A distributed approach for secure M2M communications

Yosra Ben Saied, Alexis Olivereau, Maryline Laurent

A Distributed Approach for Secure M2M Communications

Yosra Ben Saied and Alexis Olivereau
CEA, LIST, Communicating Systems Laboratory,
91191 Gif-sur-Yvette CEDEX, FRANCE
{yosra.ben-saied, alexis.olivereau}@cea.fr

Maryline Laurent
Telecom SudParis,
91011 Evry, FRANCE
maryline.laurent@it-sudparis.eu

***Abstract*—A key establishment solution for heterogeneous Machine to Machine (M2M) communications is proposed. Decentralization in M2M environment leads to situations where highly resource-constrained nodes have to establish end-to-end secured contexts with powerful remote servers, which would normally be impossible because of the technological gap between these classes of devices. This paper proposes a novel collaborative session key exchange method, wherein a highly resource-constrained node obtains assistance from its more powerful neighbors when handling costly cryptographic operations. Formal security analysis and performance evaluation of this method are provided; they confirm the safety and efficiency of the proposed solution.**

M2M; key establishment; resource constraints; energy efficiency; formal security analysis; AVISPA

I. INTRODUCTION

The Machine to Machine (M2M) paradigm can be characterized by three main features. First, it involves a highly diversified pool of components, ranging from low-resource sensors to powerful servers and distributed over a large geographical environment. Second, it emphasizes increased autonomy, as compared with legacy Internet. While all of the M2M systems are designed to provide decentralization and minimize the requirement of human activity, most advanced ones may even implement functions of situation awareness, self-organization or cognition. Finally, M2M systems adopt a distributed communication model wherein any two nodes may establish relationship with one another, provided that one is offering the service, or resource, which is needed at the other end.

To that respect, M2M systems differ from legacy Wireless Sensor Networks (WSN), which preceded them. Contrary to what happens in WSNs, the communication path between two nodes does not have to follow a hierarchical path, e.g. from sensor to sink, and from sink to server. A sensor in an M2M environment will have relationships with other peers irrespective of their distance, role and capabilities, provided that these relationships are desirable from the viewpoint of the M2M topology.

This novel paradigm, wherein nodes communicate with a large set of peers through a decentralized pattern, leads to situations where a highly resource-constrained node has to exchange data with a much more powerful server. When these data have to be secured, a key establishment mechanism must be run between both peers in order to agree on a session key. Nodes heterogeneity may make this key set-up phase very difficult, if not impossible. We proposed in [1] a solution based on the collaborative use of a system inspired by TLS simple handshake in one-pass key transport mode. Relying on a similar collaboration scheme, we now present a new approach in which the session key derivation has been enforced and for which we provide the result of a formal security analysis, as well as an initial performance analysis.

Section 2 of this paper presents the key establishment problem and points out the inadequacies of the related work. Section 3 describes the proposed cooperative key establishment solution for M2M networks. This solution enables a highly resource limited device to benefit from the assistance of neighboring peers for taking charge of the key exchange computational load, on a distributed and cooperative basis. Section 4 provides the results of the security analysis that we carried out on our solution, as well as those of the performance analysis. Section 5 concludes this paper.

II. PROBLEM STATEMENT

In this paper we consider a highly constrained node that needs to establish a secure end-to-end communication with a remote server within a heterogeneous M2M architecture. The resource-constrained client has to transmit sensitive data to the server in an end-to-end manner without conveying them to an intermediary entity such as the sink node in legacy sensor networks, whereas the two peers do not share any prior authentication information to protect the data exchange.

In order to secure their communication, the two peers need to establish a shared secret key as an initial phase. The set up of this shared key is considered as a crucial challenge due to the unbalanced resources capabilities of the communicating entities. The remote server requires asymmetric cryptography primitives to authenticate the client and exchange data, while we assume that this latter cannot perform heavy cryptographic operations due to its limited resources in storage, energy, and computation.

Even though there exist in the literature some studies that propose efficient key establishment schemes in sensor networks, the design of an efficient key establishment scheme that addresses these heterogeneous M2M communications has,

This work was financially supported by the EC under grant agreement FP7-ICT- 257521 IoT-A project.

to the best of our knowledge, not been undertaken yet. Most existing key establishment approaches rely on symmetric cryptography primitives due to their reasonable resources consumption. Such solutions [2] are considered more efficient for sensor nodes. However, in view of our envisioned M2M scenario, symmetric key based schemes are not applicable when a sensor node wishes to communicate with external entities since they are based on pre-distributed shared keys.

