. This network is untrusted since belongs to a competitor so no unencrypted but protected messages should be sent to the Secure Proxy. α1 sends the messages to the secure proxy to be stored until the network becomes available or a new data sink is designated.

Later on, Backend Cloud B is designated as a new data sink (labeled as α2). α1 derives the re-encryption key $RK_{\alpha 1 \rightarrow \alpha 2}$ appropriately and delivers it to the secure proxy. Once the key is received by the secure proxy, it re-encrypts the messages $C_{1_{\alpha 1}}$ into $C_{2_{\alpha 1 \rightarrow \alpha 2}}$ as in (3), so they can be decrypted at α2, as shown in (4), avoiding secure proxy to access to unencrypted data.

Fig. 4 shows a scenario in with data sinks change dynamically. For clarity, two periods of time t1 and t2 are considered. During t1 connected devices are sending messages through the gateway α1 located at M2M network from provider B. Connected devices provide information about an area of interest so the M2M provider groups that data into a stream. α1 is in charge of aggregating data from different connected devices into the stream.

The data aggregated by α1 is delivered to a secure proxy that interconnects α1 with the Internet. α1 generates a key stream and starts encrypting messages with symmetric encryption. Every message is labeled with a streamId and the key index, within the key stream, that is used for encrypting that concrete message. α1 generates management messages containing the streamId and the key from the key stream used at that time, protects it according to (2) and injects them periodically into the stream. During the period t1, α2 is designated as data sink so the secure proxy forwards the stream to α2 and re-encrypts management messages containing the keys from the key stream so α2 can decrypt the management messages containing the keys to decrypt the data.

The M2M provider advertises the data it can aggregate to its partners or customers. The backend cloud B is interested in the data so, after t1, the M2M provider designates α3 as an additional data sink. Thus, during t2, data messages are delivered to both α2 and α3. α1 generates a re-encryption key for α3 and management messages are then re-encrypted also for α3 according to (3).

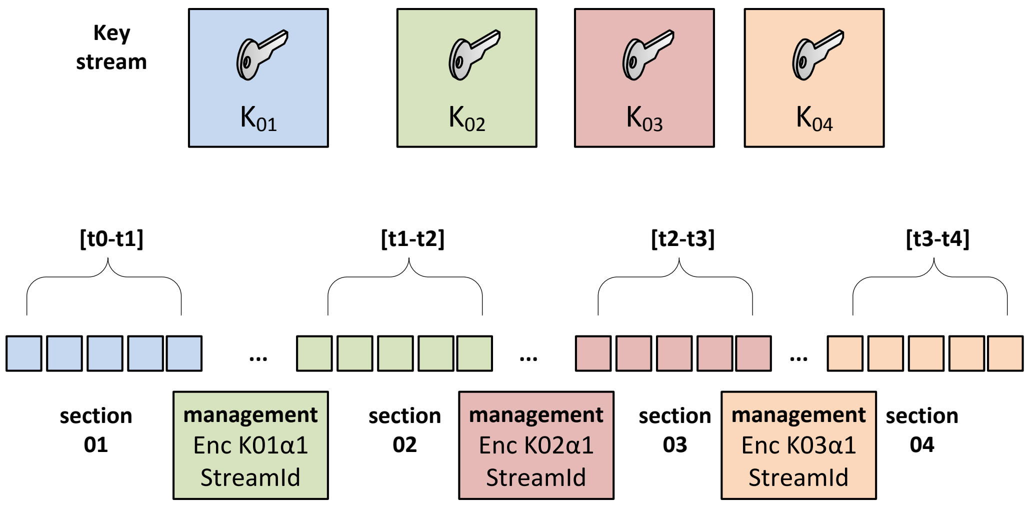

Fig. 2. Stream encryption. Data messages are encrypted with a session key using symmetric encryption and management messages containing the session key, that has been taken from a key stream, are protected according to the proxy re-encryption algorithm.

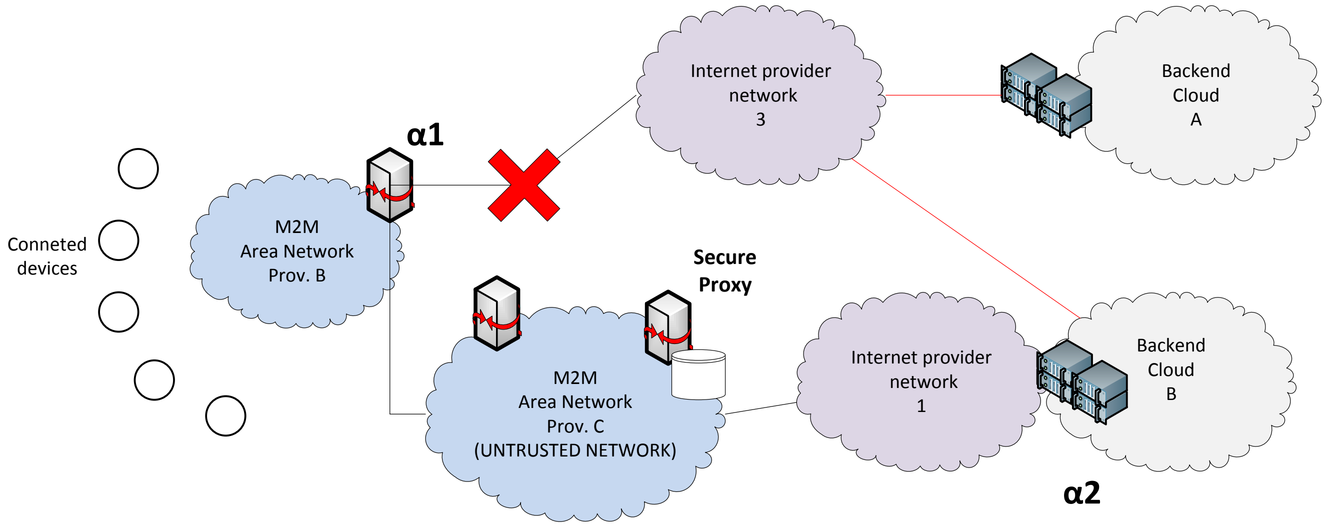
Internet provider network 3
Backend Cloud A
α1
Conneted devices
M2M Area Network Prov. B
Secure Proxy
M2M Area Network Prov. C (UNTRUSTED NETWORK)
Internet provider network 1
Backend Cloud B
α2

