Zero touch privacy preserving provisioning in an Edge-, Fog, and Cloud environment

Zero-Touch privacy preserving provisioning in an Edge-, Fog-, and Cloud environment

Raphael Schermann*, Simone Bussa‡, Rainer Urian†, Christian Steger*
Email: raphael.schermann@student.tugraz.at,simone.bussa@polito.it, rainer.urian@infineon.com, steger@tugraz.at
*Institute for Technical Informatics, Graz University of Technology, Graz, Austria
†Infineon Technologies AG, Augsburg, Germany
‡Department of Control and Computer Engineering, Politecnico di Torino, Turin, Italy

***Abstract*—IoT device onboarding, especially in the context of an edge-fog-cloud architecture, still has many challenges to solve. FIDO has already specified a zero-touch onboarding process called FIDO Device Onboarding (FDO) specification. In this paper, we present improvements to the FDO specification regarding performance and privacy. For privacy and security reasons, we show how the URL of the Owner Fog can be hidden from the Rendezvous Server. Further, we replaced the EPID protocol with a promising privacy-preserving protocol called AACKA. We also modified the last phase in the FDO protocol to create a performance improvement.**

***Index Terms*—secured zero-trust onboarding, edge-fog-cloud architecture, anonymous authenticated encryption, FDO**

I. Introduction

The number of Internet of Things (IoT) devices is constantly increasing. IDC foresees 42 billion IoT devices by 2025 [1]. The heterogeneity of these devices, each with its own hardware and operating system, needs portable security solutions as a centerpiece.

A particularly critical aspect in the life cycle of these devices is onboarding, i.e., the process through which a device establishes for the first time a trusted connection with the service/platform on which it will then operate. This process is typically done manually: an operator turns on the device, manually installs the credentials, and indicates the device which platform to communicate to. This approach, aside from its expense and time-intensive nature, mandates the presence of skilled and reliable personnel. For all these reasons, industry and academia are trying to automate the process.

Plug&play and zero-touch solutions are being sought, that the device automatically interacting with the platform when switched on, without requiring manual intervention. There exist several concepts for this automatic interaction like zero-touch provisioning [2], [3], a RFC [4], and some academic research [5], [6].

Among the various protocols that have been proposed, the FIDO Device Onboard Specification (FDO) is the most prominent one [7]. The objective of this protocol is threefold: i) authentication of the Device to the platform (also called the *Owner Platform*), ii) authentication of the Owner Platform to the Device (thus completing *mutual trust* between the parties), and iii) provision of some cryptographic material to be used to protect all subsequent communications (i.e., secure channel establishment). Late binding is another central aspect. This feature allows the *Owner*, the person who purchased the Device, to choose the Owner Platform at a later stage in the Device life cycle without being required to specify this information already during Device manufacturing. To achieve this, the *Rendezvous Server* is applied, a third entity that serves as a meeting point between the Device and the platform. On the one hand, the platform contacts the server and stores the URL on which it waits for the Device. On the other hand, the Device, which during manufacturing was configured with the IP address of the server to contact at power on, requests from it the URL of the platform.

This process, however functional, exposes the Device to privacy issues. An attacker listening on the Rendezvous Server can link the Device to the URL of the platform to which it will be connected.

This paper aims to propose a solution to overcome this privacy limitation while also increasing the performance of the protocol. In particular, we try to optimize the FDO protocol, especially regarding an edge-fog-cloud architecture. To this purpose, we propose a revised version that introduces privacy for the Device against the Rendezvous Server. Specifically, the URL that is sent to Rendezvous Server is encrypted in an anonymous and authenticated way, which allows the Device to recover the original information and, at the same time, prevents the Rendezvous Server from tracing the final destination of the Device and does not expose the owner's IP as a target of potential attacks.

For the encryption, we introduce a recently proposed encryption scheme called Anonymous Authenticated Credential Key Agreement (AACKA), [8], that adds an additional benefit in terms of performance. In the new concept proposed in this paper, we extend the encrypted payload containing the URL with a pre-shared key that can be used to optimize the secure channel establishment phase. This should make it easier for the device to communicate with the owner's platform (e.g., Owner Fog) through a simple TLS with PSK. Moreover, applying this encryption scheme also allows implicit mutual authentication, further improving the protocol's performance.

The remainder of this paper is structured as follows. We first start with a Background & related work section. This section is followed by a more detailed description of FIDO Device Onboard (FDO) specification. Our main contribution starts in section IV where we cover the integration of the FDO in

an Edge-,Fog-,Cloud- architecture and show their limitations. In V we propose our developed architecture and show their benefits and the related security analysis is done in section VI. The paper ends with an Evaluation (VII) and a conclusion and future work section (VIII).

II. Background & related work

This section explains the necessary background information and related work that is essential to understand the remainder of the paper.

