When Blockchain Makes Ephemeral Keys Authentic: a Novel Key Agreement Mechanism in the IoT World

Pietro Tedeschi*, Giuseppe Piro†‡, Gennaro Boggia†‡
*College of Science and Engineering, Hamad Bin Khalifa University, Doha - Qatar, e-mail: ptedeschi@mail.hbku.edu.qa
†Dept. of Electrical and Information Engineering (DEI), Politecnico di Bari, Bari (Italy), e-mail: {name.surname}@poliba.it
‡CNIT, Consorzio Nazionale Interuniversitario per le Telecomunicazioni

***Abstract*—Conventional Key Management Protocols establish secure communication channels by using asymmetric cryptography based on "fixed" public keys. In the Internet of Things context, it is preferable to have a good level of freshness of the cryptographic material. But efficient solutions leveraging "ephemeral" public keys are not available yet. The work presented herein solves this problem by conceiving a novel key agreement methodology integrating the Blockchain technology. Specifically, Blockchain is used to store X.509 certificates related to the initial fixed public keys of devices, to publish new ephemeral public keys, and to help the verification of the authenticity of ephemeral public keys without sending signatures and additional X.509 certificates. A preliminary comparison against Transport Layer Security protocol and other customized key agreement schema presented in the literature demonstrates that the proposed approach registers low communication overhead, limited energy consumptions, and acceptable communication latencies while ensuring the lowest memory footprint.**

***Index Terms*—Blockchain, Internet of Things, Key Management Protocol**

I. Introduction

Communication security represents a cornerstone requirement for the Internet of Things (IoT) [1]. In dynamic environments, the establishment of a secure communication channel starts with the implementation of a Key Management Protocol (KMP), generally based on asymmetric cryptography. In this context, the most of baseline solutions, like the well-known Transport Layer Security (TLS) protocol [2] or other customized mechanisms proposed in the literature [3], frequently adopts fixed public keys authenticated by means of X.509 certificates [4]. When applied to the IoT, however, these approaches introduce unpleasant issues. First, TLS and a customized scheme using explicit X.509 certificates [3] produce an increment of communication overhead, communication latencies, bandwidth, and energy consumptions. As demonstrated in [3], these problems can be mitigated with the adoption of implicit Elliptic Curve Qu-Vanstone (ECQV) certificates [5]. Second, available approaches do not natively offer a deep freshness of the cryptographic material. Both TLS and solutions illustrated in [3], in fact, generally consider a "fixed" public key exchange. In the case a device would use many time-limited public keys (i.e., cryptographic material usable only for particular services, in a given platform, and for a limited amount of time), it must have different X.509 certificates in memory. But, considering the inevitable growth of the memory footprint, this way of proceeding hardly scale in the IoT.

The contribution presented in this work formulates a novel key agreement methodology, which leaves the intrinsic staticity of conventional fixed public key exchange schema and introduces a lightweight procedure for establishing secure communication channels by means of ephemeral, but still authentic public keys. Specifically, the conceived solution leverages the *Blockchain* technology [6] for storing X.509 certificates related to the initial fixed public keys of IoT devices, publishing new ephemeral public keys, and helping IoT devices to verify the authenticity of ephemeral public keys without needs for additional X.509 certificates. The resulting architecture allows each device to generate, on demand, an ephemeral public key through the cryptographic approach which the Bitcoin Improvement Proposal 32 (BIP32) algorithm is based on.

A preliminary performance evaluation and the comparison against baseline approaches, including the well-known TLS protocol [2] or other customized mechanisms described in [3], clearly highlights the promising advantages introduced by the proposed solution. Specifically, conducted tests show that the devised key agreement mechanism based on the *Blockchain* technology guarantees the best trade-off among bandwidth usage, energy consumptions, communication latencies, and memory footprint.

The rest of this paper is organized as it follows: Sec. II introduces the technological background related to *Blockchain* and presents an overview of the state of the art. Sec. III describes the proposed approach. Sec. IV provides a performance evaluation. Sec. V presents the conclusions drawing possible future research directions.

II. Background on Blockchain

Blockchain is a peer-to-peer distributed database, defined as a list of blocks, chained to each other [7]. The complete copy of the database is only stored within the "full nodes' of the *Blockchain*, with the benefit that no single point of failure exists because there is not a central authority that manages it. A block generally stores a transaction or a group of transactions. The *Blockchain* is immutable and transparent because a transaction cannot be tampered once it is inserted in a block, and it gives a transparent view on how and what data are exchanged in the network [8]. As depicted in the Fig. 1, a block is identified by a cryptographic hash value. Furthermore, it contains the previous hash block which allows connecting the blocks and then create a chain, a timestamp, the transactions and the transactions number. The first block is called *genesis block*.

