End-to-End Secure IoT Node Provisioning

Ilker Yavuz[1,2] and Berna Ors[1]
[1] Department of Electronics and Communication Engineering, Istanbul Technical University, Istanbul, 34469, Turkey
[2] AirTies Wireless Networks, Istanbul, 34394, Turkey
Email: {yavuzi; siddika.ors}@itu.edu.tr

Abstract —Security of Internet of Things (IoT) has been one of the most critical topics since IoT devices took part in daily life. Due to resource constrained nature of IoT networks, meeting requirements of a secure infrastructure always becomes a challenge. The most prevalent method is to rely on conventional application layer protocols to secure IoT network traffic but due to IoT device capabilities, limited mobile network resources and battery powered structure of IoT nodes, most of them are not applicable in practice. Provisioning a new node into a running network also suffers from these challenges. In this study, we investigate whether pure algorithm based protocols can be used to solve secure provisioning of resource limited IoT devices problem. Trusted IoT node provisioning requires new node authentication, authorization for network credentials, secret key generation for data privacy, and distribution of secret keys. Besides that, key management for rejoining nodes should be considered due to mobility of IoT nodes. We propose an Elliptic Curve Cryptography (ECC) based solution to cover these security requirements. Our design environment has also ability to analyze power consumption of each node during node enabling into a secure network.

Index Terms—IoT, provisioning, bootstrapping, Elliptic Curve, digital signature, public key cryptography, power, security, COOJA, Contiki, Powertrace, wireless sensor networks

I. Introduction

The characteristics of IoT nodes are measuring huge amount of data, mobility, and wireless connectivity over a variety of links such as IEEE 802.15.4, low-power IEEE 802.11, or IEEE 802.15.1. The privacy of these data is an important part of IoT networks and depends on limited resources of the nodes. By the nature of IoT networks, nodes have limited processing power, battery and ensuring security under these limitations requires lightweight solutions which is differentiating from conventional internet security frameworks.

Due to resource constrained structure of IoT nodes, secure node provisioning for a running network before being functional should be considered differently. Most of IoT networks are dynamic environments and network nodes are not static. Besides that, generally, nodes are mobile devices and need to leave and rejoin networks several times. For example, a cloud based voice assistant should have security credentials before collecting and sending environment data for the privacy of users or a temperature sensor on a moving device in an industrial area leaves a network and then rejoins during its operation and needs to be provisioned several times depending on security policies of the network.

As given above examples, each IoT node needs to be provisioned with network related information and key, before being functional. There are several methods proposed by standards for network device provisioning [1]. These standards are not directly point out the solution so we target to propose an efficient and secure provisioning method using elliptic curve cryptography [2]. Our contribution to the literature will be surveying previous studies and proposing a new, optimized method.

In this paper, we present our end-to-end public key cryptography based IoT node provisioning method. Our method is based on elliptic curve cryptography and optimized for lightweight IoT nodes. The proposed schema offers node authentication, node authorization and privacy without using certificate based application layer protocol. In addition, our proposed schema is resistant to man in the middle attacks [3] and replay attacks [4].

A detailed literature survey on standards and secure provisioning methods for IoT networks are given in section II. Our motivation for this research, our proposed schema, and background for underlying security algorithms are given in section III. Implementation details, findings and numerical results are given in IV. In the last section, we evaluate our findings, discuss on them and list the future works.

II. Related Work

When applying conventional security schema to IoT devices, limited device resources become a barrier therefore pre-shared key based solutions became quickest solution to overcome resource constrained environment bottlenecks [5]. Flashing a security key on an IoT device during manufacturing avoids considerable process during run-time but brings along several problems such as key leakage management [5].

For the sake of attacking open issues in secure device provisioning, we will first elaborate networking group standards to classify present methods and then list recent studies in the literature. Based on the survey [6] conducted by Network Working Group, IoT bootstrapping mechanisms are classified into managed, opportunistic and leap-of-faith, hybrid, and peer-to-peer (P2P) methods. Each method is discussed with their examples, advantages and disadvantages. Managed

Manuscript received February 7, 2021; revised July 4, 2021.
Corresponding author email: yavuzi@itu.edu.tr
doi:10.12720/jcm.16.8.341-346

methods are based on sharing credentials in advance during manufacturing or via token such as smart card.