A. Cloud computing

Cloud computing [9] is a centralized computing paradigm that offers on-demand computational resources and storage. An IoT Cloud is a platform used to manage devices with restricted computational capabilities, providing them with all the underlying infrastructure for their real-time operations and processing. This is achieved by exchanging data over the network. It is inherently scalable, cost-effective, and efficient. It has numerous advantages: it provides devices with storage for the data they collect and computational power to analyze it, allows users to access this data at any time from anywhere using any device with an Internet connection, makes it easier to manage large numbers of devices without having to design the entire infrastructure from scratch.

B. Fog computing

Fog computing, introduced by Cisco [10], extends cloud computing and positions itself close to the edge of the network, near the physical IoT device. It is a decentralized/distributed computing model which performs most of the data collecting and processing locally, near the edge. Then it sends only the pre-processed data to the cloud for more complex operations. This way, a smaller amount of data travels on the network, limiting bandwidth consumption and minimizing response time. This makes this paradigm ideal for real-time and low-latency applications.

C. Zero touch onboarding

Zero-touch onboarding is an automated process that enables organizations to quickly and securely onboard devices to a Fog/Cloud platform, eliminating the need for manual configuration, reducing the cost and complexity and increasing security.

The onboarding process nowadays is very complex. As described in [2], part of this complexity is due to the wide variety of existing IoT devices, each with its own hardware and software. Most of the devices have no display and different devices may have different connectivity (wired/wireless). The onboarding process is very often carried out manually by an operator who must have qualified skills and be trusted by the owner of the device. Therefore, an automatic, hardware/software-independent solution is more than ever necessary.

The first Zero touch protocols that appeared on the market were all proprietary: each company implemented its own protocol and this made the interoperability with different platforms difficult. Devices were configured to communicate only with platforms of a specific vendor already in the manufacturing phase, thus violating the principle of late binding. The need to decide later on which platform to connect the device and also overcome the other mentioned limitations is driving industry and academia to standardize the onboarding process. Several attempts for zero-touch provisioning have been published over the years, [3], [4], [5], [5]. What appears, to the best of our knowledge, FIDO FDO stands out as the most promising candidate at present [2].

D. Privacy-preserving protocols as a candidate for secured zero touch onboarding

This subsection is divided into a high overview of the Enhanced Privacy IDentification (EPID) scheme and a newly developed scheme called Anonymous Authenticated Credential based Key Agreement (AACKA). We explain the AACKA protocol in more detail because this one is later used in our proposed model in section V.

1) Enhanced Privacy IDentification (EPID): The EPID protocol is a cryptographic scheme developed by Intel Corporation [11]–[13] to address the need for secure device authentication while preserving user privacy. It achieves this by employing a group signature scheme, which allows multiple devices to generate signatures on behalf of a group, making it difficult to trace a signature back to a specific device. The signature is a BBS+ signature scheme [14] and a variant of the Boneh-Boyen signature scheme [15]. In particular, it addresses two problems with PKI solutions. First, anonymity and second, membership revocation.

The EPID security ecosystem encompasses three distinct roles. Firstly, the Issuer acts as the authority group responsible for assigning EPID Group IDs and Keys to individual platforms. The Issuer manages group membership and maintains up-to-date versions of all revocation lists. By generating a new private key for the group, the Issuer produces a single group public key and multiple EPID member private keys as needed, each paired with the group public key. The Member role represents an end device and constitutes one member among a group of several members who share the same level of access. Members utilize the EPID credentials provided by the Issuer to authenticate their identities and participate in the group. Lastly, the Verifier role assumes the role of a gatekeeper within the EPID ecosystem. Verifiers are responsible for verifying EPID signatures generated by platforms and ensuring their alignment with the correct group. Basically, the EPID scheme consists of four algorithms:

1) Setup or key generation: The protocol begins with a trusted authority generating a group public key and a corresponding private key. The group public key is shared with all devices in the EPID group, while the private key remains securely held by the authority.
2) Join: When a device wishes to join the EPID group, it submits its public key to the trusted authority. The authority verifies the authenticity of the device through

rigorous procedures and issues a membership credential upon successful validation. This credential serves as proof of the device's membership in the EPID group.

3) Sign: When a device needs to prove its identity or integrity in a particular scenario, it utilizes its private key and the group public key to generate an EPID signature. The signature includes a randomized component, making it difficult to link the signature back to the specific device that generated it. This randomization enhances privacy protection by preventing the potential tracking and identification of individual devices.
4) Verify: A remote party (verifier) receives the EPID signature and employs the group public key to verify its validity. The verifier can ascertain whether the signature originates from a member of the EPID group without having knowledge of the specific device that generated it. This capability allows the verifier to establish trust and verify the authenticity of devices without compromising user privacy.

EPID is currently used in the FDO specification by FIDO.

