Novel Lightweight Signcryption-based Key Distribution Mechanisms for MIKEY

Kim Thuat Nguyen[1], Nouha Oualha[1], and Maryline Laurent[2]

[1] CEA, LIST, Communicating Systems Laboratory,
91191 Gif-sur-Yvette CEDEX, France
kimthuat.nguyen@cea.fr, nouha.oualha@cea.fr
[2] Institut Mines-Telecom, Telecom SudParis, UMR CNRS 5157 SAMOVAR,
9 rue Charles Fourier, 91011 Evry, France
maryline.laurent@telecom-sudparis.eu

Abstract. MIKEY is a standard key management protocol, used to set up common secrets between multiple parties for multiple scenarios of communications. As MIKEY becomes widely deployed, it becomes worthwhile to not confine its applications to real-time or other specific applications, but also to extend the standard to other scenarios as well. For instance, MIKEY can be used to secure key establishment in the Internet of Things. In this particular context, Elliptic Curve Cryptography-based (ECC) algorithms seem to be good candidate to be employed by MIKEY, since they can support equivalent security level when compared with other recommended cryptographic algorithms like RSA, and at the same time requiring smaller key sizes and offering better performance.
In this work, we propose novel lightweight ECC-based key distribution extensions for MIKEY that are built upon a previously proposed certificateless signcryption scheme. To our knowledge, these extensions are the first ECC-based MIKEY extensions that employ signcryption schemes. Our proposed extensions benefit from the lightness of the signcryption scheme, while being discharged from the burden of the public key infrastructure (PKI) thanks to its certificateless feature. To demonstrate their performance, we implemented our proposed extensions in the Openmote sensor platform and conducted a thorough performance assessment by measuring the energy consumption and execution time of each operation in the key establishment procedure. The experimental results prove that our new MIKEY extensions are perfectly suited for resource-constrained devices.

1 Introduction

Multimedia Internet KEYing (MIKEY) [4] is a key management protocol which is intended for use with real-time applications. MIKEY provides different methods to establish a session key with multiple parties, in addition to the authentication of parties if required. For example, MIKEY pre-shared key method permits any two parties with a pre-shared secret to set up a secure communication. However, this mechanism suffers from scalability issues since it is unpractical

to pre-distribute a common key for any two parties in large networks, e.g. the Internet of Things (IoT). To be scalable, public key encryption-based methods, where any two parties can establish security communications without any *a priori* shared common keys, have been proposed to be employed by MIKEY.

These different key distribution mechanisms can be classified into two categories: (i) a key exchange mode and (ii) a key transport mode. The MIKEY key exchange modes, such as, MIKEY-DHSIGN [5], MIKEY-DHHMAC [13], are usually based on the Diffie-Hellman (DH) key exchange [20]. These modes provide the perfect forward secrecy property, i.e. the compromise of long-term keying materials does not reveal previously derived session keys. Additionally, both communicating parties participate in the session key generation process. As a result, DH-based modes require at least two message exchanges to set up a common secret key. As another disadvantage, these modes do not support the establishment of group keys.

In key transport modes, on the other hand, the initiating party is responsible for the key generation. The generated keys are then encrypted using the public key of the responding party. Even if key transport modes do not provide perfect forward secrecy, they are more efficient in terms of computation and communication than DH-based modes. Indeed, only a half roundtrip is needed to set up a common key between two parties. Existing key transport modes of MIKEY generally employ a public key encryption algorithm to protect transferred keys, such as RSA [18] or ECIES [15] and an additionally public key signature algorithm to sign MIKEY message. In this paper, we propose to use more lightweight key transport modes built upon a signcryption scheme defined in [21], which is an authenticated encryption algorithm that combines encryption and signature procedures in an optimized manner. The signcryption scheme is based on Elliptic Curve Cryptography (ECC), thus inheriting multiple advantages of ECC in terms of performance. As mentioned in [15], ECC-based schemes require smaller key sizes and offer better security per bit, when compared with known cryptographic algorithms like RSA. Moreover, the signcryption scheme in [21] offer the certificateless feature that allows to dispense the two parties with the provision of a digital certificate issued by a Public Key Infrastructure (PKI).

Our contribution: In this paper, we first introduce two novel key transport mechanisms for the standardized key management protocol MIKEY [5]. The main idea is to apply the certificateless elliptic curve Korean signature-based signcryption scheme, namely ECKSS, defined in [21] as the public key encryption algorithm to construct MIKEY messages. Then, we present experimental performance results of the two proposed key distribution methods by measuring the energy consumption and the execution time for each operation. Our solutions have been implemented and validated using the Openmote sensor platform [2]. The experimental results show that our proposed extentions to MIKEY are suited for resource-constrained devices.