Extensible Authentication Protocol (EAP) [7] based TLS certificate and PSK (Pre-Shared Key) methods are given as examples for managed provisioning method. An application layer protocol based CoAP-EAP [8] and Protocol for Carrying Authentication for Network Access (PANA) [9] are other managed provisioning method examples. PANA does not define a new protocol on the other hand uses EAP over UDP. A good example for PANA was given in [10]. Sarikaya et. al. shows how a 6LowPAN border router functions as a PANA Authentication Agent (PAA) and authenticates a constrained PANA Client identity while joining to network. In their schema, router is responsible to network and security parameters to the joining device after client authenticated successfully. Another bootstrapping method in this class Generic Bootstrapping Architecture (GBA) which is based on 3GPP Authentication [11]. Kerberos protocol [12] is also given as a network authentication protocol allowing endpoints to communicate over an insecure network using symmetric keys [6].

Peer to peer methods given in [6] solved credential pre-sharing problems. This method generally relies on out-of-band (OOB) channel communication to protect to communication man-in-the-middle attacks. OOB communication requires extra channel so it requires extra cost and management. EAP-NOOB [13] is one the implementations of P2P OOB based provisioning method.

Opportunistic and Leap-of-faith bootstrapping method is the last one counted in this group. [14] was given as an example of this method. Secret key sharing is based on some messaging sequences. This method is open for vulnerabilities since secret key is shared without any security protection after first negotiation.

To the best of our knowledge, classification given in [6] covered methods in the literature. For instance, [15], [13], [16], [17], and [18] can be grouped in EAP or EAP-P2P hybrid methods. EAP authentication framework details can be found in [16]. In addition, [19] is another P2P provisioning method based on OOB communication channel for credential sharing.

Certification based provisioning solutions such as [5], [20] require more storage for certification. This is one of the well-known weaknesses of certificate based provisioning methods given in the literature. Given studies [21] and [22] use compression methods to shrink certificate size as well.

III. Proposed Provisioning Architecture

A. Security Properties

1) Elliptic curve common secret generation

In this study, Elliptic Curve Based Diffie-Hellman (ECDH) key establishment protocol is used to generate shared secret since ECDH is energy efficient and requires less ROM/RAM compared to RSA [23].

ECDH is a key agreement protocol which provides each node holding a public-private key pair to create a shared secret agreement over an unsecure communication channel.

In public key infrastructure, the only thing that nodes have to align is elliptic curve domain parameters which can be public.

ECDH relies on two public parameters, a large prime number p and an integer g that is less than p. These parameters can be predefined public values for the network. Each node chooses a private integer, node owns private key for ECDH, which are $Private_A = a$ and $Private_B = b$, respectively. Then each node generates its public keys, $Public_A = g^a mod_p$ and $Public_B = g^b mod_p$, based upon the known public parameter and their corresponding private keys. The private keys are unique and different for each node. Each node sends its public to other party via public communication channel.

$$(g^a)^b mod_p = (g^b)^a mod_p \quad (1)$$

2) Elliptic Curve Digital Signature Algorithm (ECDSA)

Digital Signature is a kind of unique message derived from a message that is generated by the authority who owns the private key and can be verified by many who has the public key of signer [24]. Elliptic Curve Digital Signature is also a kind of Digital Signature Algorithm (DSA) which uses ECC background to generate and verify unique message.

Public and private key pairs are used for digital signing process therefore ECDSA has 2 main steps which are key generation and signing.

Verification step needs signers public key, message which is signed and sign of the message therefore receiver can verify if received sign is derived from acquired message with senders' public key. Message signing process is not applied directly to message. Instead, unique message hash is signed hence signed data size is reduced. Block level architecture for digital signing flow can be seen in Fig. 1.