2) Anonymous Authenticated Credential based Key Agreement (AACKA): Schermann et al. [8] presented a novel building block called Anonymous Authenticated Credential Key Agreement (AACKA). This building block can be used to enable Anonymous Authenticated Credential based Encryption (AACE). Further, the AACKA protocol can be enhanced by pseudonyms (Pseudonymous Authenticated Credential Key Agreement - PACKA) that gives the user the opportunity to decide whether to be anonymous or linkable/pseudonymous to a communicating entity. The AACKA protocol is designed to enable anonymous or pseudonymous encryption to resource-constrained devices and uses only basic cryptography. It combines Camenisch-Lysyaskaya (CL) credentials [16] with an Elliptic Curve Diffie-Hellman (ECDH) key agreement. The AACKA consists of three roles and three phases. The basic roles are split into an issuer $\mathcal{I}$, a Device $\mathcal{D}$, and a Service Provider $\mathcal{SP}$. For performance reasons and offline-package provision a Privacy-Proxy $\mathcal{PP}$ is used. While the phases are divided into SETUP, JOIN, AACKA, and AACE. In the end, the service provider and the device have an anonymous and authenticated encrypted channel established based on CL credentials.

In detail, AACKA can be seen as an ephemeral/static Diffie-Hellman key agreement with an anonymous and authenticated extension between a Device $\mathcal{D}$ and a Service-Provider $\mathcal{SP}$, e.g., a Cloud node (Owner Cloud). Instead of a classical X.509 certificate, the AACKA uses a Camenisch-Lysyanskaya (CL) credential A, B, C, D.

A CL-credential consists of four elliptic curve points $A, B, C,$ and D. It certifies a private key that is associated with a private key $D = K_{private} * B$. The crucial point for anonymity is that a scalar multiple, $R, S, T, W \rightarrow l * A, l * B, l * C, l * D$, of such a CL-credential can also be verified as a valid CL-credential. This does not reveal the actual points of the CL-credential.

The protocol consists of four phases. A Setup, Join, AACKA, and AACE phase. In the Setup phase the Issuer $\mathcal{I}$ generates a public and private key pairs (X,Y;x,y) from the systems parameter. In the Join phase, the Device $\mathcal{D}$ joins to $\mathcal{I}$ with the help of his private AACKA key f and gets a CL-credential from $\mathcal{I}$. In the AACKA phase a key agreement between $\mathcal{D}$ and the Service-Provider $\mathcal{SP}$ with the help of the CL-credential happens. In the end both parties have derived the same shared secret. This secret can then be used in the AACE phase to derive keys for the ECIES protocol. In our paper, it is important to understand the AACKA-ECIES phase. The role of $\mathcal{I}$ and randomization of the CL-Credential can be taken over by the manufacturer. Further, the Privacy Proxy can be skipped because it is only useful for resource-limited devices. $\mathcal{D}$ is in possession of a private AACKA key f and a randomized CL-credential A, B, C, D. This CL-credential must be randomized (R, S, T, W) by $\mathcal{D}$ to infiltrate anonymity. This randomized credential is then forwarded to the Service Provider $\mathcal{SP}$. $\mathcal{SP}$ performs a bilinear mapping to verify if the current randomized CL-credential is a valid member. With the help of R, S, T, W $\mathcal{SP}$ derives in the end symmetric keys and creates an encrypted package with a Tag. The ephemeral key U together with the encrypted package and the Tag is sent to $\mathcal{D}$. $\mathcal{D}$ can now create the same keys as $\mathcal{SP}$ with the help of the private AACKA key f and U to first verify the tag and then decrypt the package.

III. FIDO Device Onboard Specification

The FIDO Device Onboarding (FDO) is a protocol specified by the FIDO Alliance in an early version in 2020, [2]. Unlike other specifications from the same Alliance, it is a protocol specific for securely onboarding a device on a cloud/edge platform. The Onboarding process is zero-touch, plug&play and hardware independent. As partially anticipated in the Introduction, the protocol consists of different actors. *Device*: the "thing" to onboard. *Manufacturer*: the company that produced the Device. *Owner*: the person/company which currently owns the Device. *Owner Platform*: the platform on which the Device will be onboarded. *Rendezvous server*: the server that indicates the correct Owner Platform for the Device. From when the Device is manufactured to when it is turned on for the first time, the Owner can change several times (e.g., manufacturer, distributor, reseller, etc.). To track these changes of ownership, the *Ownership Voucher* (OV) has been introduced. Fig. 1 shows the OV structure. It is a digital chained certificate characterized by multiple entries, each new one added every time the Device changes ownership. Initially, the ownership of the Device belongs to the manufacturer. Therefore the first entry of the OV certifies the public key of the manufacturer. The Device trusts this first key and internally has stored the corresponding private key. All transfers of ownership can take place with the Device switched off. Each time the Device is sold, a new entry is added to the OV. This new entry, N_i contains the public key of the new Owner. It is certified by the previous Owner, who signs it with the private key corresponding to the public key contained in the entry N_{i-1} of

the new OV. Concretely, the signature affixed by the previous Owner is used to certify the transfer of ownership of the Device. Among the other fields, the OV also contains the GUID of the Device and the IP address of the Rendezvous Server. The protocol is divided into 4 phases:

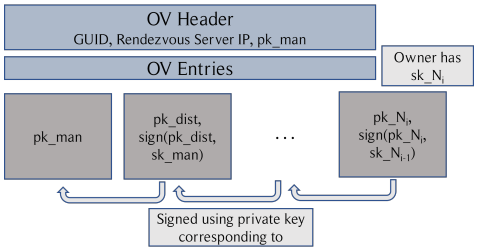

Fig. 1. Ownership Voucher structure

1) Device Initialize (DI) protocol: takes place during manufacturing in a safe environment. The GUID, the private key from the manufacturer, and the URL of the rendezvous server are installed inside the Device. The OV is then created, which contains the same values and the public key.
2) Transfer Ownership protocol 0 (TO0): the Owner registers with the Rendezvous server. After receiving the OV and taking ownership of the Device, the Owner extracts the URL of the rendezvous server from the OV and registers on it the IP address of the Owner Platform. The Owner has the private key corresponding to the public key in the last entry of the OV. Therefore it uses the OV to authenticate to the Server. At the end of the protocol, the IP address of the Owner Platform is saved on the Rendezvous Server while the Owner Platform is waiting for the device connection at the specified URL. Notice that, to authenticate the Owner, the Server must trust at least one of the keys present in the OV. Typically, there is trust between the Server and the manufacturer. Since it is the manufacturer who decides which Server its devices have to use, it is conceivable that there is an agreement between the two parties and the Server trusts the manufacturer's public key in the first entry of the OV.
3) Transfer Ownership protocol 1 (TO1): the Device, just turned on, contacts the Rendezvous Server to get the IP address of the Owner Platform and the corresponding Ownership Voucher. It uses its GUID and the private signing key to authenticate to the Server.
4) Transfer Ownership protocol 2 (TO2): Onboarding. It is the most important phase of the protocol between the Device and the Owner in which mutual authentication takes place. The Owner authenticates to the Device by providing to be the actual Owner of the OV. The Device authenticates to the Owner by using the private key received from the manufacturer, whose corresponding public key is contained in the first entry of the OV. Once mutual authentication is completed, a key exchange protocol is performed to create a secure channel on which to communicate.

More details about the protocol can be found in [7].

IV. Integration of classic FDO in Fog computing and its limitations

This subsection introduces our interpretation of integrating a Fog node (Owner Fog) into the current FDO specification. In this case, the role of the Owner is split into a Cloud-Owner and at least one Fog-Owner, which is the final node on which the device will be installed. The difference with respect to the protocol described above is that, in this case, the Fog node (and not the Cloud Platform) must be the last Owner of the device. For this, the ownership voucher has to be extended with a new entry and transferred to the corresponding Owner Fog.

Figure 2 shows such an architecture as it would work with the actual FIDO device onboarding specification (FDO). DI remains the same as it is specified above (Step 1). Then, the Cloud takes ownership of the OV (Step 2). TO0 protocol is executed between the Rendezvous Server and the Cloud, the actual owner of the OV (Step 3). The IP address registered on the Rendezvous Server, in this case, is the address of the Owner Fog. The Owner Cloud than extends the OV, adding an entry for the public key of the Owner Fog and sends it to the Owner Fog (Step 3). T01 remains the same (Step 4). TO2 is executed between the Fog-node and the device (Step 5-6).

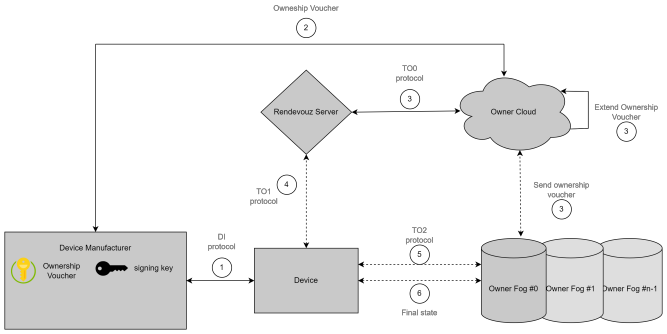

Fig. 2. Architecture concept to be FDO compliant

In this FDO conform integration, we figured out following disadvantages:

- The Rendezvous Server sees all target IP addresses of the corresponding Fog nodes (Owner Fogs) in plain. This totally impacts the privacy of the Owner Fog, allowing an attacker who is listening on the Rendezvous Server to know the IP address of the final node and also, most likely, what task it is intended to do, having information on the characteristics of the device to be onboarded. Moreover, following the entry chain present in the OV, a possible attacker can easily discover who is the actual Owner Cloud of the corresponding Owner Fog, and this can be another problem affecting its privacy.
- The private key stored inside the device and used for device authentication can easily lead to linkability issues

in case of subsequent invocations of FDO. The FDO protocol foresees that in case of resale of the device, all the cryptographic material and device identifiers are substituted, with the exception of the private key used for attestation. Using this key makes it possible to correlate past owner and future owner of the device. A possible countermeasure addressed in [2] to avoid this is by using Intel EPID protocol, which allows the device to use a group signature to authenticate and never expose its private/public attestation key. However, this is a quite complex and heavy protocol, which involves group signatures and revocation lists and still has some disadvantages.