Paper outline: The rest of this paper is organized as follows. Section 2 surveys several existing ECC-based key transport solutions proposed for MIKEY and presents briefly related work on signcryption schemes. Section 3 provides

several notations and recalls our proposed signcryption scheme provided in [21]. We describe in details our proposed key transport mechanisms for MIKEY in section 4. Section 5 discusses several security considerations needed for our proposals. The performance assessment of our proposals is given in section 6, while section 7 concludes our work.

2 Related work

As MIKEY [5] becomes more deployed, extensions to the base protocol have emerged [15], [4], [6]. Several of these extensions brought additional key distributions methods to MIKEY, for instance based on Elliptic Curve Cryptography (ECC) [23]. Since ECC support requires smaller keys while keeping the same security level as other asymmetric algorithms like RSA, ECC usage is considered interesting for devices with limited performance and storage capabilities. ECC extensions to MIKEY offer new mechanisms for authentication, encryption and digital signature to provide secure key distribution. ECC-based mechanisms such as ECDH to extend the Diff-Hellman exchange [20], ECDSA or ECGDSA for digital signatures, Elliptic Curve Integrated Encryption Scheme (ECIES) and Elliptic Curve Menezes-Qu-Vanstone Scheme (ECMQV) to provide, respectively, integrated encryption and authenticated key exchange, have been defined in [23]. To the best of our knowledge, ECC-based signcryption mechanisms have not been proposed for MIKEY, even though these mechanisms have been present in the literature for many years, and many ECC-based signcryption mechanisms offer a good performance thanks to their optimized authenticated public key encryption besides the advantages of ECC.

Signcryption schemes allow to simultaneously perform the functions of both digital signature and encryption. Most of existing signcryption schemes are derived from popular signature schemes (refer to the survey in [27]). For examples, Zheng's scheme [28] is based on ElGamal encryption and signature schemes [12], and Shin et al.'s scheme [24], called SCDSA+, is based on DSA (Digital Signature Algorithm) signature scheme [14]. Zheng's scheme requires complex interactive zero-knowledge proof to validate the non-repudiation and does not provide insider confidentiality. On the other hand, the security of Shin et al.'s scheme has not been formally proven. KCDSA (Korean Certificate-based Digital Signature Algorithm) [19] is a variant of DSA, whose design allows to relieve the signature generation and verification procedures of modular inversions required in DSA. Two signcryption variants based on KCDSA have been proposed by Yum et al. in [26]. However, the first variant is confidentiality insecure in the insider model, while the second one is not semantically secure due to the disclosure of the hash of the message.

ECC-based signcryption schemes have also been proposed in several papers like [17] and [25], which are both based on ECDSA. In [21], we have proposed a new signcryption scheme based on ECKCDSA, and we have formally proven the security of this scheme in the random oracle model, thus providing outsider/insider confidentiality and unforgeability, in addition to non-repudiation,

while being more efficient in terms of communication and computation costs than existing signcryption schemes. Moreover, our scheme offers the certificateless feature, so certificates are not needed to verify initiator/responder's public keys. In this paper, we propose to extend the MIKEY base protocol with new key distribution methods based on our signcryption scheme [21], and we demonstrate the advantages and gains in terms of performance achieved by these methods.

3 Preliminaries

In this section, we introduce several notations and review briefly our elliptic curve-based signcryption scheme proposed in [21].

3.1 Abbreviations and Notations

The definitions and abbreviations, as described in Table 1, and used throughout the rest of this document are consistent with those used in the MIKEY standard [5].

$P + Q$	Addition of two elliptic curve points P and Q.
$t.P$	Addition of P with itself t times.
$s\|\|t$	Concatenation of two strings s and t.
$\perp$	Error symbol.
KMS	Key Management Server
I	Initiator
R	Responder
HMAC	Hash Message Authentication Code
MAC	Message Authentication Code
TEK	Trafic-Encrypting Key
TGK	TEK Generation Key

Table 1. Abbreviations

3.2 The certificateless elliptic curve Korean signature-based signcryption

In this subsection, we describe our certificateless signcryption scheme based on elliptic curve, named as ECKSS. The security of this scheme has been formally proved in the random oracle model [21].

Considered actors We consider three main actors in our scenario presented in the following.

– Two parties: an Initiator (I) and a Responder (R), which respectively initiates the communication and responds to incoming requests.
– A trusted Key Management Server (KMS), which is responsible for generating keying materials and that acts as the root of trust of the responder and initiator. The proposed solutions support also multiple KMSs i.e., the initiator or the responder may use a different KMS, but this multi-authority setting is considered as out of scope of the paper.

Security parameter generation process Depending on the security parameter as input, KMS first runs the Setup algorithm to define an elliptic curve E over finite field $\mathbb{F}_p$ with a generator G, where p is the prime modulus. Two hash functions are also defined: $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p$ and $H_2 : \{0,1\}^* \rightarrow \{0,1\}^*$. ($\mathsf{Enc}$, Dec) are the encryption and decryption algorithms of a symmetric cipher. Then, KMS executes the KeyGen algorithm to generate the keying material for I and R. KMS first chooses the master key as mk from $\mathbb{Z}_p$. Its public key is then calculated as $PK_{KMS} = mk.G$. For an entity A with the identifier id_A, KMS generates the public and private values of an entity A as follows.