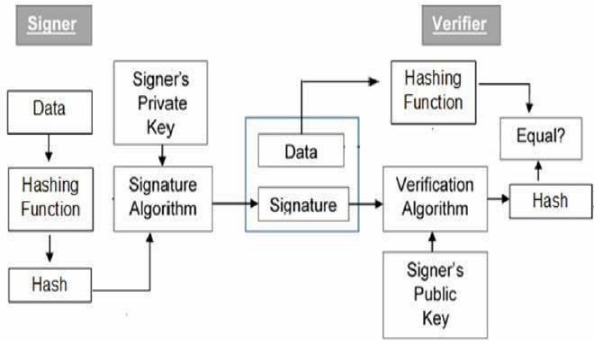

Fig. 1. Digital signature scheme

ECDSA steps are similar to the DSA scheme and explained with the following steps.

Key Generation for ECDSA:

1. Use an elliptic curve E with modulus p, curve coefficients a, b, and point A which generates a cyclic group of prime order q

2. Choose a random integer d with $0 < d < q$
3. Compute $B = d.A$

The keys are now $k_{pub} = (p, a, b, q, A, B)$, $k_{pr} = (d)$. Note this is a discrete logarithm problem where the integer d is the private key and the result of the scalar multiplication, point B, is the public key. Similar to DSA, the cyclic group has an order q which should have a size of at least 160 bit or more for higher security levels [23].

ECDSA Signature Generation:

Like DSA, an ECDSA signature consists of a pair of integers (r, s). Each value has the same bit length as q, which makes for fairly compact signatures. Using the public and private key, the signature for a message x is computed as follows:

1. Choose a random integer as ephemeral key k_E with $0 < k_E < q$.
2. Compute $R = k_E.A$
3. Let $r = x_R$ ($r = x\ coordinate$ of the point R)
4. Computes $s \equiv (h(x) + d \cdot r){k_E}^{-1} mod\ q$

The message x has to be hashed using the function h in order to compute s. The hash function output length must be at least as long as q.

ECDSA Signature Verification:

1. Compute auxiliary value $\omega \equiv s^{-1} mod\ q$
2. Compute auxiliary value $u_1 \equiv \omega \cdot h(x) mod q$
3. Compute auxiliary value $u_2 \equiv \omega \cdot r\ mod q$
4. Compute $P = u_1 A + u_2 B$
5. The verification $ver_{k_{pub}}(x, (r, s))$ follows from:

$$x_p = \begin{cases} \equiv r\ mod\ q, & Valid\ Signature \\ \not\equiv r\ mod\ q, & Invalid\ Signature \end{cases}$$

In the fifth step, the notation x_p indicates the $x-$coordinate of the point P. The verifier accepts a signature (r, s) only if the x_p has the same value as the signature parameter r $(mod\ q)$.

B. Key Management Topology

Our proposed secure IoT node provisioning schema is based on ECC functionalities listed in section III-A. The main problems of IoT node provisioning are new node authentication, authorization for relevant credentials and security key distribution. Storing node information to grant access is the authentication problem. It is not feasible to distribute all nodes' credentials to all border routers on the field so credential should be stored in central location and distributed on demand. When a new node requests to join an IoT network, credentials are distributed to master node with the permission of central vault which is called Authentication Cloud (AC). AC can be manufacturer's cloud location which stores nodes' identity and related security credentials which are serial number (SN) and public key pairs. Once master node has the approval of AC for new nodes, the node can be added to trusted list and is eligible to share network secret key. Since all these operations have identity information and secret key, ECC based sign & verification mechanism given in section III-A are used for cryptographic operations. Therefore, our ECC implementation is deployed on all IoT nodes and on server side. Under this assumption, interoperability problem will not be a concern. Beyond that, sharing ECC curve parameters will be enough for interoperability.