Fig. 3. The backend cannot be reached so data messages need to be stored in an intermediate proxy. Later on, data is re-encrypted and delivered to a new destination α2.

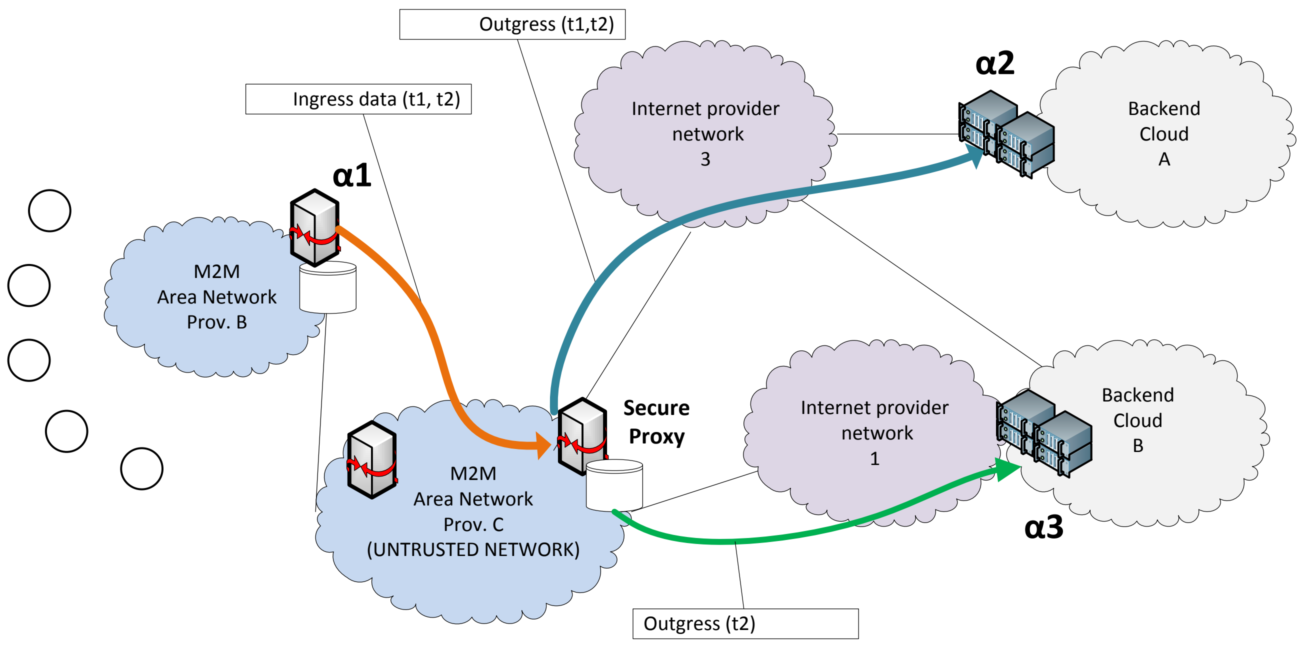

Fig. 4. α2 is the data sink to which data is delivered during a period of time (t1). A new data sink α3 is added dynamically later (t2).

E. Protocol Mapping

The design has considered CoAP and UDP as transports for the protected messages, which is consistent with several embodiments of M2M [46]. Both protocols allow for asynchronous delivered messages by constrained devices. CoAP provides additional reliability that is especially interesting for radio interfaces. Protected messages carried over UDP use a binary header, its meaning depends if messages are protected independently or part of a stream. For independently protected messages the binary header contains a 64 bit random number, a URL identifying the data source, a URL identifying the data sink or, if unknown before sending the message, a URL identifying within a namespace belonging to the provider. If messages are not handled individually at the proxy but are part of a stream, then they are flagged accordingly. Moreover, the 64 bit number is, in this case, divided in three parts: a 32 bit random number that identifies the message, a 16 bit number that corresponds to the streamID, and a 16 bit number (key index) that identifies the key within the key stream assigned to that data stream section.