– Compute $V_A = x_A.G$, where x_A is a random number on $\mathbb{Z}_p$
– Compute the private key for A: $priv_A = (mk + x_A.H_1(id_A||V_A||G||PK_{KMS}))^{-1}$
– Compute $P_A = priv_A^{-1}.G$
– Set the public key of A as $PK_A = (P_A, V_A)$

As we shall see, we can validate the public key of A by using the following equation:

$$P_A = PK_{KMS} + H_1(id_A||V_A||G||PK_{KMS}).V_A \tag{1}$$

ECKSS description The detailed procedure of ECKSS is described in the following:
– $\mathsf{Signcrypt}(priv_I, PK_I, PK_R, M) \rightarrow CT$: To signcrypt a message M intended to R, I executes the following steps:

1. Choose randomly $x \leftarrow \mathbb{Z}_p$
2. Compute $K = x.PK_R$
3. Generate a secret key: $\tau = H_2(PK_I||PK_R||K)$
4. Compute $r = H_1(PK_I||PK_R||K||M)$
5. Compute $s = priv_I.(x - r)$
6. Compute $c = \mathsf{Enc}_\tau(M)$
7. Send the ciphertext $CT = (r, s, c)$ to R

– $\mathsf{Unsigncrypt}(priv_R, PK_R, PK_I, CT) \rightarrow M$: Upon receiving the ciphertext $CT = (r, s, c)$ from I, R has to perform the following procedure:

1. Compute $K = (s.priv_R^{-1}).PK_I + (r.priv_R^{-1}).G$.
2. Get the secret keys: $\tau = H_2(PK_I||PK_R||K)$
3. Compute $\mathsf{Dec}_\tau(c) = M$

4. Verify that $r = H_1(PK_I||PK_R||K||M)$

Note that I and R can be sure about the public values of the other party by verifying the equation (1). This feature makes ECKSS *certificateless* since it does not require certificates to authenticate the public keys.

4 Novel signcryption-based key distribution methods for MIKEY

In this section, we first present the payload and data type formats of a MIKEY key transport mechanism. Then, we clarify our design goals for proposing new key distribution methods for MIKEY. Finally, we give details on these extensions in respect to the original MIKEY payload formats.

4.1 Basic payload and message formats of a MIKEY key transport mechanism

Figure 1 describes the basic message composition of a MIKEY key transport method that uses a public-key encryption algorithm, for example, in the MIKEY-RSA [5] and MIKEY-ECIES [15] modes. The mechanisms contain two message exchanges: the I_MESSAGE and the R_MESSAGE. The main objective of the Initiator's message is to distribute one or more TGKs and a set of security parameters in a secure manner. We recall the payload notions as defined in [5], in the following:

- HDR: The MIKEY header, which contains related data and information mapping to the specific security protocol used.
- T: The timestamp, used to prevent replay attacks.
- RAND: The random byte-string, which is used as a value for the freshness of the session key generation process.
- IDx: The identity of the entity X (IDi: Identity of the Initiator, IDr: Identity of the Responder).
- SP: The security policies.
- KEMAC: The Key Data Transport Payload, which contains the encrypted TGKs and a MAC.
- CHASH: The Cert Hash Payload, which is the hashes of the used certificates (e.g. CERTi).
- PKE: The Envelope Data Payload, which contains the encrypted envelope key, env_key.
- SIGNx: The signature covering the entire MIKEY message, which is generated using the signature key of the entity X.
- V: The verification message payload containing the MAC calculated over the entire MIKEY message.

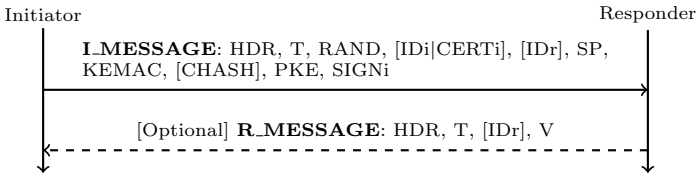

Fig. 1. Basic message format for a MIKEY public key encryption method

As described in Figure 1, the MIKEY public key encryption method first chooses an envelope key env_key. This key is then to be encrypted using the public key PK_R of the Responder and conveyed in the PKE payload: PKE $= E(PK_R, \mathsf{env\_key})$. Then, the encr_key and the auth_key are derived from the envelope key, env_key. These two keys are used to build the KEMAC payload where the encr_key is used to encrypt the TGKs. The encrypted part is then followed by a MAC calculated using auth_key over the entire KMAC payload. The whole MIKEY message is then integrity protected by the signature payload SIGNi.