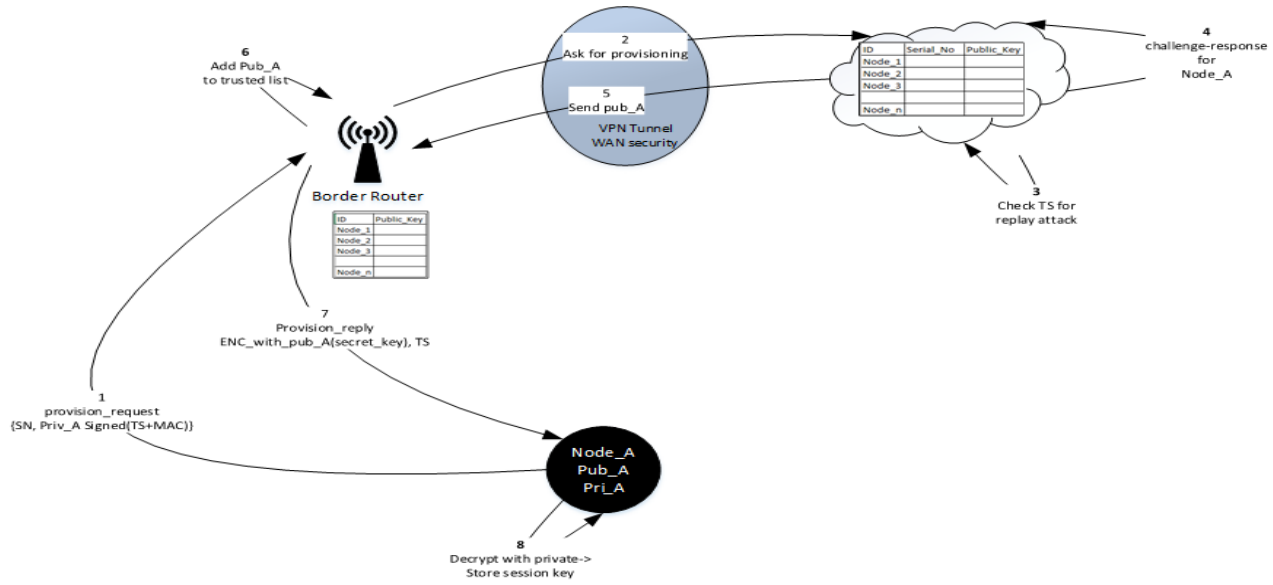

Fig. 2. Secure IoT node provisioning schema.

Our implementation given in Fig. 2 has following steps:

Message 1: When a new node requests to join the network, it sends a provisioning request to border router. This message includes SN for identity check and MAC address and time stamp (TS) of node signed by private key of the node for challenge-response check on AC.

Message 2: Since authentication process runs on AC, the responsibility of border router is to forward this request via a secure channel.

Message 3: TS check for replay attack runs on AC.

Message 4: Once the request is approved in terms of TS in AC, second check will be if the node is approved in

terms of credentials. If received SN is in the trust list of AC, the signed message is checked with the public key of node if related MAC address is correct.

Message 5: Successful operation results in sending the requester node public key to border router through a secure channel to approve that the node is in trusted list of AC.

Message 6: Border router adds the node into its trusted list and stores the public key.

Message 7: Border router still does not share any secret in unprotected mode. Instead, encrypts session secret key with requester node's public to protect and sends it to the node therefore any attacker or eavesdropper cannot able to access secret key.

Message 8: Requested node decrypts the message and stores the secret for secure communication.

IV. Implementation and Results

Our proposed schema requires digital signature, verification, and encryption algorithms. Since our proposal is for resource constrained IoT environments, the implementation bottleneck is device resources. To solve this problem, our implementation is based on elliptic curve. We selected, 160-bit Koblitz elliptic curve [2] over F_p with the parameters given standards. Our implementation is configurable for other elliptic curve parameters given in standards.

In order to analyze the tradeoff between implementation burden of security protocols and power consumption, we conducted some experimental tests on our secure IoT network topology. The choice of the ContikiOS has allowed us to use multi-threaded environment, IPv6 and RPL [25] protocols for communication and power analysis environment. Packet transmission, receiving, routing features are analyzed with COOJA. COOJA reports network timing information and simulates lossy environment behavior. In addition, Powertrace [26] power profiling tool are used to monitor power consumption of active, Low power (LPM) modes, RX and TX. Power analysis requires a real hardware information therefore we choose Zolertia experimenter board which is based on msp430f2617 based with CC2420 TRX module.

The key performance indexes for our proposal are defined as implementation efficiency which are code size and speed. In addition, we targeted power efficiency as an important performance metric for our design hence measured our implementation power consumption using Powertrace power profiling tool.