Public Key Cryptography (PKC) solutions were first considered impractical on sensors devices with limited resources. Things changed with the proposal of lightweight PKC algorithms for the sensors networks. Elliptic curve cryptography (ECC) is generally favored due to its lower energy consumption as compared with other PKC algorithms [3] [4]. Other solutions focus on making the well-established asymmetric cryptosystem RSA more adapted to resource-restrained devices using a small RSA public exponent (e) and a short key size [5]. This advantage comes, however, at the price of a lower security level [6]. Hardware solutions are also proposed aiming to extend computational capabilities of a standard node through low power hardware modules in order to make the use public key cryptography practical on sensor nodes. Nevertheless, in practice, required energy consumption and memory costs of these lightweight asymmetric key based solutions are still non negligible [7] [8] and would be hindering for highly resource-constrained nodes.

Other hybrid schemes combining both symmetric and asymmetric cryptography have been proposed. This hybrid proposal aims to reduce the heavy cost of public key operations by replacing some operations with symmetric-key based ones and hence merging the advantages of both approaches. Two families of hybrid solutions can be distinguished; the first one is based on a translating entity at border between "symmetric" and "asymmetric" domains [9]. The second one replaces expensive cryptography operations in an asymmetric algorithm with symmetric ones [10]. In the first case, security is provided on a hop-by-hop basis. Confidentiality and availability are thus compromised since the intermediary translator introduces potentially both a security flaw and a single point of failure. In the second case, communications with an external party as considered in our M2M scenario are not possible, since it requires that both peers share a symmetric key.

III. PROPOSED SOLUTION

The shortcomings of the above key establishment solutions led us to propose an efficient collaborative key establishment scheme for heterogeneous M2M communications that permits the creation of end-to-end secure associations between two nodes with unequal resource capabilities and that is based on asymmetric primitives.

The heavy cost of cryptographic operations inhibits the key establishment on resource-constrained nodes and makes them unable to establish associations with peers. We propose that the resource-constrained client assigns its computational charge of asymmetric cryptography to less constrained nodes at neighborhood. These nodes called proxies will take charge of the key exchange process in a distributed and collaborative manner.

The solution is a two-pass key transport protocol, based on an exchange of secret values between the highly resource constrained (client) node and the resource-unconstrained server. In order to push its secret value, the client splits it into multiple parts. Each proxy encrypts a part and delivers it to the server. Upon reception of the client's secret value, the server securely transmits its own secret value to the proxies, which ensure its delivery to the client.

Several constraints have been considered in the design of the protocol. The collaborative approach must not come at the expense of the risk of a key disclosure or a collusion attack. In addition, each proxy is required to prove its legitimacy to the server by proving that it is authorized to act on behalf of the client.

Considering a highly resource-constrained node A that needs to communicate directly with a powerful server B, the process of our proposed protocol is composed of the following parts:

- Selection of the supporting nodes (proxies) P_i at A to set up the session key with B.
- Retrieval of the required key materials by the supporting nodes. These key materials are needed to make them able to compute a signature on behalf of A.
- Preparation and split phases of the secret value *x* generated by A, and delivery to the proxies, followed by the secure transport of different segments of the secret key from each proxy to B.
- Validation of the different messages received from proxies at B and reassembling to retrieve the secret key *x*.
- Secure delivery of the *x*-encrypted secret value *y* generated by B to A through proxies P_i.
- Computation of the session key at A and B.

A. Assumptions

The assumptions made in the design of our protocol are listed in what follows:

- The considered network model is an M2M environment that interconnects heterogeneous nodes with different capabilities in terms of energy, memory and computational power. We especially distinguish in this paper three different platforms among them: (1) highly resource-constrained sensor nodes; (2) other sensor nodes, less constrained, able to perform with restricted asymmetric cryptographic operations; and (3) M2M nodes with high energy, computational power and storage capabilities (e.g. line-powered remote servers).
- After the initialization phase, every sensor node shares pairwise keys with a subset of its one-hop neighbors (within the same radio range). These keys may have been generated during a specific bootstrapping phase using a trusted key management server or through

more subtle mechanisms such as transitive imprinting [11].