Management messages contain important information for re-encryption. Once the data sink is determined, either the data source or other trusted entities with access to the keys generate the re-encryption key and send it to the secure proxy for re-encrypting the messages. Management messages contain a header with a URL identifying the data source a streamID and a key index if applicable.

If data is transported over CoAP instead, the header is directly mapped into several CoAP fields and options. The protocol uses the CoAP token, that correlates requests and responses to hold the 32 bit number used in either individual encryption or stream encryption as shown in Fig. 5. Data sink and source URL are placed into CoAP options for holding the Request URI [18] for the first and additional option (created for this protocol) to identify the source's callback URL. When delivered to a secure proxy, protected messages use the Proxy-Uri to identify the proxy as a forward proxy.

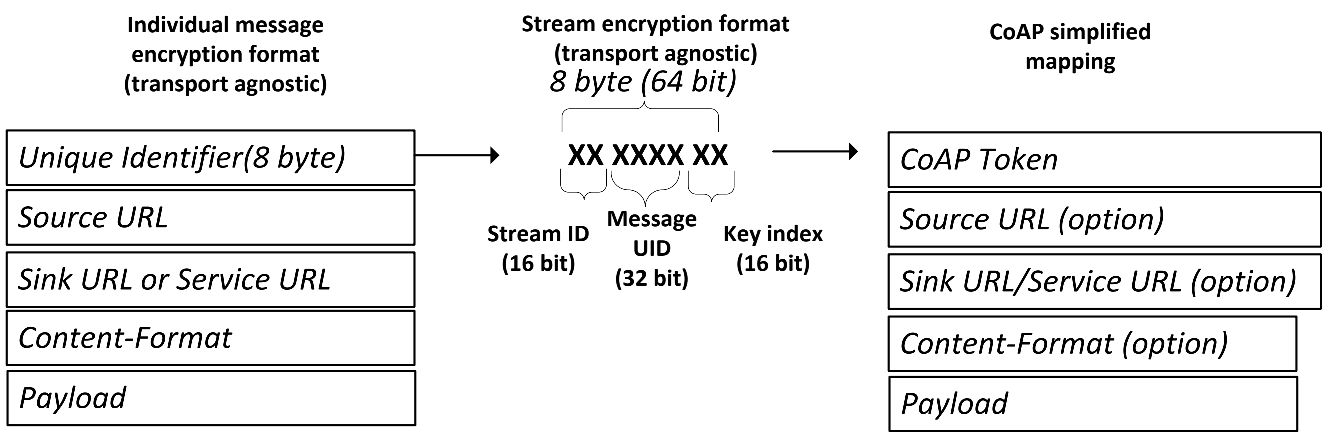

Fig. 5. Simplified secure proxy message format and mapping.

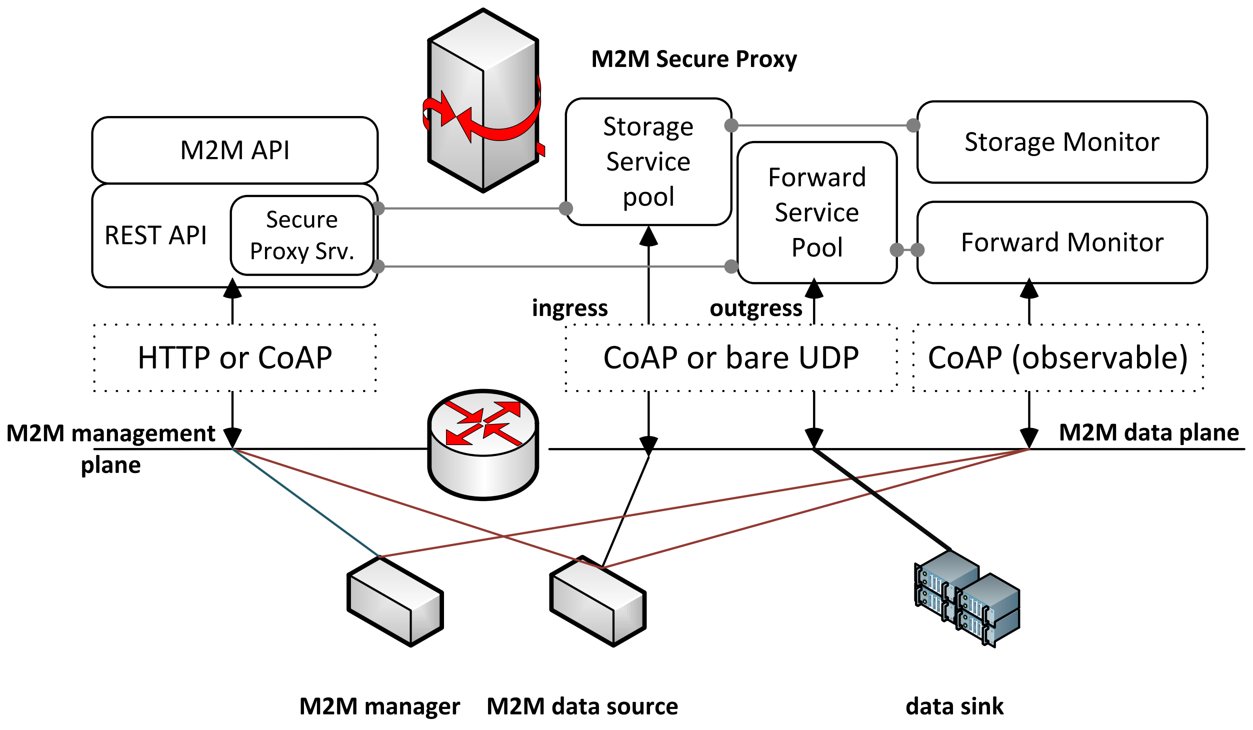

Fig. 6. Secure proxy architecture.