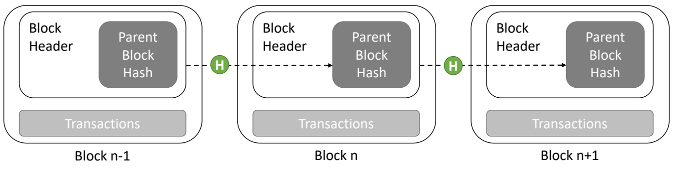


Fig. 1: The Blockchain Architecture.

Each node interacts with other nodes by using a key pair composed of a public key and a private key. While the private key is used to sign a transaction, a public key is used as a pseudonym network address. Each transaction is sent in broadcast by a node. Then, it should be validated by special entities, namely *miners*, which are in charge for effectively creating the block and store it within the database. The validation process is called *mining*. The validation mechanism leverages a consensus protocol, which specifies the agreement algorithm used to achieve data consistency and reliability. Possible solutions are: Proof of Work [9], Proof of Stake [10], Proof of Authority [11] and Practical Byzantine Fault Tolerance [12]. All receiving nodes have the ability to verify the transaction by using the sender public key [13].

A. The BIP32 algorithm

The Hierarchical Deterministic BIP32 protocol defines the procedure to build a tree of key pairs, starting from a random seed, and to generate a Hierarchical Deterministic wallet (i.e., a repository storing private keys related to the generated public keys) [14]. A key derived from a parent key simply refers to as child key.

Let $\mathcal{G}$ be an elliptic curve group of order n, and $G \in \mathcal{G}$ be a generator (or base point) of the curve. Normally, the well-known elliptic curve cryptography assumes that a private-public key pair is generated in a way that (i) the private key S is defined by choosing a random integer k, with $0 < k < n$, and (ii) the correspondent public key P is obtained by means a scalar point multiplication $P = k \cdot G$.

The BIP32 algorithm extends the aforementioned approach introducing the concept of the *extended* key, defined as the concatenation of the root private key of 256 bits, S_{root}, and the chain code of 256 bits, c, that is (S_{root}, c). Given a public string key $string$ and the root seed $seed$, it is calculated as $(S_{root}, c) = \Gamma(string, seed)$, where $\Gamma(string, seed)$ represents a symmetric authentication algorithm (e.g., HMAC-SHA512) that uses $string$ as the key and $seed$ as the content to be hashed. The related root public key, P_{root}, is generated as $P_{root} = S_{root} \cdot G$.

The extended key can generate 2^{31} child key pairs and 2^{31} hardened child key pairs, each one identified by the 32-bit integer index i. In particular, child key pairs are defined by an index falling in the interval $[0 \sim 2^{31} - 1]$ and hardened child key pairs are defined by an index falling in the interval $[2^{31} \sim 2^{32} - 1]$.

For the i-th child key pair, the generation procedure starts by computing $I = \Gamma(c, P_{root}||i)$ and by splitting the obtained output into two 256 bits long sequences I_L^c and I_R^c. Therefore, the child private key, S_i^c, is set to $S_i^c = S_{root} + I_L^c \mod n$. The related child public key, P_i^c, is obtained as $P_i^c = P_{root} + I_L^c \cdot G$.

In turns, each derived child key pair could produce other child key pairs, where the chain code c_i is set to I_R^c. Regarding the hardened child key pair (keys that cannot have child keys), the procedure assumes to calculate $I = \Gamma(c, 0x00||S_{root}||i)$, where $0x00$ is a pad. As in the previous case, the output of the hash function is divided into two 256 bits long sequences I_L^h and I_R^h. According to [13] and [14], the hardened child private key, S_i^h, is set to $S_i^h = S_{root} + I_L^h \mod n$; the hardened child public key, SP_i^h, is set to $P_i^h = P_{root} + I_L^h \cdot G$.

BIP32 can be used to generate a tree of ephemeral keys, that can be authenticated without the intervention of a trusted, centralized or distributed, authority. In fact, by assuming that a child key pair is not hardened and that the chain code c, the parent public key P_{root}, and the key index i are known, a generic network node is able to verify that a child public key $P_i^c = P_{root} + I_L^c \cdot G$ is really derived from a parent key.