A. Code Size Results

Code size performance of our implementation for a node on MSP430 is given in Table I. Flash memory is crucial since cryptographic algorithms require considerable amount of code size, especially asymmetric algorithms' parameters such as elliptic curve parameters and symmetric algorithms' permutation and combination blocks. Besides that, RAM size is a constraint for IoT nodes specially for elliptic curve multiplication operations.

TABLE I: CODE SIZE INFORMATION FOR NODE (SIZES ARE GIVEN IN DECIMAL)

	Text Size	**Data Size**	**BSS Size**
Noze.Z1	55389 Bytes	358 Bytes	5346 Bytes
Area	Flash	RAM	RAM
Total Size	99208	8192 Bytes	8192 Bytes

B. Power Consumption Results

Powertrace is used for power profiling analysis. It has a linear model which means instantaneous power is the sum of all active power states. In this model, measuring the time during which component m has been in state n has important role since Powertrace tracks system states by using this time difference during which components are in each power state. For this purpose, software stack is modified to report a time difference when a component changes its state.

The profiling tool reports Energest value which is obtained by taking the difference between number of ticks between a time interval and its previous time interval.

TABLE II: PLATFORM PARAMETERS FOR POWER MODEL

Parameter	**Value**	**Description**
CPU Voltage	3V	Between 1.8V to 3.6V
CPU_ACTIVE	10mA	CPU current. Consumption in active mode.
CPU_LPM	0.5uA	CPU current. consumption in standby mode.
RX	8.8mA	TRX module current consumption in receive mode.
TX	17.4mA	TRX module current consumption in transmit mode.
RTIMER_SECOND	32768	Real time clock tick count

The power consumption is calculated by using the bellow formula where current and voltage parameters are the datasheet values of the module in the measured state as given in Table II. RTIMER_SECOND is the clock frequency and Runtime is the time interval in which we perform measurements.

$$Power\_Con = \frac{Energest\_Value.Current.Voltage}{RTIME\_SECOND.Runtime} \quad (2)$$

TABLE III: IMPLEMENTATION RESULTS (TIMING, POWER, AND ENERGY CONSUMPTION)

Operation	**#Times**	**Time**	**Power**	**Energy**
ECC Sign	1	5.73 seconds	32mW	173mJ
ECC Decrypt	1	5.3 seconds	30mW	160mJ

Since Energest value which is reported by Powertrace is number of ticks for a time interval, $\frac{\text{Energest\_Value}}{\text{RTIMER\_SECOND}}$ equals to time spent for reported operation in seconds.

Based on Fig. 3 and Fig. 4 our findings can be seen in ECC sign row of Table III. As observed in Fig. 3, there are some spikes on power consumption. To elaborate this spikes, transmit power is analyzed and it is observed that

although we don't send any packets, due to RPL protocol internals, periodic synchronization packets consume considerable power.

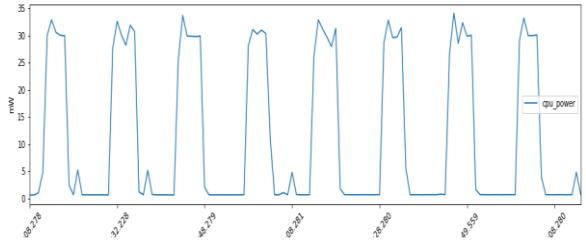

Fig. 3. CPU power consumption during periodic ECC-Sign operation.

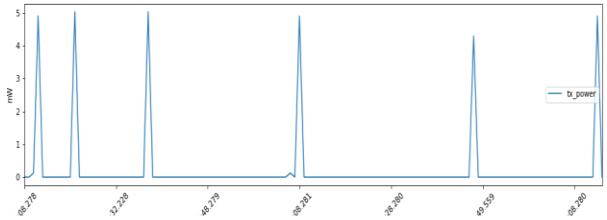

Fig. 4. Transmit power consumption during periodic ECC-Sign operation.