IV. Secure Proxy Architecture and Operation

The proposed secure proxy is an intermediate entity that has available storage and cryptographic services. The secure proxy should be known to the network management applications so it can be used when needed. Nevertheless, it could respond to discovery protocols for opportunistic users. Discovery, service agreement and reimbursement calculations for using the proxy are out of the scope of this article.

A. Architecture

The architecture of the secure proxy is shown in Fig. 6. The M2M node acting as proxy has a RESTful API accessible either using HTTP or CoAP. The API allows obtaining information about the secure proxy, for instance, the available storage. This method returns the storage capacity and the available space at that time. The load method allows checking the load of the proxy during a time window. The average throughput allows determining the rate this secure proxy can forward data to the data sink.

The API exports functionality to use the secure store and forward service. The store method instructs the secure proxy to reserve space for upcoming messages. Once invoked, this method creates a storage resource identifiable by an URL with a unique id. Messages will be delivered to the proxy for storage by sending CoAP POST containing the message to store to a sibling URL of the storage resource. Under the storage resource URL created by the proxy, a monitor resource that can be observed [47] provides statistics about the service. The monitor resource can be retrieved and updated by the server over a period of time.

For plain UDP a datagram socket is created for receiving the messages after calling the store method (that should be called using CoAP). Once the messages have been stored by the proxy and the final destinations are known, the messages can be delivered to the destinations after re-encryption. To do so, the API provides the forward method. It consumes the re-encryption keys and the destination URLs (or UDP endpoints) and the storage resource unique identifier. The delivery rate can also be defined. Several other optimizations as compressing or grouping messages are considered. A unique id for the forward operation is also generated and a URL for the operation is created. Under the generated URL two sibling services are instantiated, one for monitoring that allows the resource to be observed and other for controlling the forward operation including pausing, resuming and changing the destination data sinks and re-encryption keys.

B. Operation

This section describes the message flow in a possible scenario using CoAP. α1 is the entity protecting messages. It can be either an end device or an M2M gateway aggregating others end devices' traffic. If α1 acts as a gateway, M2M applications behind it that are sending batch traffic should terminate the request/response in α1 (first hop after the radio link) or clear transmission buffers after confirmation from α1 leaning on the M2M network reliability. Otherwise, limited devices will be forced to hold the request in the buffer until the message arrives to the final destination that can happen a long time later if messages are stored in a proxy. Consider α1 as a gateway delivering aggregated data as a stream to an endpoint until it fails.

α1 can discover a gateway offering the secure store and forward service, or be instructed by other M2M entity with management attributions. Once the secure proxy is known to α1, it uses the proxy API to create a storage resource. After a successful response from the proxy, α1 generates a key stream to protect messages. Once α1 has data ready to be sent, it sends the data to the storage resource created previously at the proxy. α1 can monitor the entire proxy by subscribing to the store monitor [47]. α1 generates regularly a protected management message containing the session key in use.

Once a new data sink and its public key is known to α1, α1 computes the re-encryption key and invokes the forward method of the secure proxy API passing the re-encryption key $RK_{\alpha 1 \rightarrow \alpha 2}$ as a parameter. This method can also be invoked by any other authorized M2M entity. A forward resource and a monitor are created as a result. Once α1 receives a positive response from the secure proxy, it can monitor the forward process and instruct the secure proxy to deliver the data to the final destination. To do so, the secure proxy forwards the messages and re-encrypts the session keys from the key stream found in the management messages.

Finally, once data and management messages arrive at the destination, the destination decrypts the management messages to access the session keys and decrypts the data messages with the session key.

V. IMPLEMENTATION AND RESULTS

A major concern of this research was to identify if typical constrained CE devices could deal with this strong encryption and if the throughput is reasonable for the intended application. To evaluate this, two candidate consumer electronics device profiles have been considered: end devices encrypting messages individually that can be easily found in any plausible scenario and small-medium gateways acting as backhaul receiving data from LRWPANs and forwarding them using long range radio interfaces or terrestrial networks. Both devices are usually implemented on commodity hardware and, thus, present in target scenarios. The devices would make either the initial encryption (end devices) or the re-encryption (gateways).

For the evaluation, a half credit card size micro computer equipped with a RISC single core processor and 1GB of ram has been used. The device is connected in two ways to a traffic generator: 1) using 6LoWPAN over an 802.15.4 and 2) 6LoWPAN over Ethernet (or 802.11.) The operating system is based on linux and uses open source drivers. In the first case, the effective Maximum Transfer Unit (MTU) is limited to 127 bytes and in the second case the MTU is 1280. Two key lengths have been used. For medium encryption r has been set to 512. For medium-strong encryption r has been set to 1536. In both cases CPU usage was stable and equal to 24.2% for asymmetric encryption with AFGH. The memory footprint was 18.5Mb of RAM for medium-strong encryption and 16Mb for medium encryption.