To conclude, it is important to remark that the BIP32 algorithm is generally used by *Blockchain* for dynamically updating the key pair used to sign a transaction, while guaranteeing a good level of anonymity of involved users. This paper introduces a novel approach: BIP32 is integrated within the key agreement algorithm for generating, on-demand, ephemeral, and authentic public keys.

B. Key Management Protocols and Blockchain

Web of Trust and *Sovrin* are possible platforms enabling KMP protocols. *Web of Trust* has been conceived for creating a decentralized and *Blockchain*-based Public Key Infrastructure, where involved nodes can vote the authenticity of public key certificates associated to end users. *Sovrin* extends the aforementioned approach by leveraging the Blockchain technology for the self-sovereign identity service. The usage of *Blockchain* in the key management process was addressed in two recent scientific contributions. The work in [15] proposes a security framework for vehicular communication systems. A decentralized architecture based on *Blockchain* is adopted to facilitate the key management in a heterogeneous domain, thus reducing the latencies due to the transmission of keys, especially during the handover. The *Blockchain* stores cryptographic materials of vehicles, e.g. identities, public keys, and correspondent certificates. The work in [16] resolves the issue related to the storage of a Certification Revocation List within a single Certification Authority. The contribution assumes to distribute and share the Certification Revocation List across multiple Certification Authoritys, by adopting a decentralized ledger. Accordingly, they avoid the possibility to have a point of failure when a Certification Authority goes down.

At the time of this writing, and to the best of author's knowledge, *Blockchain* was never used to dynamically manage ephemeral public keys during a KMP protocol. Therefore, the proposed approach is novel with respect to the current state of the art.

III. The Proposed Approach

The reference scenario considered in this paper is depicted in Fig. 2 and envisaged in the H2020 symbIoTe project (see https://www.symbiote-h2020.eu). It consists of a federation of IoT platforms, connected to each other through gateway nodes. Smart devices may establish secure channels while moving from one platform to another one, by using new ephemeral cryptographic material. Herein, a private *Blockchain* is used as an immutable and tamper-proof distributed ledger to store the initial X.509 certificates of IoT devices, to publish (and track the history of) ephemeral public keys, and to help the verification of the authenticity of ephemeral public keys without sending X.509 certificates. Thanks to integrity, transparency, and resiliency properties, *Blockchain* does not allow a node to forge a public key, but only to verify its correctness. Gateways are full *Blockchain* nodes, store the entire *Blockchain* and work as miner/validator for blocks and transactions. The Proof of Authority is selected as the consensus mechanism for authenticating, validating, and recording the transactions within the *Blockchain*. In particular, it is assumed that the whole system embraces a set of predefined nodes, namely trusted authorities. During the time, a specific authority is selected among the others on a round-robin basis. Such a node will be table to validate transactions and blocks during a given time window. Note that the considered approach does not require the resolution of a mathematical problem (i.e., like what is done with the Proof of Work). This brings to less computational demands.

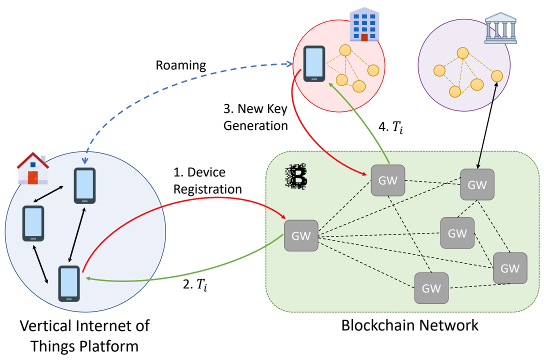


Fig. 2: Reference Network Architecture.

As illustrated in Fig. 3, the designed KMP protocol embraces four different phases: setup, device registration, ephemeral key generation, and key agreement. Each step includes many atomic operations, as discussed in what follows.