4.2 Design motivations

The novel key transport mechanisms for MIKEY are designed to put forwards the following motivations:

- **Performance and Efficiency**: Our proposed ECKSS signcryption scheme is able to transport secret data in a secure manner without intensive calculation. Thus, ECKSS-based methods for MIKEY is able to address the same scenario as the other key establishment methods in MIKEY [15]. In fact, existing MIKEY modes are intended for application-layer key management and multimedia applications. However, thanks to ECKSS lightweight computation requirements, the proposed methods can be considered in constrained environments such as IoT. We prove the feasibility of our proposed mechanisms in such environment in section 6. Furthermore, the mechanisms are based on elliptic curve cryptography (ECC). Additionally, when compared with existing ECC-based asymmetric methods of MIKEY, our proposed mechanisms are the most efficient while offering equivalent security guarantees. More details are provided in section 6.1.
- **PKI Independence**: ECKSS can be applied in the context where no access to a public-key infrastructure (PKI) is available. Indeed, the validation of entity's public keys is realized in equation (1) without certificates. Moreover, as pre-shared master secrets are not required, the proposed ECKSS-based schemes should be as scalable as other existing asymmetric mechanisms of MIKEY.

4.3 The MIKEY-ECKSS mode specification

Figure 2 defines the message exchanges for our first proposal MIKEY-ECKSS. Similarly to other MIKEY public key encryption methods such as MIKEY-RSA [5] or MIKEY-ECIES [15], the main objective of the Initiator's message is to distribute one or more TGKs in a secure manner. This method reuses the defined payload in section 4.1 except the payload ECKSSi in the I_MESSAGE. This payload transports actually the encrypted TGKs through the triple (r, s, c) as defined in section 3.2. To guarantee the integrity protection, we employ the payload SIGNi, which is a signature covering the entire I_MESSAGE. As described in [5], the SIGNi payload will use either ECDSA or ECGDSA as the signature algorithm.

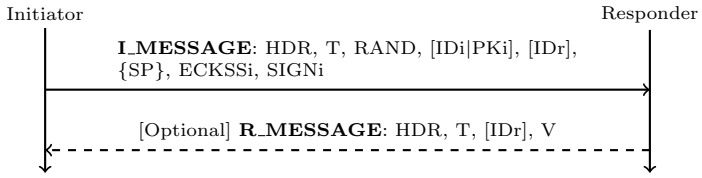

Fig. 2. Elliptic curve Korean signature-based signcryption key distribution method for MIKEY

Upon receiving the I_MESSAGE, R first approves the integrity of the message by verifying the appended signature payload SIGNi. If the verification holds, it then uses the Unsigncrypt algorithm to decrypt the payload ECKSSi in order to obtain the values of TGKs. In case mutual authentication is required, the verification message, V, is calculated by building a MAC over the concatenation of the header HDR (the same as the one that was used in the I_MESSAGE), the timestamps, the identities of the two parties, using the authentication key. The latter is derived from the received TGKs. Then, we append the V payload to the concatenation (HDR, T, [IDi, PKi], [IDr]) to form the R_MESSAGE. However, as depicted in Figure 2, the R_MESSAGE is optional.

4.4 MIKEY-ECKSS-HMAC mode specification

In this subsection, we describe in detail our second key distribution extension for MIKEY. We call this mode MIKEY-ECKSS-HMAC, since this mode uses ECKSS to envelop the TGKs and HMAC to ensure the authentication of the messages exchanged. As we shall see, the use of the signature payload SIGNx still requires multiple exponentiations in the signature generation and verification processes, e.g. 3 modular exponentiations are needed if using ECDSA. As a result, MIKEY-ECKSS-HMAC is even more lightweight than MIKEY-ECKSS, which is suitable for constrained devices.

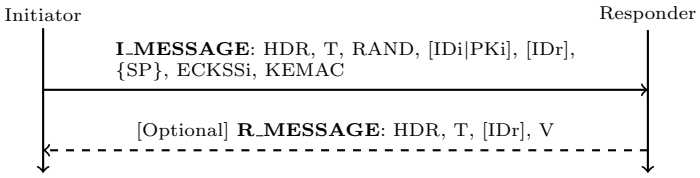

Fig. 3. HMAC-authenticated Elliptic Curve Korean signature-based signcryption key distribution method for MIKEY

Figure 3 describes in detail the MIKEY-ECKSS-HMAC mode. We use the same notations for the payload names as defined in section 4.1. In the I_MESSAGE, the ECKSSi payload contains the triple (r, s, c) as depicted in section 3.2. The KEMAC payload conveys the Hash Message Authentication Code (HMAC) of the entire MIKEY message. This technique is also employed in the MIKEY-DHHMAC method [13]. The HMAC value is calculated using the secret key k_{auth}. This key is generated during the encryption of TGKs using the ECKSS algorithm. Indeed, we make modifications in step 4 of the Signcrypt algorithm and in step 2 of the Unsigncrypt algorithm (see section 3.2 for more details). Another secret key k_{auth} in addition to τ is generated, as depicted in Table 2. This key is to be used in the creation of HMAC.