- The highly resource-constrained node is able to identify a set of less resource-constrained nodes that are available for supporting expensive cryptographic functions on its behalf.
- There exists a trusted entity within the sensor network that owns a shared secret with all nodes in the sensor network and a public/private key pair. This entity may be logical/distributed.
- Powerful M2M nodes do not communicate with the sensor network trusted entity but are statically configured with or able to validate its public key.
- Powerful M2M nodes trust the sensor network trusted entity to faithfully guarantee that a node within the sensor network has the right to compute cryptographic operations on behalf of another.

B. Protocol Description

The protocol is initiated with a first exchange between A and B to negotiate security capabilities and the proposed collaborative key exchange. Once the server B agrees on the client A's request, the following protocol phases can be started. The exchange includes also random values (Na, Nb) used as nonces to compute the session key.

1) Preparation of the Involved Entities

As an initial phase, the client carefully selects the $P_1 \dots P_n$ proxies it will require assistance from to support its key exchange based on a trust model that monitors the reputation of the nodes in the network and their actual resource capabilities. These nodes will then contact the server and send it messages on behalf of the client. Hence authorization and authentication questions arise at the proxy side, since the proxies must prove on one hand the integrity of the sent messages and on the other hand their representativeness of the client. For this purpose, we propose to assign each proxy an ephemeral pair of private and public keys, using the lightweight one-time signature scheme of Lamport [12] in order for the proxy to sign messages on behalf of the client.

The computational load corresponding to the generation of those key pairs is moved from the proxies to the sensor network trusted entity T, which is also the only entity able to assert that a proxy node is authorized to use its ephemeral private key to sign on behalf of the client. Therefore, the delivery of ephemeral key pairs to all proxies begins with the client informing the trusted entity T of their respective identities, along with the size of the secret key that will be transmitted to the server. T requires this latter since in the one-time signature scheme, the lengths of the public and private keys depend on the size of the message to sign.

Upon receiving the identities of the selected proxies, T generates a pair of private/public key (LK_i^{-1}, LK_i) and an identifier ID_i for each proxy P_i. The identifier ID_i is built as follows: $ID_i=H(LK_i)$ with H() being a one-way hash function, in order for the identifier to have a reduced size when exchanged between nodes. Then, T sends the list of the proxies' identifiers to A. At the end of the key establishment procedure, this list information contained in these triplets will be used by A to identify the well behaving proxies in the list of participating nodes' identifiers sent by B.

T provides then securely each proxy with the key material required to sign on behalf of the client. This consists of a one-time private key and the associated public key, this latter being signed by T along with a mention of A's identity. Actually, the public keys sent by T to all proxies are not individually signed since the signature and verification of each signature would be heavy for respectively T and B. Instead, the public keys are grouped within a Merkle tree [13] so that B has only to verify the signature of the Merkle tree's root MT_{Root} to authenticate all the proxies' public keys. Upon receiving their key material, proxies are prepared to participate to the collaborative establishment of the session key and each proxy P_i contacts B to request for B's certificate and to provide it with its own one-time Lamport public key LK_i.

At this stage, it is worth noting that the key generation at the trusted entity T only consists of a facility for our proposed system, aiming at making our solution more efficient. It is not considered as a solution enabler since proxies could reasonably perform a asymmetric cryptography operation with –authorized– classical key material, as they are less resource-constrained than the client when they are selected for the key establishment assistance.

This phase of the proposed solution, corresponding to the preparation of the involved entities, is depicted in Figure 1.

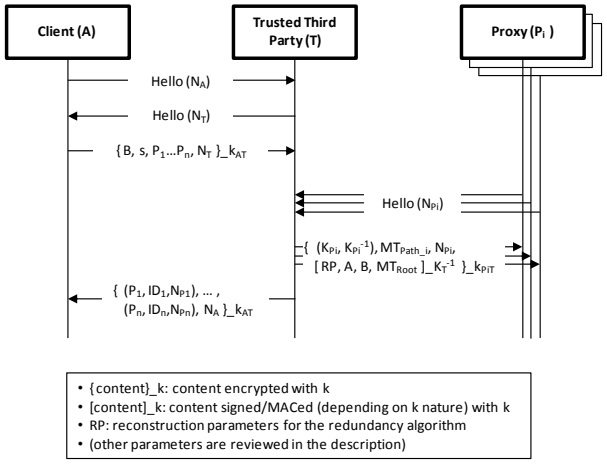

Figure 1. Preparation of the involved entities.