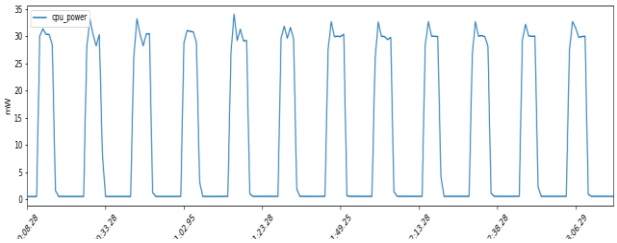

Fig. 5. CPU power consumption for periodic ECC-Sign operation (TX is disabled)

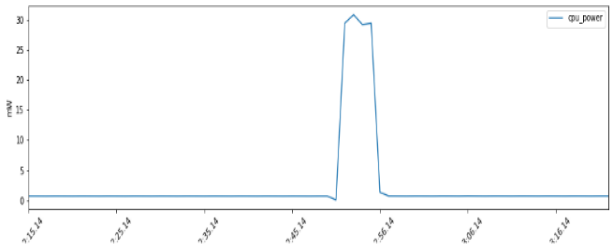

Fig. 6. CPU power consumption for single decryption operation.

To observe power consumption for ECC sign operation more clear, we removed transmission and receive power consumption just for measurements. We disabled CC2420 transmitter module at driver level for this purpose. Disabled transmit module configuration can never be the real use case since due to nature of RPL protocol, periodic network related packets have to be in the network. TX power eliminated plots can be found on Fig. 5.

As given in section III-B, IoT nodes in our schema requires 1 signing and 1 decryption operation. In addition to our results and findings for signing operation given above, decryption results can be found in ECC-decrypt row in Table III. As it is seen on Fig. 6, an IoT nodes consumes approximately 30mW for a single operation and this operation takes 5.3 seconds on our platform.

V. Conclusion and Future Work

In this paper we introduced an end-to end secure IoT node provisioning schema and its power analysis. Our proposed schema does not need any application layer certificate based protocol. Besides that, as we list in section II, some studies rely on dedicated credential server or hardware. On the other hand, our mechanism is based on efficient elliptic curve cryptography without dedicated local credential server. For all we know, our schema differs from previous studies in the literature with these features. Since we don't use any certificate based schema and high level agreement protocol such as Internet Key Exchange(IKE) [27], our proposal is suitable for resource constrained environments. As it is given in Table I our implementation fits into limited RAM and ROM area. In addition, our power consumption results show that, an IoT node consumes 32mW power for single sign operation for step 1 and 30mW for step 8 which are given in section III-B.

A potential future direction would be to implement different asymmetric encryption algorithms and evaluate timing, code size, and power consumption for same schema. A combination of symmetric and asymmetric algorithms can be applied to extend proposed solution as well. In addition, efficient elliptic curve implementations and improvements listed in the literature is another potential research direction.

Conflict of Interest

The authors declare that there is no conflict of interest.

Author Contributions

The first author Ilker Yavuz conducted the research; including implementation, simulation, and data analyzes. The co-author Prof. Berna Ors examined the results and supervised the research. Both authors had approved the final version.

References

[1] M. Sethi, B. Sarikaya, and D. Garcia-Carillo, "Secure IoT Bootstrapping: A Survey," *Internet Engineering Task Force*, 2020.

[2] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, pp. 203–209, 1 1987.

[3] L. Lamport, "Password authentication with insecure communication," *Commun. ACM*, vol. 24, pp. 770–772, 1981.

[4] C. Adams, "Replay attack," in E*ncyclopedia of Cryptography and Security*, H. C. A. van Tilborg and S. Jajodia, Eds., Boston, MA: Springer US, 2011, pp. 1042–1042.

[5] J. Höglund, S. Lindemer, M. Furuhed, and S. Raza, "PKI4IoT: Towards public key infrastructure for the internet of things," *Comput. Secur.*, vol. 89, 2020.

[6] R. Cragie, Y. Ohba, R. S. Moskowitz, Z. Cao, and B. Sarikaya, Security Bootstrapping Solution for Resource-Constrained Devices, 2012.