Signcrypt	Unsigncrypt
3) Generate a couple of secret keys: $\tau, k_{auth} = H_0(PK_S\|\|PK_R\|\|K)$	2) Get the secret keys: $\tau, k_{auth} = H_2(PK_I\|\|PK_R\|\|K)$

Table 2. Modifications made to the Signcrypt and Unsigncrypt algorithms

Upon receiving the I_MESSAGE, R first runs the Unsigncrypt algorithm to get the value of TGKs and k_{auth}. The authentication key k_{auth} is then used to verify that the I_MESSAGE has not been modified by an attacker. Indeed, a MAC is calculated over the entire I_MESSAGE using k_{auth}. This value is then compared with the hashed value extracted from the KEMAC payload. On the other hand, the R_MESSAGE's construction is optional as depicted in Figure 3. This message is only needed when mutual authentication between parties is required.

5 Security considerations

As this paper proposes two new methods for the MIKEY protocol, existing security considerations discussed in [5] apply here as well.

As mentioned in [5], one should select a secure symmetric cipher supporting at least 128 bits as a practical choice for the actual media protection. In our proposals, the payload ECKSSi carries the actual encrypted TGKs, the used encryption algorithm should also offer a security level of at least 128 bits. For the selection of hash functions, we recommend to work at least with SHA-256 [8] when constructing the ECKSSi payload and other payloads as well. This should be seriously taken into account in order to achieve the 128-bit level.

Similar to other key transport mechanisms, our proposed methods do not provide the perfect forward secrecy property. A Diffie-Hellman key distribution resolves this issue but requires the transmission of the R_MESSAGE in order to set up a common secret key.

In order to provide the *certificateless* feature, our proposed methods rely on the binding of public values of communicating parties with the public keys issued by KMS. Thus, after validating a provided public value using equation (1), we can be sure that only KMS is able to generate such value. It also means that the KMS can read all traffic between any parties administrated by the KMS. However, we assumed that the KMS is a fully trusted party.

6 Performance Assessment

In this section, we first quantify the performance of our schemes. Then, we describe our testing environment and the used methodology to achieve the experimental measurements. Finally, we provide in detail the performance results in terms of energy consumption and the time execution of our proposals including the ECKSS algorithm and the two proposed MIKEY modes.

6.1 Comparison with related work

Table 3 illustrates the performances of our two proposed methods and multiple ECC-based MIKEY modes in related work. The table first identifies if the scheme is a key exchange method or a key transport method. Then, it shows if the scheme is independent to the public key infrastructure or not. This property means that a PKI-independent scheme does not require standard digital certificates to provide authentication between communicating parties and hence the scheme is discharged from complex operations during certificate verification, revocation and management processes. Then, the efficiency of each scheme is evaluated with respect to the computational cost demonstrated in terms of the number of expensive operations needed to generate the I_MESSAGE and the R_MESSAGE. Here, we consider the three most expensive operations for an ECC-based scheme: modular point multiplication (PM), modular inversion (I) and pairing operation (e). Furthermore, we provided also the name of the payload that requires these expensive operations. For example, in a PM column, the line "2 (PKE) + 1 (SIGNi)" means that two point multiplications are executed to build the PKE (public key encryption) payload and 1 other point multiplication is calculated to build the SIGNi (signature) payload. For simpler comparison, if

not explicitly specified in each mode, we assume that SIGNi payload carries an ECDSA signature and its related data.

Mode	Type	PKI	I_MESSAGE			R_MESSAGE		
			PM	I	e	PM	I	e
MIKEY-DHSIGN [15]	KE	Yes	1 (DH) + 1 (SIGNi)	1(SIGNi)	0	1 (DH) + 2 (SIGNi)	1 (SIGNi)	0
MIKEY-ECQMV [15]		Yes	1(ECCPT)+1 (SIGNi)	1 (SIGNi)	0	1 (PKE)+2 (SIGNi)	1 (SIGNi)	0
MIKEY-SAKKE [16]	KT	No	3 (SAKKE)+1 (SIGNi)	1 (SIGNi)	0	2 (SAKKE)+4 (SIGNi)	0	1 (SAKKE)
MIKEY-ECIES [15]		Yes	2 (PKE)+1 (SIGNi)	1 (SIGNi)	0	1 (PKE)+2 (SIGNi)	1 (SIGNi)	0
MIKEY-ECKSS		No	1 (ECKSSi) + 1 (SIGNi)	0	0	2 (ECKSSi) + 2 (SIGNi)	0	0
MIKEY-ECKSS-HMAC		No	1 (ECKSSi)	0	0	2 (ECKSSi)	0	0

Table 3. Performance comparison of our propositions and ECC-based MIKEY modes in related work

Meaning of abbreviations: PM: Modular point multiplication; I: Modular Inversion; e: Pairing operation; PKI: Public Key Infrastructure; KE: Key Exchange; KT: Key Transport.