2) Secret Key Exchange

A random secret key *x* generated by the client and a second random secret value *y* generated by the server are used to compute the session key.

The client first applies an error redundancy scheme to the original message *x*, splitting it into *n* parts $x_1, \dots, x_n$ and then sends each message x_i to the corresponding proxy P_i. The error correction scheme is used to make the recovery of the secret *x* possible at the server if a sufficient number of packets from supporting nodes is received, without requiring the reception of all of them. This system protects our solution from

misbehaving supporting nodes, which may refuse to send their parts of the secret *x*. In addition, it does not impose reliable delivery in each proxy→server connection. In the proposed solution, we choose to rely on a Reed-Solomon code [14], which is a widely-used non-binary cyclic error-correcting code

Upon reception of the part of the secret key x_i, the proxy P_i encrypts it using server's public key and signs the result using its one-time Lamport private key. The proxy then sends the encrypted x_i along with its authenticated Lamport public key to the server.

In turn, the server checks the authenticity of the proxy's one-time Lamport public key, verifies the integrity of the received message and eventually decrypts x_i.

After having received a sufficient number of x_i fragments, the server becomes able to reconstitute the original message and obtain the secret value *x*. It then generates a secret key *y* to be provided to the client.

The secret value *y* must be encrypted and authenticated by the client. However this latter cannot decrypt and verify the authentication and the integrity of the message because of its resource constraints. For this purpose, we propose that the proxies support also the reception of the secret key y on behalf of the client in a cooperative manner. These nodes take charge of the computational load required to verify the received message from the server and then transmit it securely to the client. However the divulgation of the secret key y to the proxies would affect the security of our system. In order to preserve the secrecy of *y*, we propose to have it encrypted with the secret key *x* reassembled at the server. The *x*-encrypted secret key *y* is MACed with the secret *x* and signed with the server's private key. It is finally sent to each proxy P_i, which has to verify the integrity of the received packet from the server before decrypting it. Then the packet's content (that is, *y* encrypted and MACed with *x*) is securely transmitted to the client. As long as an appropriate number of the same packet is received from different proxies, the client ensures the validity of the transmitted message from the server. Consecutively, it checks the MAC in order to ensure that the server has obtained the same secret x and verify the message integrity. Once the client receives a valid message, it can obtain the transmitted secret value *y* in order to complete the set up of the session key.

3) Session Key Derivation

The output master key MK is calculated as follows:

$$MK: = H (A \mid B \mid N_A \mid N_B \mid x \mid y) \quad (1)$$

with H() being a one-way hash function and N_A, N_B being nonces exchanged between A and B during the initial protocol phase of security capabilities negotiation.

A final message authenticated with the computed session key completes the set up of the key establishment procedure. The message sent by the server B provides A with the list of the proxies' identifiers that participated to the exchange. Thanks to this list and the identifiers/identities bindings received previously from T, A deduces which proxies did not participate to the collaborative process. This information can be used to refine the trust model for a better selection of supporting nodes in the future.

The phases of the proposed solution that correspond to secret key exchange and session key derivation are depicted in Figure 2.

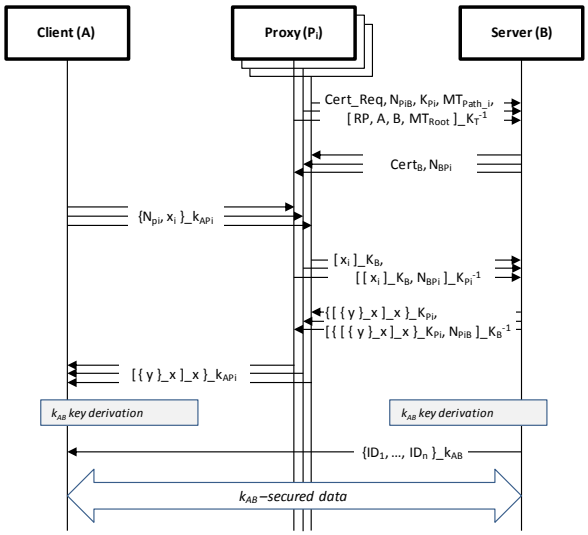

Figure 2. Secret key exchange and session key derivation.