According to Fig. 7 and Fig. 8, encryption is more computationally expensive than re-encryption. Considering the CPU consumption, a proxy would be able to re-encrypt and aggregate traffic from up to 4 streams simultaneously. This shows the solution is appropriate for the cases discussed previously in the article since the most computationally expensive tasks are performed at the proxy, that is expected to be more powerful than constrained end devices. Results show it is possible to manage up to 4 streams simultaneously giving up to 800kbps of aggregated traffic for r=512 and MTU=1280 and 200kbps for r=512 and MTU=127, that is appropriate for asynchronous monitoring applications considering the constraints of the hardware used during the evaluation. The network MTU has an almost linear impact in the throughput of the encrypted streams, as expected.

The 6LoWPAN setup over 802.15.4 shows a small throughput due to the reduced MTU value imposed by the physical layer. The throughput for 6LoWPAN over less constrained physical layers shows significantly improved behavior.

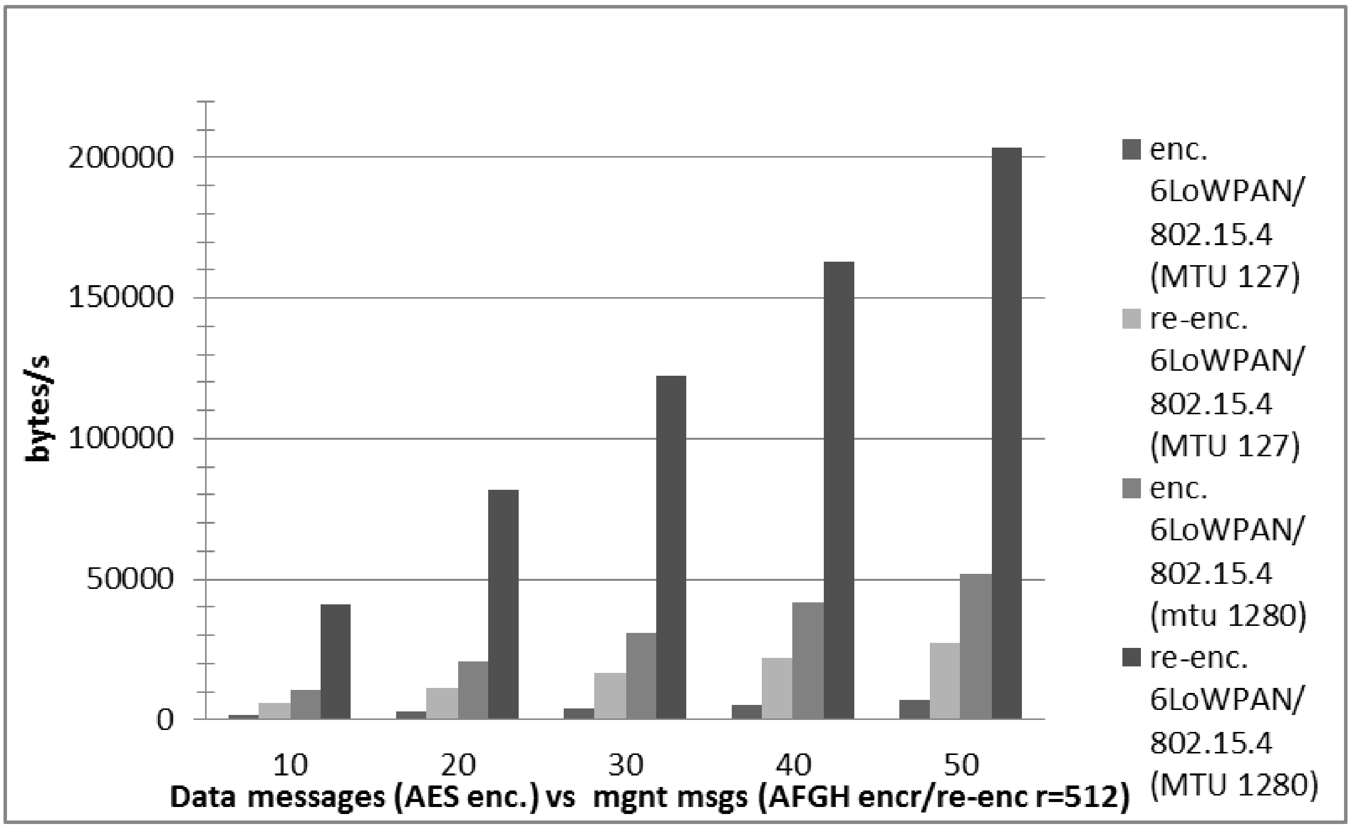

Fig. 7. Throughput (total data delivered divided by total processing time) for stream encryption with r=512. It has been calculated for several combinations of data messages (encrypted with AES) per management message (encrypted with AFGH-containing the AES session key).