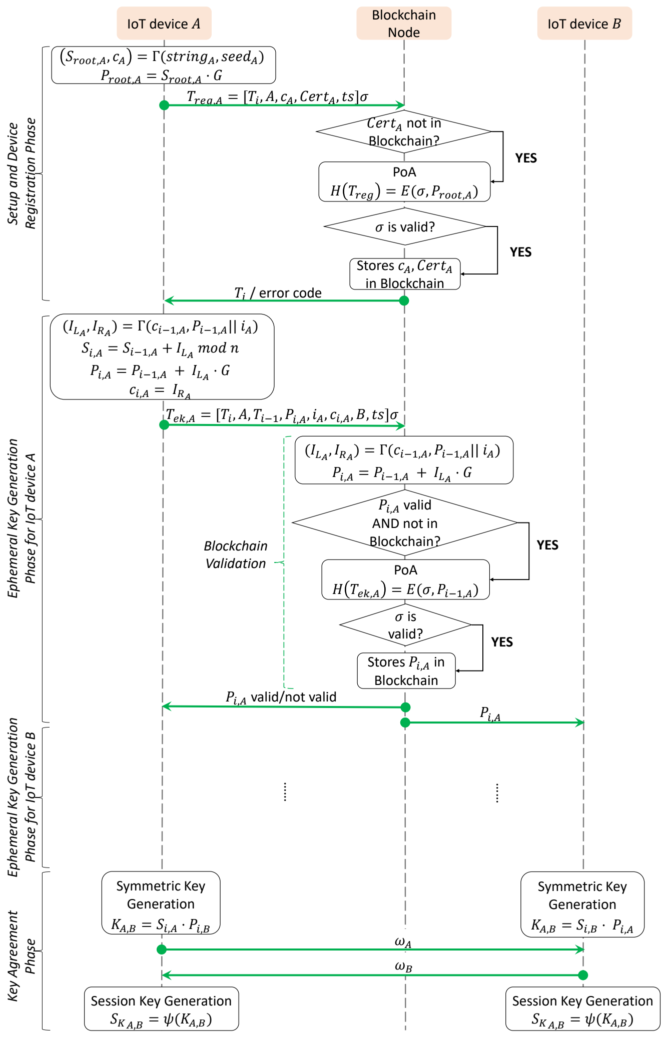

IoT device A
Blockchain Node
IoT device B
Setup and Device Registration Phase
$(S_{root,A}, c_A) = \Gamma(string_A, seed_A)$
$P_{root,A} = S_{root,A} \cdot G$
$T_{reg,A} = [T_i, A, c_A, Cert_A, ts]\sigma$
$Cert_A$ not in Blockchain?
YES
PoA
$H(T_{reg}) = E(\sigma, P_{root,A})$
σ is valid?
YES
Stores $c_A, Cert_A$ in Blockchain
T_i / error code
Ephemeral Key Generation Phase for IoT device A
$(I_{L_A}, I_{R_A}) = \Gamma(c_{i-1,A}, P_{i-1,A} || i_A)$
$S_{i,A} = S_{i-1,A} + I_{L_A} \ mod \ n$
$P_{i,A} = P_{i-1,A} + I_{L_A} \cdot G$
$c_{i,A} = I_{R_A}$
$T_{ek,A} = [T_i, A, T_{i-1}, P_{i,A}, i_A, c_{i,A}, B, ts]\sigma$
Blockchain Validation
$(I_{L_A}, I_{R_A}) = \Gamma(c_{i-1,A}, P_{i-1,A} || i_A)$
$P_{i,A} = P_{i-1,A} + I_{L_A} \cdot G$
$P_{i,A}$ valid AND not in Blockchain?
YES
PoA
$H(T_{ek,A}) = E(\sigma, P_{i-1,A})$
σ is valid?
YES
Stores $P_{i,A}$ in Blockchain
$P_{i,A}$ valid/not valid
$P_{i,A}$
Ephemeral Key Generation Phase for IoT device B
Key Agreement Phase
Symmetric Key Generation
$K_{A,B} = S_{i,A} \cdot P_{i,B}$
Symmetric Key Generation
$K_{A,B} = S_{i,B} \cdot P_{i,A}$
ω_A
ω_B
Session Key Generation
$S_{K_{A,B}} = \psi(K_{A,B})$
Session Key Generation
$S_{K_{A,B}} = \psi(K_{A,B})$

Fig. 3: Key Management Protocol.

A. Setup phase

At the beginning, the j-th device is pre-configured by the administrator with a string key $string_j$ and a seed $seed_j$. Therefore, according to both BIP32 algorithm and Elliptic Curve Cryptography (ECC), it calculates root private key $S_{root,j}$, chain code $c_{root,j}$, and root public key as following:

$$(S_{root,j}, c_{root,j}) = \Gamma(string_j, seed_j) \tag{1}$$

$$P_{root,j} = S_{root,j} \cdot G \tag{2}$$

Then, the public key is stored within an X.509 certificate, namely $Cert_j$, signed by a trusted authority.