As we shall see, the first two modes in Table 3: MIKEY-DHSIGN and MIKEY-ECQMV, are two ECC-based key exchange methods proposed for MIKEY. These methods are based on the Diffie-Hellman key exchange [20]. Hence, the R_MESSAGE is compulsory in order to setup a common secret key. On the other hand, in a key transport mechanism, I envelops and sends directly a secret key/message that can be used as a session key without the response from R.
As our proposed schemes are key transport mechanisms, we only make direct comparison with other key/message distribution mechanisms proposed for MIKEY. As depicted in Table 3, MIKEY-ECIES [15] seems to be our direct competitor in terms of performance since it is slightly more heavyweight than our first proposal MIKEY-ECKSS (with two more modular inversions to compute). In addition, MIKEY-ECKSS is more lightweight in the generation of the I_MESSAGE which can be beneficial for a very resource-constrained initiator. Our second proposal MIKEY-ECKSS-HMAC is even more efficient than our first one. As such, it requires only 1 point multiplication for generating the I_MESSAGE and 2 point multiplications for generating the R_MESSAGE. Furthermore, both proposals do not require certificates to validate the public values of communicating parties, which is not the case in MIKEY-ECIES mode. MIKEY-SAKKE [16] is also exempted from the use of PKI. However, this mode is much more expensive than our two methods since a pairing operation needs to be executed when receiving the R_MESSAGE.

6.2 Experimental tools and platforms

We implemented our assessment program in C for the operating system Contiki 3.0 [10]. Based on the Relic library version 0.4.0 [3], we evaluated our proposed MIKEY modes on the elliptic curves secg_k256. Its domain parameters have been recommended by SECG [22], which provides a security level of 128 bits. In addition, we opted for the sensor node Openmote to evaluate the required operations. Openmote [2] is a low power wireless sensor module featured with 32 MHz Cortex-M3 microcontroller, a CC2520-like radio transceiver, 32 kB of RAM,

512 kB of ROM and an IEEE 802.15.4 radio interface. This platform supports 32 bit addressing and sufficient RAM and ROM capacities. Such features are needed in order to use a cryptographic library along with an application on top of it.

In our testing scenario, we encrypted data using AES in CBC mode. For MAC and message verification function, we used SHA-256 as secure hash algorithm, which provides digests (hash values) that are 256 bits. Furthermore, we transported each time a TGK with the size of 32 bytes in our tests. In each case, the experimental measurements have been obtained by calculating the average of 100 executions for each operation.

6.3 Methodology

For measuring the processing time, we used two timers provided in the rtimer library of Contiki [10]. The first timer with 128 ticks per second was employed to measure the execution time of expensive operations. The second one is more powerful with 32768 ticks per second. It was used to measure the time duration of the mote running on a specific mode.

On the other hand, in order to assess the energy consumption, we employed a software-based online energy estimation mechanism described in [11]. In this model, we can calculate the energy consumption E in Joule of an operation on Contiki using the following equation:

$$E = U * \sum I_m * t_m \tag{2}$$

where U is the supply voltage, I_m is the hardware specific current draw, t_m is the time duration and m means a particular mode (e.g. active power mode, low power mode, transmit mode and receiver mode). In our scenario, the value of U is typically 3V, as with two new AA batteries. Besides, the current draw of the sensor node in each mode is extracted from its data sheet [1]. Concretely, we considered the following modes in our measurement: power mode 1 (cpu active mode), power mode 2 (low power mode), active-mode rx (receive mode) and active-mode tx (transmit mode). The consuming current draw for each mode are respectively: $I_{pm1} = 0.6mA$, $I_{pm2} = 1.3\mu A$, $I_{rx} = 20mA$ and $I_{tx} = 24mA$, as described in [1]. The time duration t_m that the mote is in mode m, is measured using Powertrace and Energest power profile [9]. These latters are pre-loaded tools in the Contiki OS, which provide an accuracy up to 94% of the energy consumption of a device.

6.4 Experimental results of ECKSS

In this subsection, we provide the experimental results of our signcryption scheme ECKSS. Table 4 shows the execution time and energy cost of ECKSS algorithms on the elliptic curve secg_k256 over the Openmote platform. The results reveal that even for a really lightweight signcrypt algorithm with only one point multiplication, it requires up to 2.6 s to compute and consumes 5.6 mJ. The resources

Algorithm	Time (s)	Energy (mJ)
ECKSS Signcrypt	2.64	5.6
ECKSS Unsigncrypt	5.8	12.4
Public Key Validation	3.12	6.6

Table 4. Energy consumption and time execution of ECKSS algorithms on the Openmote platform

required for an unsigncrypt are practically doubled since the algorithm has to compute 2 point multiplications. We provide also in Table 4 the resources consumed by an entity to validate other party's public values. As we shall see, this process consumed the same order of magnitude of time and energy as the signcrypt algorithm. Such property is advantageous since the verification of certificates in a PKI-based scheme is usually complex and energy and time consuming.