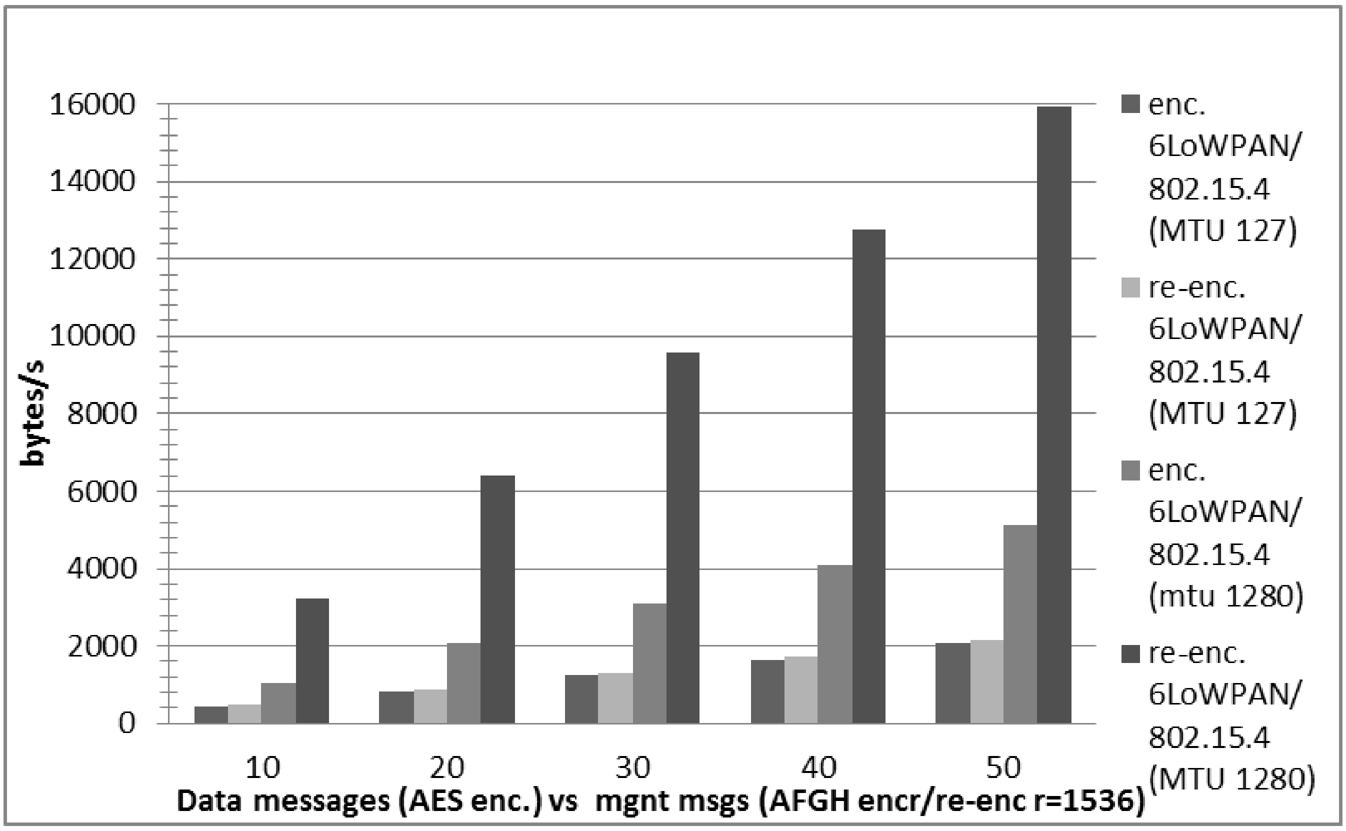

Fig. 8. Throughput calculated for stream encryption for r=1536.

VI. Conclusions

This research describes the design, architecture and operation of a secure store and forward service for intermediate M2M proxies and its protocol bindings. The major purpose of this solution is to avoid M2M traffic collapsing the network while also maintaining confidentiality. It has been carefully designed to be useful for limited CE devices and constrained physical layers and protocols. The solution allows storing valuable data in untrusted intermediate entities in such a way they can be eventually delivered to the appropriate data sink without letting the untrusted entity to access the data. The data can be re-encrypted to the appropriate destination at the untrusted entity without revealing the content by using a re-encryption key. It allows also sending data to several destinations and changing the destination dynamically using different keys. A further advantage is that the re-encryption throughput is 4-8 times more than that of the encryption process. A prototype has been developed and tested showing it can be deployed in target CE devices providing an adequate throughput.

References

[1] K. Ashton, “That ‘internet of things’ thing,” *RFID J.*, vol. 22, no. 7, pp. 97-114, Jun. 2009.

[2] P. Verma, R. Verma, A. Prakash, A. Agrawal, K. Naik, R. Tripathi, M. Alsabaan, T. Khalifa, T. Abdelkader, and A. Abogharaf, “Machine-to-Machine (M2M) communications: a survey,” *J. Network and Computer Applications*, vol. 66, pp. 83-105, May 2016.

[3] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, “Internet of things for smart cities,” *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22-32, Feb. 2014.

[4] D. C. Brabham, “Crowdsourcing as a model for problem solving: an introduction and cases,” *Convergence: The Int. J. Research into New Media Technologies*, vol. 14, no. 1, pp. 75-90, Feb. 2008.

[5] R. Glitho, “Application architectures for machine to machine communications: research agenda vs. state-of-the art,” in Proc. 6th IEEE International Conference on International Conference on Broadband Communications and Biomedical Applications, Melbourne, VIC, pp. 1-5, Nov. 2011.