B. Device registration phase

During the registration phase, the j-th device sends to its gateway, node of the *Blokchain* network, the X.509 certificate, $Cert_j$, storing its root public key and the previously calculated chain code, $c_{root,j}$. To this end, it issues a registration transaction, $T_{reg,j}$, containing the transaction ID T_i (calculated as the hash function of the entire transaction), the device index j, the X.509 certificate $Cert_j$, the root chain code $c_{root,j}$, the timestamp ts, and a transaction signature σ:

$$T_{reg,j} = \{ \underbrace{T_i, j, c_{root,j}, Cert_j, ts}_{transaction\ payload} \}||\sigma, \quad (3)$$

where $T_i = H(j, c_{root,j}, Cert_j, ts)$ and $\sigma = E(H(T_i, j, c_{root,j}, Cert_j, ts), S_{root,j})$.

The gateway of the IoT platform which the device is attached to delivers the transaction to the rest of the *Blockchain* networks. Here, in line with the Proof of Authority consensus mechanism, a random gateway is chosen to verify the transaction signature σ, the authenticity and the certificate, and to store the received data within the *Blockchain*. If consensus is reached, the *Blockchain* sends back to the IoT device the ID of the processed transaction.

C. Ephemeral key generation phase

The cryptographic material stored in the registration transaction is used by the j-th device to generate, in the future, new ephemeral keys. The example reported in Fig. 3 depicts a KMP protocol established between the devices A and B. The ephemeral key generation phase is performed by both the devices.

According to the BIP32 algorithm (non-hardened derivation with extended keys), device A and device B calculate $I_A = \Gamma(c_{i-1,A}, P_{i-1,A}||i_A)$ and $I_B = \Gamma(c_{i-1,B}, P_{i-1,B}||i_B)$. Then, they split the obtained output into two sequence of 256 bits each. Specifically, device A generates I_{L_A} and I_{R_A}, while device B generates I_{L_B} and I_{R_B}. It is important to note that $c_{i-1,j}$ and $P_{i-1,j}$ are the latest chain code and the latest public key stored by the j-th device in the *Blockchain*, respectively.

Device A generates an ephemeral key pair with:

$$S_{i,A} = S_{i-1,A} + I_{L_A} \mod n, \quad (4)$$

$$P_{i,A} = P_{i-1,A} + I_{L_A} \cdot G, \quad (5)$$

and sets $c_{i,A} = I_{R_A}$.

At the same time, device B generates an ephemeral key pair with:

$$S_{i,B} = S_{i-1,B} + I_{L_B} \mod n, \quad (6)$$

$$P_{i,B} = P_{i-1,B} + I_{L_B} \cdot G, \quad (7)$$

and sets $c_{i,B} = I_{R_B}$.

Each IoT device registers the new ephemeral public key and the new chain code within the *Blockchain*.

First, device A issues a new transaction, that is $T_{ek,A}$, containing the new transaction ID T_i (obtained and calculated as the hash function of the entire transaction), the device index A, the previous transaction ID T_{i-1}, the new ephemeral public key $P_{i,A}$, the index of the current ephemeral public key, the new chain code $c_{i,A}$, the device index B, the timestamp ts, and a transaction signature σ:

$$T_{ek,A} = \{ \underbrace{T_i, A, T_{i-1}, P_{i,A}, i_A, c_{i,A}, B, ts}_{transaction\ payload} \}||\sigma, \quad (8)$$

where $T_i = H(A, T_{i-1}, P_{i,A}, i_A, c_{i,A}, B, ts)$ and $\sigma = E(H(T_i, A, T_{i-1}, P_{i,A}, i_A, c_{i,A}, B, ts), S_{i-1,A})$.

The gateway of the IoT platform which device A is attached to deliver the transaction to the rest of the *Blockchain* network. First of all, the gateway verifies that the ephemeral public has never been stored in the *Blockchain*. Now, since the gateway knows the parameters used by A to generate the ephemeral public key $P_{i,A}$ (because they are stored in the *Blockchain)*, it is able to verify that the ephemeral public key is really derived from the right parent key and the right chain code. In the affirmative case, the transaction will be processed according to the Proof of Authority consensus mechanism, as already discussed before.

In the case ephemeral public key $P_{i,A}$ is valid, the gateway sends it to device B.

In a similar manner, device B issues a new transaction, that is $T_{ek,B}$, containing the new transaction ID T_i (calculated as the hash function of the entire transaction), the device index B, the previous transaction ID T_{i-1}, the new ephemeral public key $P_{i,B}$, the index of the current ephemeral public key, the new chain code $c_{i,B}$, the device index A, the timestamp ts, and a transaction signature σ:

$$T_{ek,B} = \{ \underbrace{T_i, B, T_{i-1}, P_{i,B}, i_B, c_{i,B}, A, ts}_{transaction\ payload} \}||\sigma, \quad (9)$$

where $T_i = H(B, T_{i-1}, P_{i,B}, i_B, c_B, A, ts)$ and $\sigma = E(H(T_i, B, T_{i-1}, P_{i,B}, i_B, c_{i,B}, A, ts), S_{i-1,B})$.