- The owner has to extend the ownership voucher with a public key of the Owner Fog. This extended voucher has to be transmitted to the Owner Fog. This increases the complexity and time for the owner verification and therefore makes it more expensive.
- In step TO2, the Owner Fog has to perform asymmetric cryptography to authenticate the device. It would be more convenient to use standard TLS with pre-shared key simply.

The next sections describes our proposed architecture and provides a solution to the listed drawbacks. This proposed model should fit with minor changes into the classic FDO scheme.

V. Proposed model

In the following, we propose a privacy-preserving zero touch provisioning mechanism that is especially suited in the context of a Fog environment. Our solution provides anonymity by hiding the Owner Fog URL from the Rendezvous server. This is done by encrypting the URL of the Owner Fog to which the device should be connected with a novel privacy-preserving encryption protocol AACKA-ECIES [8]. In addition to anonymity, we show how AACKA-ECIES can be exploited to improve the performance of the onboarding process in a Fog environment. The basic idea is as follows. Besides the URL, the Owner Cloud also encrypts a symmetric key for the device. This symmetric key is also shared with the Owner Fog and can subsequently be used to form a TLS with pre-shared key connection between the device and the Owner Fog. This relieves the Owner Fog from implementing the complex EPID and ECDSA protocols as used by the standard FIDO FDO protocol.

We now describe our novel concept in detail and its relation to the FIDO FDO protocol [7]. Figure 3 shows the graphic representation of our concept.

Precondition

Each Owner Fog has established a secured channel with the Owner Cloud. Over this channel, the Owner Cloud can share a symmetric key K_{fog}. The details of how this channel is established and the key is shared are outside the scope of the onboarding protocol.

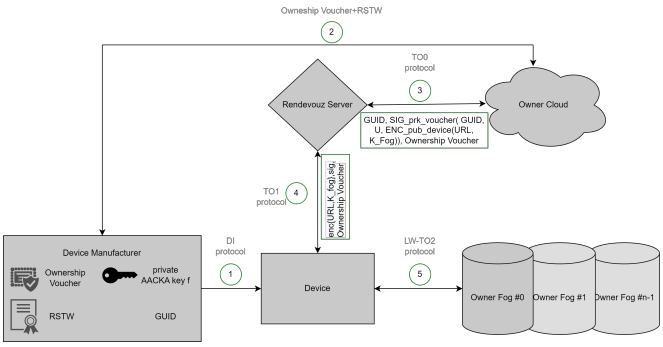

Fig. 3. Our proposed zero touch onboarding architecture

DI Protocol (1)

As in the FIDO FDO protocol, the manufacturer provisions the device with a GUID and the URL of the Rendezvous Server. However, in our modification, the device will be provisioned with an AACKA-ECIES decryption key and an associated CL-certificate (R, S, T, W) instead of an EPID or ECDSA signature key as used in the original FIDO FDO. The manufacturer constructs the ownership voucher, which contains a certificate of the signature public key of the Owner Cloud. The device will also be personalized with a public key which is used to verify the certificate chain of the Ownership Voucher.

Ownership Voucher (2)

The device manufacturer sends the ownership voucher and the AACKA CL-credentials (R, S, T, W) to the Owner Cloud. The Owner Cloud verifies the CL-certificate (R, S, T, W) as described in [8].

TO0 protocol (3)

In the TO0 protocol (3), the Owner Cloud derives a key $k_{fog,device} := KDF(K_{fog}, GUID_{device})$ by using some arbitrary key derivation function KDF. This key will be used in the TO2 protocol as a pre-shared key between the Owner Fog and the Device. The Owner Cloud has to construct the encrypted rendezvous blob. This blob consists of the URL from the Owner Fog URL_{fog} and the shared key $k_{fog,device}$. The Owner Cloud encrypts the corresponding data for the Owner Fog (URL, K_{Fog}, *additional data*) with the AACKA-ECIES encryption key (S, W) of the Device as described in [8]. Then the Owner Cloud signs the encrypted blob together with the GUID and auxiliary data used for the encryption (e.g. the ephemeral key U for the AACKA-ECIES encryption) with his private signing key associated to his Ownership Voucher. The Owner Cloud sends the GUID, the rendezvous blob ($SIG_{prk,voucher}(GUID, U, ENC_{pub,device}(URL, K_{Fog},$ *additional data*)), and the Ownership Voucher to the Rendezvous Server.

