Bartsch *et al. J Surveill Secur Saf* 2023;4:155-79
DOI: 10.20517/jsss.2023.19

Journal of Surveillance, Security and Safety

Original Article

Open Access

Zero knowledge registration of PKI authentication for symbiotic security in FIDO IoT

Witali Bartsch[1], Owen Millwood[2], Elif Bilge Kavun[3]

[1]Security Solutions, WIZnet Germany GmbH, Kirchgasse 5, Kleinniedesheim 67259, Germany.
[2]Department of Computer Science, The University of Sheffield, Sheffield S10 2TN, UK.
[3]Faculty of Computer Science and Mathematics, University of Passau, Passau 94032, Germany.

Correspondence to: Witali Bartsch, Security Solutions, WIZnet Germany GmbH, Kirchgasse 5, Kleinniedesheim 67259, Germany. E-mail: witali.bartsch@wiznet.eu

How to cite this article: Bartsch W, Millwood O, Kavun EB. Zero knowledge registration of PKI authentication for symbiotic security in FIDO IoT. *J Surveill Secur Saf* 2023;4:155-79. http://dx.doi.org/10.20517/jsss.2023.19

Received: 20 Jun 2023 **First Decision:** 8 Oct 2023 **Revised:** 29 Nov 2023 **Accepted:** 4 Dec 2023 **Published:** 15 Dec 2023

Academic Editor: Josef Pieprzyk **Copy Editor:** Pei-Yun Wang **Production Editor:** Pei-Yun Wang

Abstract

In this paper, we critically discourse on the current Fast IDentity Online specification, its underlying security layer and the resulting misconceptions or even weaknesses. We juxtapose the presented view against some of the fundamental and long-standing public key infrastructure problems such as the initial enrolment of identities and their keys (e.g., X.509v3 certificates). We then observe a novel approach of zero-knowledge initial enrolment of resource-constrained Internet of Things devices and investigate how its idea of symbiotic security, i.e., a consistent and consolidated cooperative design which involves all architectural building blocks such as software, hardware, networks, and processes, can contribute to the new Fast IDentity Online Internet of Things specification in the making. Finally, we suggest by way of an outlook or future work how this already modular security-by-design mindset can be further enhanced through a dynamic management system to keep cryptographic keys in volatile memory only. The idea is also based on a symbiotic connection between hardware and software to create a number of added benefits from the security perspective.

Keywords: FIDO IoT, secure IoT, symbiotic key management system, symbiotic security, initial enrolment, secure attestation

1. INTRODUCTION

The generation of solid passwords has long been known to be a non-trivial issue, especially if the process involves human interaction. Users tend to have their own perception of good passwords, and even if influenced accordingly or presented with better password generation schemes such as those based on one-time passwords or tokens[1], they still find ways to misuse them for the sake of comfort and simplicity. To overcome those limitations or impose better control structures, certain technical implementations have been suggested. These implementations use software and probabilistic methods to ascertain a high-quality level of the resulting passwords[2]. Even with the most recent regulatory recommendations in mind such as those published by the German Federal Office for Information Security [Bundesamt für Sicherheit in der Informationstechnik (BSI)] on the handling of passwords securely[3], which now even allow for partial password documentation (provided a minimum of complexity or length respectively), device-assisted two-factor authentication is still the general direction to go.

At this point, we found ourselves at a crossroads: either we accept users as agents selecting passwords, which is obviously error-prone and still requires considerable education and discipline, or we start using password management tools to outsource the tedious but critical task to machines. The resulting choice in favour of the latter seems intuitively correct given the fact that the technical approach not only solves the problem of weak passwords but also anticipates a number of subsequent issues such as password reuse or revocation of login credentials. The price for this evolution is a higher complexity entailed by the use of sophisticated cryptographic protocols that, in turn, also require a distributed or centralised infrastructure[1]. However, the design rationale behind any such security scheme may also be somewhat misguided, as we see in the subsequent discussion of a popular standard in the next sections of this paper. Then again, a combination of security specifications that, on the one hand, target machine-to-machine communication, which is essential to the Internet of Things (IoT), and on the other hand, strive to elevate security of the human factor, is to our knowledge, still quite unusual, if not to say exotic in the field. Intuitively, machine-centric security protocols look to leverage certain very specific traits to reduce very specific intrinsic weaknesses. Human-oriented designs do the same with respect to users and the tools they may be using in a particular setting. Consequently, a tandem approach between two such schemes should create a symbiotic relationship, as it was originally proposed in Bartsch and Huebner's work[4], such that positive features can be established across the board to utilise each and every aspect of the collective strength of both machines and humans. This (which we call Symbiotic Security) is the approach that we depict here by means of a paradigm shift.

To pave the way towards better control of password complexity, the Fast IDentity Online (FIDO) alliance was born in 2012 to encourage and carry out standardisation of strong authentication schemes that, as per the reasoning above, would also leverage hardware elements such as embedded Secure Elements, smart cards, or USB security tokens, alongside strong public key cryptography through its asymmetric nature. This enables registration of user public keys instead of dealing with inferior symmetric passwords. On top of that, once a solid device-centric authentication scheme is established, additional functionality becomes available such as two-factor authentication or Single Sign-On (SSO).