Also, in this case, the gateway of the IoT platform which device B is attached to deliver the transaction to the rest of the *Blockchain* network and the Proof of Authority consensus mechanism will be used to finalize the authentication of the ephemeral key and the mining process.

In the case ephemeral public key $P_{i,B}$ is valid, the gateway sends it to device A.

D. Key agreement phase

During the key agreement phase, the devices exchange their ephemeral keys, some random parameters, and authentication tags for the final mutual authentication.

In the case ephemeral public keys are authentic, device A and device B calculate the symmetric key through the Elliptic Curve Diffie-Hellman (ECDH) algorithm. Specifically, device A computes:

$$K_{A,B} = S_{i,A} \cdot P_{i,B} \quad (10)$$

and device B computes:

$$K_{A,B} = S_{i,B} \cdot P_{i,A} \quad (11)$$

Furthermore, devices exchange authentication tags, thus making the resulting protocol resilient against replay and impersonation attacks. Thus, devices A computes:

$$\omega_A = \Gamma(K_{A,B}, (A, B, P_{i,A}, P_{i,B})) \quad (12)$$

and sends it to B. Similarly, devices B computes:

$$\omega_B = \Gamma(K_{A,B}, (B, A, P_{i,B}, P_{i,A})) \quad (13)$$

and sends it to B.

Finally a session key $S_{K_{A,B}}$ is be derived through a Key Derivation Function ψ, like the HMAC-based Extract-and-Expand Key Derivation Function, that is:

$$S_{K_{A,B}} = \psi(K_{A,B}) \tag{14}$$

IV. Performance Evaluation

The proposed approach has been evaluated in a IoT scenario based on the IEEE 802.11ah [17] communication technology, where the data transmission rate is set to 54 Mbps/s, the Maximum Transmission Unit is set to 1500 bytes, and the average Round Trip Time (RTT) is set in a range from 3 ms and 12 ms. Its performance has been compared against those achieved by the following baseline strategies: the TLS 1.2 protocol adopting explicit X.509 certificates, the lightweight KMP presented in [18] and based on explicit X.509 certificates, and the lightweight KMP presented in [18] and based on implicit ECQV certificates. From the security perspective, explicit X.509 certificates are formatted according to the Privacy Enhanced Mail specification. Their digital signature is generated according the Elliptic Curve Digital Signature Algorithm (ECDSA) algorithm, through the *secp256k1* elliptic curve provided by Standards for Efficient Cryptography Group (SECG) [19]. The resulting size of an explicit certificate is equal to 833 bytes. Implicit ECQV certificates are generated by considering the well-known *secp160r1* curve provided by NIST [20]. Their size is set to 78 bytes [18]. In all of these cases, each device may authenticate its fixed public key through a certificate chain. To properly evaluate the impact of this functionality, the size of the certificate chain is chosen in a range from 1 to 3 certificates.

Fig. 4 shows the total number of bytes exchanged between the two devices willing to establish a secure communication channel and the gateway (which is the access point) during the provisioning of the KMP protocol. First of all, it is possible to observe that all the solutions considered in the comparison register a communication overhead that increases with the size of the certificate chain. As expected, the higher the number of certificates that each device should deliver during the authentication process, the higher the number of bytes exchanged within the network. Moreover, it is evident that the well-known TLS protocol experiences the highest communication overhead, due to the huge amount of data exchanged during all the phases of the handshake protocol. Regarding the lightweight protocol presented in [18], two comments can be formulated. From one side, the usage of explicit X.509 brings to a communication overhead significantly higher with respect to the solution proposed in this paper. A quite reverse behavior is observed when implicit ECQV certificates are used. Here, the lightweight protocol presented in [18] reaches a lower communication overhead when the size of the certificate chain limited. But, when the certificate chain stores more than 3 certificates, the minimum communication overhead is measured for the solution proposed in this paper.

It is important to note that the communication overhead also gives an idea about the amount of energy consumed by involved devices to exchanges data during the execution of the KMP protocol. For this reason, Fig. 4 also shows the energy consumptions, expressed as the total number of *unit of energy* spent on each transmitted byte. In line with the previous comments, the proposed approach presents very limited energy demands and the amount of consumed energy reduces to the minimum values when the size of the certificate chain used by other schema is equal or higher than 3.