The encryption prevents an attacker and even the Rendezvous Server from discovering the final destination (Owner Fog URL) of the device. In addition, encryption and signing

of the rendezvous blob prevent the Rendezvous Server from forging the URL of the Owner Fog. Note, the encryption approach also allows the Owner Cloud to directly inject additional confidential information in the encrypted blob, e.g., a firmware update or other symmetric/private keys in addition to $k_{fog,device}$.

TO1 protocol (4)

Our TO1 protocol is defined as follows. After the registration of the device to the Rendezvous Server, the device fetches the Ownership Voucher and the rendezvous blob from the Server. The device performs a signature validation of the rendezvous blob by using the public key of the Ownership Voucher and decrypts the encrypted blob with its private AACKA key. Note that the device attestation step is not necessary. This is done implicitly because only a correct device can decrypt the rendezvous blob.

TO2 protocol (5)

Basically, our proposed T02 protocol is the right part of Figure 9 "Transfer Ownership Protocol 2 (TO2)" [7]. The left side of the original T02 protocol has already been performed in our modified T01 protocol.

Our modification on the right-hand side of T02 is only in the establishment of a secure channel. This will be done by exploiting the symmetric key K_{fog}. The device contacts the Fog node (Owner Fog) and sends $GUID_{device}$. The Owner Fog can now derive the key $k_{fog,device} := KDF(K_{fog}, GUID_{device})$. The Owner Fog and the device have now established a shared symmetric key. This key can then be used to establish a secured channel over TLS with a pre-shared key. The remainder of the onboarding protocol provisioning between the Owner Fog and the device can be done with this channel. Note that the signing operation with the Owner2 key can be performed and sent to the Owner Fog by the Owner Cloud.

Figure 4 shows the more detailed protocol flow. In grey are the messages that are involved in the FIDO FDO standard and replaced by our steps. Please note that to simplify the presentation, we have assumed that the ownership voucher has only one entry, i.e., the device manufacturer.

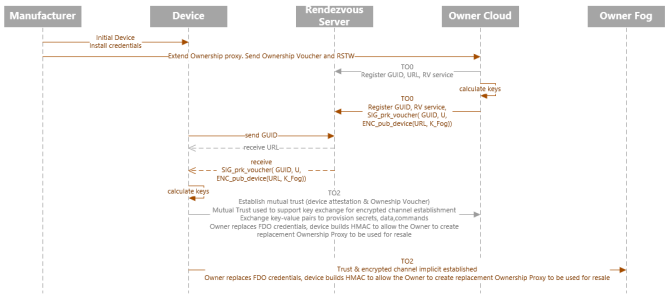

Fig. 4. Protocol flow

Summarized Benefits

At the end of this section, we want to summarize the benefits that our concepts generate:

- URL is hidden from the Rendezvous Server by encryption with AACKA-ECIES. This improves the privacy and supports unlinkability.
- Faster onboarding because the left side of the original FDO T02 protocol can be skipped. This means there is also no further need for the complex mutual authentication that is required there.
- The Owner Fog does not need the capability to perform asymmetric cryptography. Only TLS with PSK is needed.
- Complex EPID protocol for anonymity is replaced by the simpler AACKA protocol.

VI. Security Analysis

The protocol described above was verified using Proverif [17], a widely used automatic tool that performs formal verification. Formal verification, among the various existing static analysis methods, allows us to reach high guarantees of the security of an analyzed protocol by building a formal (mathematically rigorous) model of it. Proverif takes as input the abstract model of the protocol and some security properties that are expected to be satisfied and automatically verifies that no attacks are possible against the system under certain modeling assumptions. The verification is reliable but needs to be completed: the tool can find false attacks, but whether it proves the correctness of a property, the property is guaranteed to hold.

A. Component model

Even the model for the formal verification is made up of several actors: the Device to be onboarded, the Owner Cloud, the Owner Fog, the Manufacturer and the Rendezvous Server. Each of these entities can execute different instances of the protocol. The Server can accept different connections from different Owners and Devices. The Owner can onboard multiple Devices and control a number of Fog nodes.

B. Channel and Attack model

All the channels on which the various entities communicate are public, with the exception of the channel between Device and manufacturer (i.e., the one used for the DI protocol), which is private since the standard assumes it is a safe environment in the manufacturer's factory. The attacker in this scenario can act as a traditional Dolev-Yao attacker: it can read, delete, replay, and modify messages if they are not protected adequately. We also tested a second scenario in which the attacker controls a second valid device (with valid credentials issued by the manufacturer). This was done to give the attacker the possibility to participate in all phases of the protocol and interact with the entities also as an "honest actor" by using this second device. Therefore, it can receive a valid OV for this second device from the manufacturer, correctly authenticate to the Rendezvous Server, and store its IP on it. The main idea of this second scenario is to check if exists a strict association

between the device and its credentials/OV. The attacker should not be able to use the second device's valid credentials/OV to impersonate the owner of the first (honest) device.