1.1. Contributions

Given that FIDO leverages public key cryptography with everything it entails (including the fundamental issues we explain in this article), just several years back, FIDO consequently announced two new standards and certification initiatives: the Identity Verification and Binding Working Group (IDWG) and the IoT Technical Working Group (IoT TWG). The motivation for this step is to achieve high assurance in remote identity verification for interactive user account recovery and non-interactive IoT device authentication. Specifically, for IoT, secure and automated onboarding and application binding are the new focal points to defeat the lack of global IoT security standards and default concepts that make use of pre-defined password credentials or other secrets (which is incidentally a fundamental issue in FIDO at the same time, too, as we outline later).

Following this trend, in this publication, we present and discuss the original idea of Zero-Knowledge Initial Enrolment (ZKIE)[4] in the light of its applicability to FIDO IoT and introduce a reference implementation in pseudo-code notation of Algorithm 1. The major idea behind this approach is to combine a human-targeting authentication scheme, such as FIDO, with another one akin to ZKIE, which originally addresses the strong mutual authentication needs of machines, particularly in highly isolated and automated environments nowadays typically represented by IoT.

Algorithm 1 Cascaded Registration of FIDO IoT Devices with Zero Knowledge Initial Enrolment of Attestation Certificates

Ensure: All requirements are met, as stated in the text
Require: Installed IoT device as per Original Algorithm III of Bartsch and Huebner[4] (power supply and connectivity) and pre-configured FIDO2 HW authenticator token to enable the session
1: **function** ZKIEAttestationCert
↪ ($NodeSRPAuthParams$, $SecureChannelSessionKey$)
2: Invoke Original Algorithm I[4]
↪ ($NodeSRPAuthParams$, $bUseAttestationCA$); ▷ Run modified Version of the Original Algorithm I using AttestationCA instead of regular RootCA. $NodeSRPAuthParams$ contains all IoT device specific info such as ID.
3: **return** $enc_{SecureChannelSessionKey}$
↪ (AttestationCertKeyPair); ▷ Return over session encrypted channel Attestation Key Pair + Attestation Certificate to IoT device signed by AttestationCA.
4: **end function**
5: **procedure** GenericZKIE(bUseTrustToken)
6: **if** $bUseTrustToken = false$ **then**
7: Invoke Original Algorithm III[4]( ); ▷ Establish regular (byte flag set to true) authenticated and encrypted Zero Knowledge session as is, i.e., with interactive session authentication and authorisation.
8: **else**
9: $FIDO2Session \leftarrow$ Do FIDO2 Authentication with pre-configured authenticator token;
10: Invoke Original Algorithm III[4]($FIDO2Session$); ▷ The Original Algorithm III shall accept the existing FIDO2-authenticated session for authorisation instead of the original user name/password logon on line 1.
11: **end if**
12: Invoke Orignal Algorithm I[4]( ); ▷ Enrol IoT device under tenant CA.
13: Invoke Original Algorithm II[4]( ); ▷ Register IoT device with tenant IoT Hub.
14: **end procedure**
15: main:
16: **if** $InstallerType \equiv trusted$ **then**
17: GenericZKIE($false$);
18: $AttestationCertKeyPair \leftarrow$ ZKIEAttestationCert($NodeSRPAuthParams$, $SecureChannelSessionKey$);
19: IoT_Node ::
↪ InstallLocally($AttestationCertKeyPair$);
20: **else**
21: GenericZKIE($true$);
22: $AttestationCertKeyPair \leftarrow$ ZKIEAttestationCert($NodeSRPAuthParams$, $SecureChannelSessionKey$);
23: IoT_Node ::
↪ InstallLocally($AttestationCertKeyPair$);
24: **end if**

A working prototype based on the proposed reference implementation showcases the entire enrolment sequence built into a typical device from the lighting industry. We include a corresponding video clip as supplementary material to display the entire IoT device onboarding sequence reinforced through FIDO. Finally, we also indicate by way of a future enhancement how another emerging scheme of Symbiotic Key Management System (SKMS) (which is also based on the concept of Symbiotic Security and, as such, combines strong characteristics of both hardware and software)[5] can fortify the combined approach of FIDO and ZKIE at the lowest level where secrets are generated and kept. SKMS can, therefore, be seen as a third scheme in line to sup-

port the symbiotic progression to the point where cryptographic keys both for FIDO and ZKIE are managed in a secure hardware enclosure following the operational principle of Hardware Security Modules (HSMs). However, the symbiotic difference of SKMS in contrast to regular HSMs is the strong bond between software and hardware to make the hardware-generated and protected secrets dynamic and recoverable/changeable in case of a successful attack. The use of hardware-assisted Restricted Operating Environments (ROE) with security resistance against physical attacks is also recommended by the FIDO Alliance for better protection and certification levels[6].