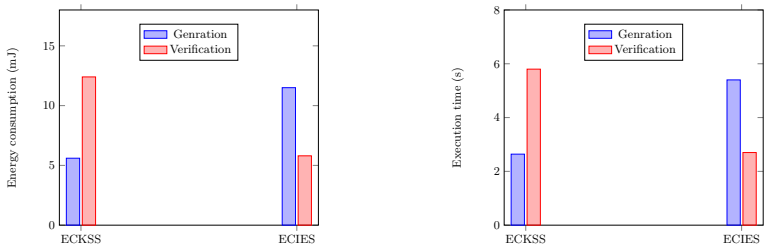

Fig. 4. Performance comparison of our proposal ECKSS with the algorithm ECIES

In Figure 4, we compare the performance of our ECKSS implementation with the standard algorithm ECIES, as specified in [7]. ECIES's implementation is provided in the Relic library [3]. We remark that the total work load required by our scheme ECKSS is practically identical to the one of ECIES. As we can see in Table 3, ECKSS is more rapid in the data encryption process but slower in the data decryption process.

6.5 Experimental results of the proposed MIKEY modes

In this subsection, we describe the performance of our prototype implementations for the two proposed MIKEY modes.

In MIKEY-ECKSS's implementation, we use ECDSA as the signature algorithm. Table 5 provides the performance of ECDSA algorithms on the Openmote platform. These experimental results are measured based on the implementation of ECDSA provided in [3]. As we can see, ECKSS is even slightly more efficient than ECDSA both in the generation and verification processes. This fact

Algorithm	Time(s)	Energy (mJ)
ECDSA signature generation	2.75	6.3
ECDSA signature verification	5.83	13.6

Table 5. Energy consumption and time execution of ECDSA algorithms on the Openmote platform

is understandable since our proposal is exempted from two modular inversions compared to the ECDSA algorithm.

Additionally, to be more adapted to resource-constrained devices, we replace the timestamps payload by an incremental counter payload. This counter is used together with the random number (carried in RAND payload) to mitigate the replay attacks. If it is the first time that I communicates with R, the counter is set to 0. It is increased by 1 after every successful key/data transportation.

Mode	I_MESSAGE		R_MESSAGE	
	Time (s)	Energy (mJ)	Time (s)	Energy (mJ)
MIKEY_ECKSS	5.6	11.9	11.4	24.3
MIKEY_ECKSS_HMAC	2.6	5.6	5.7	12.2

Table 6. Energy consumption and time execution of our proposed MIKEY modes on openmote

Table 6 demonstrates the average time and energy consumption for generating the I_MESSAGE and R_MESSAGE, respectively. The measures show that the MIKEY-ECKSS-HMAC mode is approximately two times more efficient than the MIKEY-ECKSS mode. The performance gap between them lies in the cost of creating the SIGNi payload. As such, instead of certificates and signatures, MIKEY-ECKSS-HMAC uses a keyed hash message authentication code (carried by the KEMAC payload) to guarantee the integrity of the messages exchanged.

Figure 5 provides a graphical view of the performance of our proposals in comparison with the MIKEY-ECIES mode. The performance of the latter are roughly estimated by summing the experimental results of the two algorithms ECIES and ECDSA given in Figure 4 and Table 5. As we shall see, the MIKEY-ECIES mode has a slightly higher computational cost in comparison with our proposed modes. However, it requires certificates to validate the public keys. This constraint could be very costly for a sensor node, since the verification of certificates is usually complex and consuming in time and energy.

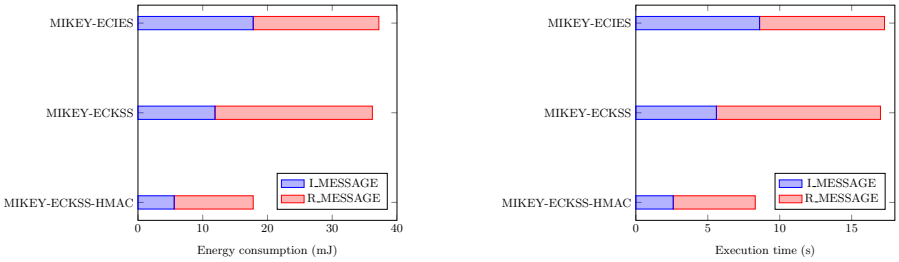

Fig. 5. Performance comparison of our proposed MIKEY modes and MIKEY-ECIES mode

7 Conclusion

In this paper, we proposed two novel signcryption-based key transport methods for MIKEY. Both methods are relieved from the dependance on a public key infrastructure thanks to their certificateless feature, and are more lightweight in terms of computation when compared with existing ECC-based key transport mechanisms proposed for MIKEY. The performance of the proposed methods have been demonstrated by experimental implementations on the Openmote sensor platform. The results confirmed that our proposed MIKEY extensions are feasible on resource-constrained devices. Hence, they can be used not only as key distribution mechanisms for real-time applications but also as lightweight key distribution solutions for the Internet of Things applications.