IV. VALIDATION: SECURITY AND PERFORMANCE ANALYSIS

A. Security Analysis

Our protocol is based on a distributed exchange of random generated secrets. Cryptographically secure pseudorandom number generators [14] are therefore required at both the client and the server in order to ensure an adequate level of security. The corresponding constraint in computing power can be assumed at the client-side, since the generation of a pseudorandom number amounts to a symmetric encryption operation.

Only the client and the server are involved in the final secret key derivation, while assisting nodes support the secure delivery of generated secrets between the two peers. The protocol is designed so as to ensure the required security services of authentication, confidentiality and data integrity and to counter various denial-of-service and man-in-the-middle (MitM) attacks during the packets exchange.

In order to validate our proposed key establishment scheme, the first point of focus was to prove the security of each independent simple exchange A→Pi→B / B→Pi→A. This was formally achieved as detailed in next subsection. The second point of focus consisted in an analysis of the proxy-based scheme itself; this analysis is provided in the second subsection.

1) Formal Validation with AVISPA

A formal security analysis using the AVISPA [16] tool was carried out for proving the fulfillment of the desired security goals of the protocol. AVISPA (Automated Validation of Internet Security Protocol and Applications) is a push button security protocol analyzer based on formal methods,

performing analytical rules to illustrate whether the candidate protocol is safe or not. If vulnerability is detected, verification results revolve the attack trace, showing at which step and under which conditions an attack was made possible. The tool implements the Dolev-Yao intruder [17] able to eavesdrop, intercept messages, insert bogus data, or modify traffic passing through. AVISPA incorporates four different automatic protocol analysis techniques for protocol falsification on-the-fly model-checker (OFMC), constraint-logic based attack searcher (CL-AtSe), SAT-based model checker (SATMC), and tree automata based on automatic approximations for the analysis of security protocols (TA4SP) and furnishes a large library of well known Internet security protocols.

The first step of the protocol verification consists of modeling it using HLPSL formal language of AVISPA. The specification language HLPSL is used to describe the security protocol as sequences of exchanged messages between different parties and to express desired properties and security goals. The HLPSL specification is later translated into an IF specification providing a low-level description of the protocol and given as an input to the four automatic analysis back-ends of the AVISPA tool. Then the verification of the protocol's security properties, namely authentication, integrity, anti-replay and secrecy, starts. If a specified security property is violated, the back-ends return a trace explaining the sequence of actions that gave rise to the attack and exhibits which goal is violated.

First we have specified the actions of each participant in a module, which is called a basic role; the role of the constrained client in the protocol is modeled as follows:

```
role peer(A, B, Pi, T : agent,
      Kat, Kapi, Kpit: symmetric_key,
      SND_BA, RCV_BA, SND_PiA, RCV_PiA,
      SND_TA, RCV_TA: channel (dy))
played_by A
def=
```

The RCV and SND parameters indicate the channels upon which the participant playing role peer will communicate with other roles. Here A communicates with B and P_i both sending and receiving packets.

After defining basic roles, we have defined composed roles which describe the whole session of our protocol by the execution of all basic roles simultaneously.

```
role session(
      )
Def=
local
composition
          peer()
     /\   trustparty()
     /\   proxy()
     /\   server()
```

Finally, a top-level role was defined including the intruder activity trying to play some roles as a legitimate user.

```
role environment()

def=
      const
      ()
      intruder_knowledge = {  a, b, pi, t, kt, kb, ks,
                              ki,kipi,kai,kti,inv(ki),
                              {i.ki}_(inv(ks)) }

      composition
          session(a,b,pi,t,h,keygen,kat,kapi,kpit,kt,kb,ks)
      /\  session(a,b,i,t,h,keygen,kat,kai,kti,kt,kb,ks)
      /\  session(a,i,pi,t,h,keygen,kat,kapi,kpit,kt,ki,ks)
      /\  session(i,b,pi,t,h,keygen,kti,kipi,kpit,kt,kb,ks)
```

In the above extract, one can notice that the intruder may have had its public key (i.ki) signed by the same certificate authority that authenticates B, as represented by its knowledge of an {i.ki}_(inv(ks)) statement. Another noticeable point is the variation of the roles that the intruder i may assume in the protocol test, as shown in the last three lines: i is successively described as being able to act as P_i, A and B.