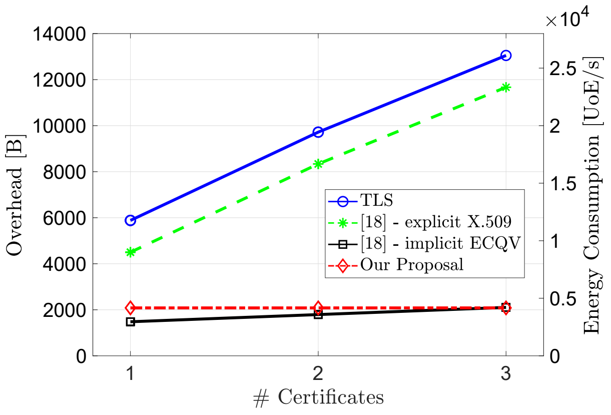
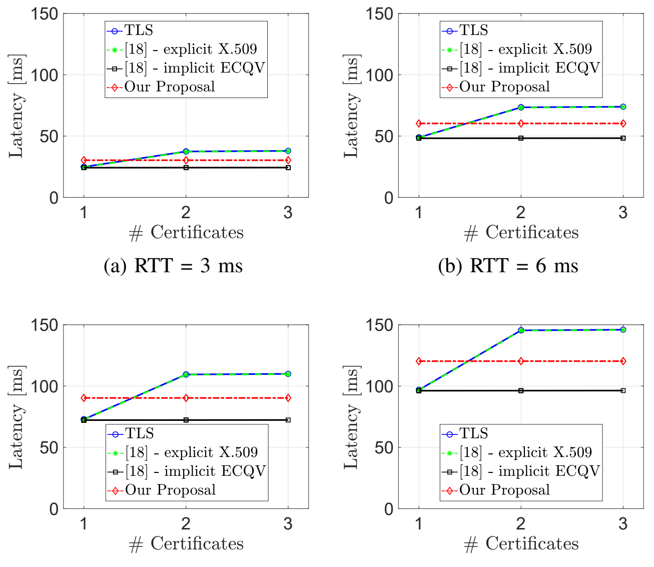


Fig. 4: Communication overhead and energy consumptions.

The amount of time needed to finalize the KMP is reported in Fig. 5. As expected, all the investigated solutions register an increment of communication latencies when the average RTT increases. The TLS protocol and the lightweight strategy presented in [18] and based on explicit X.509 certificates reach the worst performance and register similar communication latencies. They, in fact, envisage the exchange of the highest number of data and the delivery of the same number of packets. In addition, both the solutions experience higher communication latencies while the size of the certificate chain increases. In this case, in fact, data exchanged during the authentication process can be delivered in a number of packets that increases with the size of the certificate chain. And, given the average Round Trip Time, the amount of time spent to complete the overall message exchange increases as well. The lightweight protocol presented in [18] and based on implicit ECQV certificates provides lower delays due to the lower number of packets issued during the time. But, despite this evident result, the proposed approach registers satisfactory communication latencies, ranging from 30 ms and 130 ms.

To provide a further insight, the amount of memory for storing the cryptographic material useful to accomplish N consecutive key negotiation procedures with ephemeral public keys is reported in Fig. 6. The results clearly highlight that the solution proposed in this work outperforms the approaches available in the current literature by always registering the minimum memory footprint. The protocols taken into account for the comparison are in charge of storing N different certificates storing the set of time-limited public keys. Here, the memory footprint strictly depends on the size of the X.509 certificate (note that the solution leveraging implicit ECQV



(a) RTT = 3 ms (b) RTT = 6 ms

(c) RTT = 9 ms (d) RTT = 12 ms

Fig. 5: Average communication latencies.

certificates register better performance). On the contrary, the novel methodology presented in this work does not require the storage of N different certificates because the ephemeral public keys are generated, on demand, during the time.

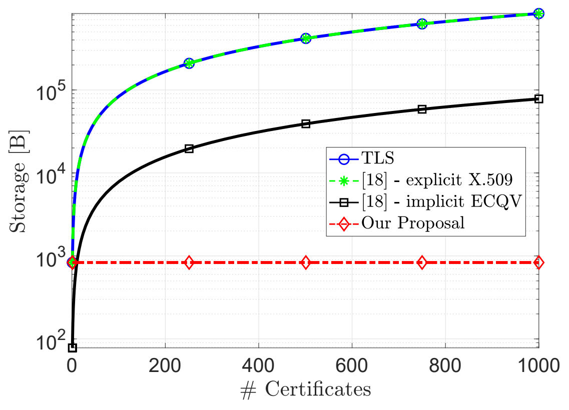


Fig. 6: Memory Footprint.