C. Security properties

We used Proverif to verify *confidentiality* and *authentication* properties in our protocol. In particular:

- *Secrecy of messages exchanged between the device and the Owner Fog.* At the end of the TO2 protocol, the device and the Owner Fog share a $k_{fogdevice}$ key, used as a pre-shared key to establish a TLS connection. Subsequent messages encrypted using this key must be unreadable by the attacker. Implicitly with this property, we are also proving the secrecy of $k_{fogdevice}$. This was modeled in Proverif as a Reachability property.
- *Authentication of the Owner to the Device.* If the device believes it has received a valid encrypted blob signed by the Owner, then the Owner must have signed and sent that blob (because it is the only entity in possession of the private key necessary for the signature). This proves that there can be no such thing as a man in the middle attacker impersonating the Owner. This was modeled in Proverif as an Injective Correspondence Assertion.
- *Authentication of the device to the Owner Fog.* If the Owner Fog believes it has received a valid message encrypted with $k_{fogdevice}$ from the device, then the device must have sent it (because it is the only entity able to extract that key from the encrypted blob). This prevents false malicious devices from interacting with the Owner Fog. This was modeled in Proverif as a Simple Correspondence Assertion.

Proverif proved that the described properties hold in all scenarios. Source files containing the Proverif model and the complete list of properties that were verified can be found here[1].

VII. Evaluation

In this section, we do a comparison of the current FDO specification against our concept. Note that we do not show a performance evaluation and, therefore, no dedicated metrics. An implementation of this concept is planned together with a detailed performance analysis.

We expect that our concept will consume more time in the TO0 and TO1 protocols because we add additional encryption. The TO2, on the other hand, is significantly improved because the whole complex mutual authentication is implicitly done in the TO1 protocol.

In Detail, Table I shows the overview of the main differences in an Edge-,Fog-,and Cloud architecture. In the DI protocol, we replaced the signing key with an AACKA encryption key and introduced CL-credential to enable anonymous authenticated encryption.

Our TO0 protocol adds additional authentication encryption with AACKA-ECIES. This improves privacy because the URL does not appear in plaintext.

[1]https://github.com/netgroup-polito/verification-zero-touch-provisioning

In our TO1 protocol, we substituted signing with encryption. Only the device with the correct private AACKA key can decrypt the package received from the Owner Cloud and this leads to implicit authentication.

The TO2 in our approach does not need an asymmetric cryptography functionality in the Owner Fogs. It works with symmetric cryptography (TLS with PSK). Further, the expensive mutual authentication part can be skipped because it has already been performed implicitly in the TO1 protocol.

TABLE I
COMPARISON OF THE PROTOCOLS

Protocol	**FDO**	**Our approach**
DI	Signing key	Encryption key
TO0	GUID,URL + extend and send Ownership Voucher to fog	GUID,signed & encrypted packet, Ownership Voucher
TO1	plaintext URL, explicit authentication	encrypted URL, implicit authentication
TO2 (Part1)	Mutual authentication and key exchange based on asym. cryptography	implicit authentication by step TO0, No asymmetric cryptography
TO2 (Part2)	exchange keys to establish a secured channel	TLS with PSK channel

Figure 5 shows the generic TO2 protocol. In the FDO approach, the whole chain has to be processed (Part1+Part2). While in our approach, due to the TO0 and TO1 modifications, only Part2 has to be executed. This is expected to generate a significant performance improvement.

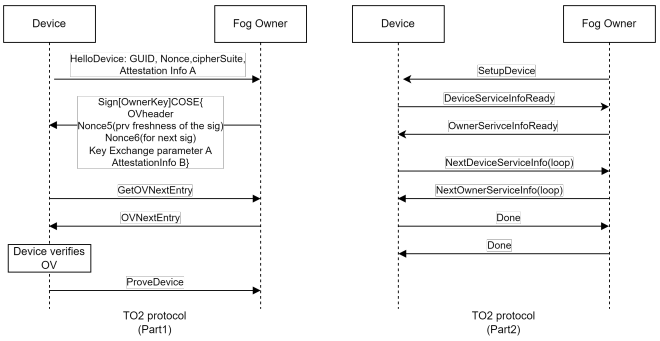

Fig. 5. TO2 comparison (based on [2])

For privacy, we replaced the complex EPID protocol based on anonymous signatures with a newly presented anonymous authenticated encryption scheme (AACKA-ECIES). This is expected to improve the performance additionally.

VIII. Conclusion and Future Work

In this paper, we presented (i) how the current FDO specification fits into an Edge-, Fog-, Cloud architecture and what are these limitations there (ii) we proposed a new concept to overcome the limitations.