[6] M. Cavada, C. Rogers, and D. Hunt, “Smart cities: contradicting definitions and unclear measures,” in Proc. 4th World Sustainability Forum, f004, Nov. 2014.

[7] M. Deakin and H. Al Waer, “From intelligent to smart cities,” *Intelligent Buildings International*, vol. 3, no. 3, pp. 140-152, Aug. 2011.

[8] B. Guo, Z. Yu, X. Zhou, and D. Zhang, “From participatory sensing to mobile crowd sensing,” in Proc. IEEE International Conference on Pervasive Computing and Communication Workshops, Budapest, pp. 593-598, Mar. 2014.

[9] R. Hollands, “Will the real smart city please stand up?” *City*, vol. 12, no. 3, pp. 303-320, Nov. 2008.

[10] P. Ballon, J. Glidden, P. Kranas, A. Menychtas, S. Ruston, and S. van der Graaf, “Is there a need for a cloud platform for European smart cities?” in Proc. IIMC eChallenges e-2011, Florence, Italy, pp. 1-7, Oct. 2011.

[11] A. Coe, G. Paquet and J. Roy, “E-governance and smart communities: a social learning challenge,” *Social Science Computer Review*, vol. 19, no. 1, pp. 80-93, Spring 2001.

[12] J. Holler, *From machine-to-machine to the internet of things*, 1st ed., Academic Press, 2014.

[13] A. Weiss, “Computing in the clouds,” *netWorker*, vol. 11, no. 4, pp. 16-25, Dec. 2007.

[14] D. Evans, “The internet of things: how the next evolution of the internet is changing everything,” Cisco Internet Business Solutions Group, 2011.

[15] Cisco Systems, “Global Mobile Data Traffic Forecast Update, 2015–2020,” Document ID:1454457600805266, Feb. 2016.

[16] V. Galetić, I. Bojić, M. Kušek, G. Ježić, S. Dešić, and D. Huljenić, “Basic principles of machine-to-machine communication and its impact on telecommunications industry,” in Proc. 34th IEEE International Convention on Information and Communication Technology, Electronics and Microelectronics, Opatija, Croatia, pp. 380-385, May 2011.

[17] H. Kim and N. Feamster, “Improving network management with software defined networking,” *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 114-119, Feb. 2013.

[18] Z. Shelby, K. Hartke, and C. Bormann, “The constrained application protocol (CoAP),” Internet Engineering Task Force, RFC 7252, Jun. 2014.

[19] D. Díaz-Sánchez, R. S. Sherratt, P. Arias, F. Almenares, and A. M. López, “Proxy re-encryption schemes for IoT and crowd sensing,” in Proc. IEEE International Conference on Consumer Electronics, Las Vegas, USA, pp. 15-16, Jan. 2016.

[20] K. Gill, S. H. Yang, F. Yao, and X. Lu, “A zigbee-based home automation system,” *IEEE Trans. Consum. Electron.*, vol. 55, no. 2, pp. 422-430, May 2009.

[21] IEEE Computer Society, “Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs),” IEEE 802.15.4 Specification, 2006.

[22] D-M. Han and J. H. Lim, “Smart home energy management system using IEEE 802.15.4 and zigbee,” *IEEE Trans. Consum. Electron.*, vol. 56, no. 3, pp. 1403-1410, Aug. 2010.

[23] A. Usman and S. Shami, “Evolution of Communication Technologies for Smart Grid applications,” *Renewable and Sustainable Energy Reviews*, vol. 19, pp. 191-199, Mar. 2013.

[24] M. Ghamari, B. Janko, R. S. Sherratt, W. Harwin, R. Piechockic, and C. Soltanpur, “A survey on wireless body area networks for eHealthcare systems in residential environments”, *Sensors,* vol. 16, no. 6, 831, pp. 1-33, June 2016.

[25] R. T. Fielding, “Architectural styles and the design of network-based software architectures”, PhD thesis, University of California, Irvine, 2000.

[26] N. Kushalnagar, G. Motenegro, and C. Schumacher, “IPv6 over low-power wireless personal area networks (6LoWPANs): overview, assumptions, problem statement, and goals”, Internet Engineering Task Force, RFC 4919, Aug. 2007.

[27] D. Karaman, N. Gözüaçık, M. O. Alagöz, H. İlhan, U. Çağal and O. Yavuz, “Managing 6LoWPAN sensors with CoAP on internet”, in Proc. 23nd IEEE Signal Processing and Communications Applications Conference, Malatya, pp. 1389-1392, May 2015.

[28] S.-C. Son, N.-W. Kim, B.-T. Lee, C. H. Cho, and J. W. Chong, "A time synchronization technique for CoAP-based home automation systems," *IEEE Trans. Consum. Electron.*, vol. 61, no. 1, pp. 1403-1410, Aug. 2010.

[29] Z. M. Fadlullah, M. M. Fouda, N. Kato, A. Takeuchi, N. Iwasaki, and Y. Nozaki, "Toward intelligent machine-to-machine communications in smart grid," *IEEE Commun. Mag*., vol. 49, no. 4, pp. 60-65, April 2011.