The security goals were finally specified in a goal section asserting that the secrecy should be achieved for the final master key (MS) between the client A and the server B, and for the Lamport private key between the sensor network trusted party T and each proxy.

The secrecy of a parameter was also declared before in the role section of the agent who has generated it. For example, after the generation of the Lamport key material in the role of the trusted party, we have further described the transition (exchanges) with the following secret facts

```
/\ Kpi' := new() %material key generation

/\ secret (inv(Kpi'),k,{T,Pi})
```

This means that the trusted party T declares that the generated Lamport private key Kpi is kept secret between T and P_i only and that this security objective is to be referred to as 'k'.

In a second part of the goal section, we asserted that authentication should be verified between each proxy and the server in order to prove that the node is legitimate and authorized to act on behalf of the constrained node and that the proxy communicates with the desired entity.

Eventually, these three goals (goal k and mutual authentication between proxy and server) translate to:

```
goal
  secrecy_of k,ms
  authentication_on server_proxy
  authentication_on proxy_server
```

Goal facts related to the mutual authentication between the proxy and the server are used at the role proxy and role server sections. The goal fact *witness* is used by the role to be authenticated in order to express that he wants to be the peer of the other role and will prove later its legitimacy. The goal fact *request* preceded by an accompanying witness is used by the authenticating role releasing in the transition after which the authentication is verified and is considered successful.

In our protocol we have used *witness* and *request* for the mutual authentication between the proxy and the server:

• proxy authenticates server on the value of Npb (because server sends back the received fresh nonce Npb signed with its private key). This translates as:

/\ witness(B,Pi,proxy_server,Npb') (at the role server)
/\ request(Pi,B,proxy_server,Npb) (at the role proxy)

• server authenticates proxy on the value of Nbp (because proxy sends back the received fresh nonce Nbp signed with its Lamport private key). This translates as:

/\ witness(Pi,B,server_proxy,Nbp') (at the role proxy)
/\ request(B,Pi,server_proxy,Nbp) (at the role server)

Subsequently, we checked the correctness of the implemented HLPSL code and of the protocol state machine by the use of the protocol animation tool called SPAN [18].

Finally, the security of the protocol was evaluated by executing the four AVISPA back ends (OFMC, SATMC, CL-AtSe and TA4SP) against our defined intended security goals. Peer authentication, secrecy, message integrity, delivery proof, identity proof and replay protection were evaluated. AVISPA tool produced a formal report as an output indicating that the protocol is "SAFE" against OFMC, CL-AtSe, and SATMC and "INCONCLUSIVE" against TA4SP database. No vulnerabilities were detected: according to the tool, it is not possible for an intruder to obtain a legitimate access to violate a security requirement and alter the successful protocol run, based on the specified security goals and the described assumptions.

2) Analysis of the Proxy-based Scheme

The formal security analysis described above and performed using the AVISPA tool consisted in validating the security of the simplex delivery of a secret fragment between the two peers A and B through a proxy Pi. However, other threats may menace the proper operation of the system due to its distributed nature. At each proxy, a cleartext part of the secret is processed. We assume that proxies can collude by sharing different secret parts delivered by the client, in order to reassemble them and reconstitute the secret key. AVISPA tool is unable to reason about this collusion attack. Hence, further security considerations were needed for the design of our protocol.

The proposed technique to counter this collusion attack is to base the selection of supporting proxies on a trust model. This model relies on a system that tracks nodes behavior in the network and takes into account previous experience, reputation and actual resources capabilities to eventually select appropriate nodes and revoke malicious ones from the cooperative delivery of the secrets *x* and *y*. The trust model should be also applied at the proxies' side in order to select honest participants with which they will accept to interact, in order to protect the proxies against exhaustion attacks – which AVISPA is not designed to detect either.

Another important point of focus in this security analysis is the unfair playing. Without need to collude with other nodes to disturb the smooth running of the collaborative process, a single proxy can refuse to process its part of the secret key or send instead bogus traffic to the server. Without an adapted protection scheme, a selfish proxy could paralyze the whole system and make the key exchange between the client and the server fail. This kind of "unfair" play has been carefully considered in the design of our solution. We have elaborated both prevention and reaction techniques to overcome this attack.