[7] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz, Extensible Authentication Protocol (EAP), RFC Editor, 2004.
[8] D. Garcia-Carrillo and R. López, EAP-based Authentication Service for CoAP, 2017.
[9] D. Forsberg, B. Patil, H. Tschofenig, and A. Yegin, Protocol for Carrying Authentication for Network Access (PANA), RFC Editor, 2008.
[10] B. Sarikaya, Y. Ohba, Z. Cao, and R. Cragie, "Security bootstrapping of resource-constrained devices," *Internet Engineering Task Force*, 2010.
[11] 3GPP, "Generic Authentication Architecture (GAA)," Technical Specification Group Services and System Aspects, 2016.
[12] D. C. Neuman, S. Hartman, K. Raeburn, and T. Yu, The Kerberos Network Authentication Service (V5), RFC Editor, 2005.
[13] T. Aura and M. Sethi, "Nimble out-of-band authentication for EAP (EAP-NOOB)," *Internet Engineering Task Force*, 2020.
[14] O. Bergmann, S. Gerdes, and C. Bormann, Simple Keys for Simple Smart Objects, 2012.
[15] A. Peltonen, E. Inglés, S. Latvala, D. Garcia-Carrillo, M. Sethi, and T. Aura, "Enterprise security for the internet of things (IoT): Lightweight bootstrapping with EAP-NOOB," *Sensors*, vol. 20, p. 6101, 10 2020.
[16] G. Zorn and D. Harkins, Extensible Authentication Protocol (EAP) Authentication Using Only a Password, RFC Editor, 2010.
[17] S. Cirani, M. Picone, P. Gonizzi, L. Veltri, and G. Ferrari, "IoT-OAS: An oauth-based authorization service architecture for secure services in IoT scenarios," *IEEE Sensors Journal*, vol. 15, no. 2, pp. 1224–1234, 2015.
[18] D. Garcia-Carrillo and R. Marin-Lopez, "Lightweight CoAP-Based bootstrapping service for the internet of things," *Sensors*, vol. 16, p. 358, 2016.
[19] M. Hossain and R. Hasan, "Boot-IoT: A privacy-aware authentication scheme for secure bootstrapping of IoT Nodes," in *Proc. IEEE International Congress on Internet of Things*, 2017.
[20] D. Simon, R. Hurst, and D. B. D. Aboba, The EAP-TLS Authentication Protocol, RFC Editor, 2008.
[21] J. Sanchez-Gomez, D. Garcia-Carrillo, R. Marin-Perez, R. Sanchez-Iborra, and A. F. S. Gomez, "Secure bootstrapping and header compression for IoT constrained networks," in *Global Internet of Things Summit (GIoTS)*, 2020.
[22] A. Ghedini and V. Vasiliev, TLS Certificate Compression, RFC Editor, 2020.
[23] B. Preneel, "New European Schemes for Signature, Integrity and Encryption (NESSIE): A status report," in *Public Key Cryptography*, Berlin, 2002.
[24] . L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, pp. 120–126, 2 1978.
[25] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. P. Vasseur, and R. Alexander, RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks, RFC Editor, 2012.
[26] A. Dunkels, J. Eriksson, N. Finne, and N. Tsiftes, Powertrace: Network-level Power Profiling for Low-power Wireless Networks, 2011.
[27] D. Harkins and D. Carrel, The Internet Key Exchange (IKE), RFC Editor, 1998.

Ilker Yavuz received his B.Sc. and M.Sc. degrees in Electronics & Communication Engineering from Istanbul Technical University (ITU), Turkey, in 2004 and 2008, respectively. He is currently a Ph.D. candidate in the same major in ITU. He has been in the software industry since 2004 and working for AirTies Wireless Networks as a Principal Software Engineer since 2017. His research interests are in areas of cryptology, embedded systems, and wireless sensor networks.

Berna Ors received the Electronics & Communication Engineering degree and the MSc degree in 1995 and 1998, respectively, both from the Istanbul Technical University (ITU), Turkey. She received the Electrical Engineering degree in applied sciences from the Katholieke Universiteit Leuven, Belgium, in 2005. Currently, she is a Professor at ITU. Her main research interests include cryptography, embedded systems, and side-channel attacks.