References

1. Cc2538 data sheet, `http://www.ti.com/lit/ds/symlink/cc2538.pdf`, last accessed: May 2016.
2. Openmote platform, `http://www.openmote.com/`, last accessed: May 2016.
3. Relic toolkit - an efficient library for cryptography, `https://github.com/relic-toolkit/relic`, last accessed: May 2016.
4. Mohammed Riyadh Abdmeziem and Djamel Tandjaoui. An end-to-end secure key management protocol for e-health applications. *Computers & Electrical Engineering*, 44:184–197, 2015.
5. Jari Arkko, Elisabetta Carrara, Fredrik Lindholm, Mats Naslund, and Karl Norrman. Rfc 3830: Mikey: Multimedia internet keying. *Internet Engineering*, 2004.
6. Aymen Boudguiga, Alexis Olivereau, and Nouha Oualha. Server assisted key establishment for wsn: A mikey-ticket approach. In *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pages 94–101, July 2013.
7. D Brown. Standards for efficient cryptography, sec 1: elliptic curve cryptography. *Released Standard Version*, 1, 2009.
8. Quynh Dang. *Recommendation for applications using approved hash algorithms*. US Department of Commerce, National Institute of Standards and Technology, 2008.

9. Adam Dunkels, Joakim Eriksson, Niclas Finne, and Nicolas Tsiftes. Powertrace: Network-level power profiling for low-power wireless networks. 2011.
10. Adam Dunkels, Björn Grönvall, and Thiemo Voigt. Contiki-a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455–462. IEEE, 2004.
11. Adam Dunkels, Fredrik Osterlind, Nicolas Tsiftes, and Zhitao He. Software-based on-line energy estimation for sensor nodes. In *Proceedings of the 4th workshop on Embedded networked sensors*, pages 28–32. ACM, 2007.
12. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in cryptology*, pages 10–18. Springer, 1984.
13. Martin Euchner. Hmac-authenticated diffie-hellman for multimedia internet keying (mikey). 2006.
14. PUB FIPS. 186-2. digital signature standard (dss). *National Institute of Standards and Technology (NIST)*, 2000.
15. Steffen Fries and Dragan Ignjatic. On the applicability of various multimedia internet keying (mikey) modes and extensions. Technical report, 2008.
16. Michael Groves. Mikey-sakke: Sakai-kasahara key encryption in multimedia internet keying (mikey). 2012.
17. Yiliang Han, Xiaoyuan Yang, Ping Wei, Yuming Wang, and Yupu Hu. Ecgsc: elliptic curve based generalized signcryption. In *Ubiquitous Intelligence and Computing*, pages 956–965. Springer, 2006.
18. Dragan Ignjatic, Lakshminath Dondeti, Francois Audet, and Ping Lin. Mikey-rsa-r: An additional mode of key distribution in multimedia internet keying (mikey). *RFC4738, November*, 2006.
19. Chae Hoon Lim and Pil Joong Lee. A study on the proposed korean digital signature algorithm. In *Advances in CryptologyASIACRYPT98*, pages 175–186. Springer, 1998.
20. Ralph C Merkle. Secure communications over insecure channels. *Communications of the ACM*, 21(4):294–299, 1978.
21. Kim Thuat Nguyen, Nouha Oualha, and Maryline Laurent. Lightweight certificateless and provably-secure signcryptosystem for the internet of things. In *The 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-15)*, 2015.
22. Daniel R. L. Brown. Sec 2: Recommended elliptic curve domain parameters, 2010.
23. Daniel R. L. Brown, Eugene Chin, and Chi Chiu Tse. Ecc algorithms for mikey. *Work in Progress*, 2007.
24. Jun-Bum Shin, Kwangsu Lee, and Kyungah Shim. New dsa-verifiable signcryption schemes. In *Information Security and CryptologyICISC 2002*, pages 35–47. Springer, 2002.
25. Raylin Tso, Takeshi Okamoto, and Eiji Okamoto. Ecdsa-verifiable signcryption scheme with signature verification on the signcrypted message. In *Information Security and Cryptology*, pages 11–24. Springer, 2007.
26. Dae Hyun Yum and Pil Joong Lee. New signcryption schemes based on kcdsa. In *Information Security and CryptologyICISC 2001*, pages 305–317. Springer, 2001.
27. Moti Yung, Alexander W Dent, and Yuliang Zheng. *Practical signcryption*. Springer Science & Business Media, 2010.
28. Yuliang Zheng. Signcryption or how to achieve cost (signature & encryption)¡¡ cost (signature)+ cost (encryption). *http://www. pscit. monash. edu. au/yuliang/pubs/signcrypt. ps. Z*, 1999.