V. Conclusions and Future Works

The work presented herein promotes the *Blockchain* technology as a valid instrument for designing flexible Key Management Protocols, able to generate and use ephemeral, but authentic, public keys. The results clearly demonstrate that the conceived strategy ensures the best trade-off among bandwidth usage, energy consumptions, communication latencies, and memory footprint. Future research activities in this direction intend to deeply investigate the security proof of the conceived approach and to develop a proof-of-concept implementation (integrating for instance available platforms like Ethereum), as well as to evaluate system performances in terms of bandwidth, energy, and latency requirements.

VI. Acknowledgements

This work was framed in the context of the project symbIoTe, which receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreement 688156.

References

[1] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, "Internet of Things security: A survey," *Journal of Network and Computer Applications*, vol. 88, pp. 10 – 28, 2017.

[2] T. Dierks, "The transport layer security (tls) protocol version 1.2," 2008.

[3] S. Sciancalepore, A. Capossele, G. Piro, G. Boggia, and G. Bianchi, "Key management protocol with implicit certificates for IoT systems," in *Proc. of ACM workshop on IoT challenges in Mobile and Industrial Systems (IoT-Sys)*, 2015, pp. 37–42.

[4] S. L. Keoh, S. S. Kumar, and H. Tschofenig, "Securing the Internet of Things: a standardization perspective," *IEEE Internet of Things Journal*, vol. 1, no. 3, pp. 265–275, 2014.

[5] M. Campagna, "SEC 4: Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV)," *Certicom Research, Mississauga, ON, Canada, Tech. Rep*, 2013.

[6] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *International Journal of Web and Grid Services*, 2017.

[7] S. Underwood, "Blockchain beyond bitcoin," *Commun. ACM*, vol. 59, no. 11, pp. 15–17, Oct. 2016. [Online]. Available: http://doi.acm.org/10.1145/2994581

[8] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[9] D. T. T. Anh, M. Zhang, B. C. Ooi, and G. Chen, "Untangling blockchain: A data processing view of blockchain systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, no. 99, pp. 1–1, 2018.

[10] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei, and C. Qijun, "A review on consensus algorithm of blockchain," in *Proc. of IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct 2017, pp. 2567–2572.

[11] Proof of Authority Chains. [Online]. Available: https://github.com/paritytech/parity/wiki/Proof-of-Authority-Chains

[12] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future Generation Computer Systems*, 2017.

[13] A. M. Antonopoulos, *Mastering Bitcoin: unlocking digital cryptocurrencies*. 1st ed. Sebastopol, CA, USA: O'Reilly Media, Inc., 2014.

[14] D. Khovratovich and J. Law, "BIP32-Ed25519: Hierarchical Deterministic Keys over a Non-linear Keyspace," in *Proc. of IEEE European Symposium on Security and Privacy Workshops (EuroSPW)*, April 2017, pp. 27–31.

[15] A. Lei, H. Cruickshank, Y. Cao, P. Asuquo, C. P. A. Ogah, and Z. Sun, "Blockchain-based dynamic key management for heterogeneous intelligent transportation systems," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1832–1843, 2017.

[16] M. Baldi, F. Chiaraluce, E. Frontoni, G. Gottardi, D. Sciarroni, and L. Spalazzi, "Certificate validation through public ledgers and blockchains," in *First Italian Conference on Cybersecurity (ITASEC)*, 2017.

[17] E. Khorov, A. Lyakhov, A. Krotov, and A. Guschin, "A survey on IEEE 802.11 ah: An enabling networking technology for smart cities," *Computer Communications*, vol. 58, pp. 53–69, 2015.

[18] S. Sciancalepore, G. Piro, G. Boggia, and G. Bianchi, "Public key authentication and key agreement in iot devices with minimal airtime consumption," *IEEE Embedded Systems Letters*, vol. 9, no. 1, pp. 1–4, March 2017.

[19] Standards for Efficient Cryptography, "Sec 2: Recommended elliptic curve domain parameters," *Standards for Efficient Cryptography Group, Certicom Corp*, 2010.

[20] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendation for key management part 1: General (revision 3)," *NIST special publication*, vol. 800, no. 57, pp. 1–147, 2012.