Recovery is ensured through the use of error correction scheme. Adding redundancy to each fragment sent by a proxy makes the rebuild of the secret key possible even if some malicious proxies refuse to cooperate during the secret key delivery process. Thus, it ensures resiliency of our solution to this type of attack. Prevention technique is handled by the server, which identifies the cooperative proxies when reassembling the secret key and reports a feedback to the client containing the list of participating proxies' identifiers. Thereby, the client deduces the identities of misbehaving proxies and will prevent their selection in the future.

B. Performance Analysis

In this section, we analyze our collaborative key establishment scheme in terms of computation and communication energy overhead. We compare it with a non-collaborative simple scheme, based on a two-pass key transport wherein secret values *x* and *y* are public key –encrypted and then signed [19].

Let consider the notations used as follows (1) C_s, the cost of performing a symmetric operation (encryption or decryption with a symmetric key);(2) C_{sig}, the cost for performing an asymmetric private operation (plaintext decryption or signature using a private key); (3) C_{ver} the cost for performing an asymmetric public operation (plaintext encryption or signature verification using a public key); (4) C_h the cost of keyed hash function (MAC); (5) C_{ij}, the cost of a transmitted message from the entity *i* to the entity *j*; (6) C_{ji}, the cost of a received message from the entity *j* to the entity *i*;(7) C_r the cost for performing the error redundancy scheme.

TABLE I. TABLE TYPE STYLES

Table Head	**Solution based on Asymmetric Cryptography**		**Distributed Approach**		
	A	***B***	***A***	***Pi***	***B***
Cryptographic operation	$2*C_{sig}+2*C_{ver}$	$2*C_{sig}+2*C_{ver}$	$(3+n^a+m^b)*C_s+m*C_h+C_r$	$2*C_s+2*C_{sig}+2*C_{ver}$	$C_s+C_h+2n*C_{sig}+2n*C_{ver}$
Message Transmission	$2*C_{AB}$	$2*C_{BA}$	$C_{AB}+2*C_{AT}+n*C_{APi}$	$C_{PiT}+2*C_{PiB}+C_{PiA}$	$2n*C_{BPi}+C_{BA}$
Message Reception	$2*C_{BA}$	$2*C_{AB}$	$C_{BA}+2*C_{TA}+m*C_{PiA}$	$C_{TPi}+2*C_{BPi}+C_{APi}$	$2n*C_{PiB}+C_{AB}$

a. *n* proxies are selected to assist the key exchange between two peers
b. *m* is a sufficient number of messages needed by A to learn y

As shown in the table, in the simple scheme the constrained node A has to perform two public key operations (encrypt and sign) to push its secret *x* and two other public key operations (verify and decrypt) to receive the secret *y*. The total computation and communication cost at the client side is:

$$C_{Simple} = 2*(C_{sig}+C_{ver})+2*(C_{AB}+C_{BA}) \quad (2)$$

Considering our proposed solution, the computational load of these public key operations is delegated to proxies at neighborhood while the constrained node only needs to perform few symmetric operations and to exchange more messages during the protocol exchange. The total computation and communication cost at the client side is:

$$C_{Distr.} = (3+n+m)*C_s+m*C_h+C_r+C_{AB}+2*C_{AT}+ n*.C_{APi}+C_{BA}+2*C_{TA}+m*C_{PiA} \quad (3)$$

It has to be noted that the messages sent to the proxies P_i can be grouped within one single packet (when compatible with the number of proxies, their location and the size of the messages to be sent) so as to decrease the cost of the transmission required to deliver these messages to all *n* proxies.

Yet the energy cost of a symmetric operation or a message reception/ transmission is 1000 times faster than an asymmetric operation [20]. Hence, according to the above evaluation, we prove that our proposed scheme is efficient and fit for the envisioned heterogeneous M2M communication model.

V. CONCLUSION

In this paper, a novel cooperative key establishment scheme has been proposed for heterogeneous M2M communications. The proposed scheme allows a highly resource-constrained node to set up a shared secret key with a remote server using asymmetric cryptography primitives. The approach does not require the constrained node to perform heavy asymmetric operations. Instead, assisting nodes at neighborhood take charge of this computational load on a distributed and cooperative basis. A formal security analysis using AVISPA has demonstrated that our proposed protocol is safe and fulfills required security goals. Further, cost overhead evaluation has proved that the protocol is efficient and reduces significantly resources consumption at the constrained client.