[30] R. Lu, X. Li, X. Liang, X. Shen, and X. Lin, "GRS: the green, reliability, and security of emerging machine to machine communications," *IEEE Commun. Mag*., vol. 49, no. 4, pp. 28-35, April 2011.

[31] K. Zheng, S. Ou, J. Alonso-Zarate, M. Dohler, F. Liu, and H. Zhu, "Challenges of massive access in highly dense LTE-advanced networks with machine-to-machine communications," *IEEE Wireless Commun*., vol. 21, no. 3, pp. 12-18, June 2014.

[32] E. Rescorla and N. Modadugu, "Datagram transport layer security version 1.2," Internet Engineering Task Force, RFC 6347, Jan. 2012.

[33] S. Raza, D. Trabalza, and T. Voigt, "6LoWPAN compressed DTLS for CoAP," in Proc. 8th IEEE International Conference on Distributed Computing in Sensor Systems, Hangzhou, China, pp. 287-289, May 2012.

[34] S. Raza, T. Voigt, and U. Roedig, "6LoWPAN extension for IPsec," in Proc. IETF-IAB Workshop Interconnecting Smart Objects with the Internet, Prague, Mar. 2011.

[35] H. V. Nguyen and L. L. Iacono, "REST-ful CoAP message authentication," in Proc. IEEE International Workshop on Secure Internet of Things, Vienna, pp. 35-43, Sept. 2015.

[36] R. Roman, P. Najera, and J. Lopez, "Securing the internet of things," *IEEE Computer*, vol. 44, no. 9, pp. 51-58, Sept. 2011.

[37] A. Sarma and J. Girão, "Identities in the future internet of things," *Wireless Personal Commun*., vol. 49, no. 3, pp. 353-363, Mar. 2009.

[38] Modbus, I. D. A. "Modbus application protocol specification v1. 1a," North Grafton, MA, 2004.

[39] K. Bender, *Profibus: the fieldbus for industrial automation*, Prentice-Hall, Inc. Upper Saddle River, NJ, 1993

[40] S. L. Keoh, K. W. K. Au, and Z. Tang, "Securing industrial control system: an end-to-end integrity verification approach," in Proc. Industrial Control System Security Workshop, Los Angeles, CA, USA, Dec. 2015.

[41] M. Mambo and E. Okamotom, "Proxy cryptosystems: delegation of the power to decrypt ciphertexts," *IEICE Trans. Fund. Electronics Communications and Computer Science*, vol. E80-A, no. 1, pp. 54-63, Jan. 1997.

[42] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in Proc. Eurocrypt, Espoo, Finland, pp 127-144, May/June 1998.

[43] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Trans. Information and System Security*, vol. 9, pp 1-30, Feb. 2006.

[44] D. Nunez, I. Agudo, and J. Lopez, "Integrating openid with proxy re-encryption to enhance privacy in cloud-based identity services," in Proc. 4th IEEE International Conference on Cloud Computing Technology and Science, Taipei, pp. 241-248, Dec. 2012.

[45] D. Boneh, and M. Franklin, "Identity-based encryption from the Weil pairing," *SIAM Journal of Computing*, vol. 32, no, 3, pp. 586-615, 2003

[46] C. Gezer and E. Taşkın, "An overview of oneM2M standard," in Proc. 24th IEEE Conference on Signal Processing and Communication Applications, Zonguldak, pp. 1705-1708, May 2016.

[47] K. Hartke, "Observing resources in the constrained application protocol (CoAP)," Internet Engineering Task Force, RFC 7641, Sept. 2015.

Daniel Diaz-Sánchez (M'07-SM'15) is an associate professor at University Carlos III of Madrid. His research interests include distributed authentication/authorization, content protection, distributed computing, fog computing, IoT and Smart Cities. Díaz-Sánchez received a PhD in telematics engineering from UC3M.

R. Simon Sherratt (M'97-SM'02-F'12) received the B.Eng. degree in Electronic Systems and Control Engineering from Sheffield City Polytechnic, UK in 1992, M.Sc. in Data Telecommunications in 1994 and Ph.D. in video signal processing in 1996 from the University of Salford, UK.

In 1996, he has appointed as a Lecturer in Electronic Engineering at the University of Reading where he is now Professor of Biosensors. His research topic is signal processing and communications in consumer devices focusing on wearable devices and healthcare. He received the 1st place IEEE Chester Sall Memorial Award in 2006 and 2nd place in 2016. He is a reviewer for the IEEE SENSORS JOURNAL is currently the Editor-in-Chief of the IEEE TRANSACTIONS ON CONSUMER ELECTRONICS.

Florina Almenárez (M'07) is an associate professor at University Carlos III of Madrid. Her research interests include trust and reputation management models, identity management, and security architectures in ubiquitous computing. Almenárez received a PhD in telematics engineering from UC3M.

Patricia Arias-Cabarcos (M'07) is a researcher at the University Carlos III of Madrid (UC3M). Her interests include identity management, trust models, and risk assessment. Arias-Cabarcos received a PhD in telematics engineering from UC3M.

Andrés Marín Lopez (M'07) is an associate professor at University Carlos III of Madrid. His research interests include ubiquitous computing: limited devices, trust, and security in next-generation networks. Marín received a PhD in telecommunication engineering from Universidad Politécnica of Madrid