Further, we evaluated the security of our concept with Proverif. We showed that our concept holds the following security properties: (i) Secrecy of messages exchanged between the device and the Owner Fog (ii) Authentication of the Owner to the Device (iii) Authentication of the device to the Owner Fog.

For future work, several parts can be taken into consolidation. First, we are planning to do an implementation of the proposed solution to extract detailed performance values. Second, due to lack of space we could not go into privacy-based revocation for our approach, i.e., the Revocation list for the AACKA protocol. Also, one interesting point would be to transform the concept into post-quantum resilient cryptography.

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No. 871403. Furthermore, this project has partly received funding from the ECSEL Joint Undertaking, which funded the ADACORSA project under the grant agreement number 876019. ADACORSA is funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program ICT of the Future between May 2020 and October 2023 (grant number 877585).

REFERENCES

[1] International Data Corporation, "IoT Growth Demands Rethink of Long-Term Storage Strategies, says IDC," https://www.idc.com/getdoc.jsp?containerId=prAP46737220, 2020, [Online; accessed 27-August-2021].

[2] G. Cooper and B. Behm and A. Chakraborty and H. Kommalapati and G. Mandyam and H. Tschofenig, "FIDO Device Onboard Specification," https://fidoalliance.org/specs/FDO/FIDO-Device-Onboard-RD-v1.0-20201202.html, 2020, [Online; accessed 16-February-2023].

[3] Intel. Zero-touch provisioning for edge devicesand software-defined networks. [Online; accessed 10-May-2023]. [Online]. Available: https://networkbuilders.intel.com/docs/networkbuilders/zero-touch-provisioning-for-edge-devices-and-software-defined-networks.pdf

[4] K. Watsen, M. Abrahamsson, and I. Farrer, "Secure Zero Touch Provisioning (SZTP)," RFC 8572. [Online]. Available: https://www.rfc-editor.org/info/rfc8572

[5] S. Maksuti, A. Bicaku, M. Zsilak, I. Ivkic, B. Peceli, G. Singler, K. Kovacs, M. Tauber, and J. Delsing, "Automated and secure onboarding for system of systems," vol. 9, pp. 111 095–111 113.

[6] M. Liyanage, Q.-V. Pham, K. Dev, S. Bhattacharya, P. K. R. Maddikunta, T. R. Gadekallu, and G. Yenduri, "A survey on zero touch network and service management (zsm) for 5g and beyond networks," *Journal of Network and Computer Applications*, vol. 203, p. 103362, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804522000297

[7] G. Cooper and B. Behm and A. Chakraborty and H. Kommalapati and G. Mandyam and H. Tschofenig, "FIDO Device Onboard Specification 1.1," https://fidoalliance.org/specs/FDO/FIDO-Device-Onboard-RD-v1.1-20211214/FIDO-device-onboard-spec-v1.1-rd-20211214.pdf, 2021, [Online; accessed 16-February-2023].

[8] R. Schermann, R. Urian, R. Toegl, H. Bock, and C. Steger, "Enabling anonymous authenticated encryption with a novel anonymous authenticated credential key agreement (aacka)," in *2022 IEEE 21st International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. Wuhan, China: IEEE Computer Society, oct 2022, pp. 646–655. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TrustCom56396.2022.00093

[9] N. Antonopoulos and L. Gillam, *Cloud Computing - Principles, Systems and Applications*. London, UK: Springer London, 2012.

[10] Cisco, "Cisco Fog Computing Solutions: Unleash the Power of the Internet of Things," https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-solutions.pdf, 2015, [Online; accessed 16-June-2023].

[11] Intel Coperation, "A Cost-Effective Foundation for End-to-End IoT Security," https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/intel-epid-white-paper.pdf, 2016, [Online; accessed 27-August-2021].

[12] M. Chandler, "Intel Enhanced Privacy ID (EPID) Security Technology," https://www.intel.com/content/www/us/en/developer/articles/technical/intel-enhanced-privacy-id-epid-security-technology.html, 2017, [Online; accessed 27-August-2021].

[13] E. Brickell and J. Li, "Enhanced privacy id from bilinear pairing for hardware authentication and attestation," in *2010 IEEE Second International Conference on Social Computing*, 2010, pp. 768–775.

[14] M. H. Au, W. Susilo, and Y. Mu, "Constant-size dynamic k-TAA," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006, pp. 111–125.

[15] D. Boneh and X. Boyen, "Short signatures without random oracles," in *Advances in Cryptology - EUROCRYPT 2004*. Springer Berlin Heidelberg, 2004, pp. 56–73.

[16] J. Camenisch and A. Lysyanskaya, "Signature schemes and anonymous credentials from bilinear maps," in *Advances in Cryptology – CRYPTO 2004*, M. Franklin, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 56–72.

[17] B. Blanchet and B. Smyth, "Proverif 1.85: Automatic cryptographic protocol verifier, user manual and tutorial," 04 2011.