REFERENCES

[1] Y. Ben saied, A. Olivereau and D. Zeghlache," Energy Efficiency in M2M Networks: A Cooperative Key Establishment System", 3rd Int. Congress on Ultra Modern Telecommunications and Control Systems (ICUMT), 2011.

[2] H. Chan, A. Perrig and D. Song,"Key Distribution Techniques for Sensor Networks," *Wireless Sensor Networks*, T. Znati et al., eds., 2004.

[3] P. Szczechowiak, L. B. Oliveira, M. Scott, M. Collier, and R. Dahab, "NanoECC: testing the limits of elliptic curve cryptography in sensor networks," Proceedings of the 7th international Conference on Information Processing in Sensor Networks (IPSN '08), pp. 305–320, 2008.

[4] H. Wang and Q. Li, "Efficient implementation of public key cryptosystems on mote sensors," in Proceedings of the International Conference on Information and Communication Security (ICICS '06), pp. 519–528, December 2006.

[5] R. Watro, D. Kong, S. Cuti, C. Gardiner, C. Lynn, P. Kruss, "TinyPK: Securing sensor networks with public key technology," in Proceedings of the 2nd ACM workshop

[6] W. Hu, P. Corke , W. C. Shih, L. Overs, "secFleck: A Public Key Technology Platform for Wireless Sensor Networks," in Proceedings of the 6th European Conference on Wireless Sensor Networks, February 11-13, 2009, Cork, Ireland.

[7] P. Pecho, J. Nagy, and P. Hanáček, "Power consumption of hardware cryptography platform for wireless sensor," in Proceedings of the International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT '09), pp. 318–323, December 2009.

[8] G. Murphy, A. Keeshan, R. Agarwal, and E. Popovici, "Hardware—software implementation of public-key cryptography for wireless sensor networks," in IET Irish Signals and Systems Conference, pp. 463–468, June 2006.

[9] R. Riaz, A. Naureen, A. Akram, A. Akbar, K. Kim, and H. Farooq Ahmed, "A unified security framework with three key management schemes for wireless sensor networks," Computer Communications, vol. 31, no. 18, pp. 4269–4280, 2008.

[10] J. Mache, C.-Y. Wan, and M. Yarvis, "Exploiting heterogeneity for sensor network security," in Proceedings of IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, June 2008, pp. 591–593.

[11] M. Petrova et al, " Overall secure PN architecture," in My Personal Adaptive Global NET (MAGNET), D2.1.2/D4.1.3,Octobre 2005

[12] L. Lamport, "Constructing digital signatures from one-way function," in Technical Report SRI-CLS-98, SRI international (October 1979).

[13] M. Jakobsson, T. Leighton, S. Micali, and M. Szydlo, "Fractal Merkle Tree Representation and Traversal," in RSA Cryptographers Track, 2003.

[14] B. Sklar. Reed-solomon codes, available at http://ptgmedia.pearsoncmg.com/images/art-sklar7-reed-solomon/elementLinks/art-sklar7-reed-solomon.pdf

[15] A. Francillon and C. Castelluccia "TinyRNG: A cryptographic random number generator for wireless sensors network nodes", 5th Int. Symp. Modeling and Optimization Mobile, Ad Hoc, and Wireless Networks(WiOpt), 2007.

[16] A. Armando et al, "The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications," in Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, Springer, Heidelberg (2005), http://www.avispa-project.org

[17] D. Dolev and A. C. Yao. On the security of. public key protocols. In Proc. 22th IEEE Symposium on Foundations of Computer Science, pages 350-357, 1981.

[18] Y. Glouche and T. Genet. "SPAN – a Security Protocol ANimator for AVISPA – User Manual," IRISA / Université de Rennes 1, 2006. 20 pages. http ://www.irisa.fr/lande/genet/span/.

[19] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1997, pp. 509–510.

[20] N.R. Potlapally, S. Ravi, A. Raghunathan and N.K. Jha, A study of the energy consumption characteristics of cryptographic algorithms and security protocols. *IEEE Transactions on Mobile Computing* (2006), pp. 128–143.