In this work, we provide the following key contributions:

- Security review and analysis of the present FIDO security posture
- Outline how the static management of cryptographic keys in FIDO can be made dynamic and flexible by dovetailing into the design methodology of symbiotic security, which is based on a beneficial combination of all architectural components, e.g., hardware, software, and human operators
- Discuss how FIDO can reinforce IoT security, vice versa, or both
- Present a reference implementation and a working prototype of the FIDO-assisted off-the-shelf IoT device onboarding ("clean slate"), including the secure over-the-air firmware update
- Derive near-production level concepts for a tight integration of FIDO and symbiotic security into IoT devices to form unique embedded identities which can also work across multiple trust domains
- Indicate further enhancements alongside the general idea of symbiotic security by way of future research in the respective fields [e.g., Physical Unclonable Functions (PUFs), non-invasive metadata analysis of encrypted IoT communication flows, or anomaly-based network intrusion detection]

2. BACKGROUND: INFERENCES AND MOTIVATIONS

In this section, we provide a general overview of the FIDO specification, explaining the relevant inner mechanics, which we subsequently analyse from the architectural security perspective whilst comparing our findings to related work in the field. At first, we revisit the original ZKIE to facilitate the reader's understanding of how these two schemes can be connected in a symbiotic fashion and the reasons behind them.

2.1. ZKIE design motivation for dynamic key management

ZKIE was initially developed with the key observation in mind that strong mutual authentication as a fundamental building block of every solid security scheme requires a common trust anchor for all parties engaged in secure or encrypted communication. More often than not, this role is traditionally played by a Certificate Authority (CA) as part of the general public key infrastructure (PKI) approach. Perhaps the single biggest caveat is the question of how CAs can establish the requester's real identity prior to issuing a trusted digital identity or mostly an X.509v3 certificate - a step commonly known as the initial enrolment. With human applicants or requesters, CAs usually mandate the prerequisite step be handled by their Registration Authorities (RA) such that an official ID (e.g., passport) must be presented to be admitted to the workflow. In the IoT realm, however, identity owners and, therefore, the ultimate requesters are, per definition, technical or machine users (i.e., devices), for which the RA requirement is all the more difficult to fulfil - unless they carry pre-embedded and pre-trusted digital identities reliant on likewise pre-fabricated secrets to form what is known as the Root of Trust (RoT). The idea to overcome this classical barrier, which is both static and admittedly not entirely trustworthy (because the generation of said secrets occurs in an uncontrolled environment from the device owner's vantage point), was the main driver behind Bartsch und Huebner's original ZKIE[4] proposal.

As its name would suggest, an alternative scheme would use the power of Zero-Knowledge (ZK) Proof of Possession of a secret to form a dynamic and transparent successor to RoT. Also, while the initial version was based on a long-known and well-tested member of the Password Authenticated Key Exchange (PAKE) family of ZK protocols (i.e., SRP or Secure Remote Password protocol), more recent implementations may

consider leveraging other family members (including the emerging quantum computer resistant designs) given the common security goals ZK must generally achieve by definition. To that end, ZK requires a meticulously planned setup to ensure its strength: the so-called verifier is usually constructed as a computationally hard one-way function such that the receiving party (i.e., the authenticating server) cannot reverse it to reveal the client secret. This is why the verifier naturally constitutes a central element of the ZK proof of possession of the client-side secret. To be able to live up to the criteria of "Zero Knowledge", even the preliminary registration of the irreversible verifier must be sound. Otherwise, the underlying system would run the risk of unnecessary exposure to adversaries such as the Man-in-the-Middle (MitM) or the privileged internal attacker (e.g., rogue administrators). In practice, this is where ZK/PAKE generally "hits its sonic barrier" in the sense that while the verifier could actually be transferred in the clear, manipulating it in transit would still be technically possible. In other words, even if the verifier is received from an unknown or untrusted source, then MitM still poses an imminent risk. Hence, the need for additional pre-authentication arises to ensure the legitimacy of the verifier.

Realising this natural limitation of ZK systems, Bartsch and Huebner have, therefore, suggested including a *Customer Device Bootstrap* procedure (see Original Algorithm III in the Appendix) such that the end device would generate the specific verifier based on its own unique secret and then hand it over to the technical workstation to which it would be directly connected via USB or UART, for instance. It is worth noting that the device itself shall be shipped with just the bootloader inside, only implementing the necessary cryptographic primitives. An auxiliary software utility would then receive the verifier and, upon successfully establishing a secure and authenticated connection to the (e.g., cloud-based) infrastructure back-end, ideally through TLS (here: *EnrolmentVM_WebService*, see Original Algorithm III), deliver this particular device's verifier by way of a protected registration.

To add another layer of authentication and authorisation, the human operator would also have to use his tenant- or organisation-specific credentials to log on through said software utility once the secure TLS channel has been created. However, the original idea of using traditional user names and passwords (line 1, Original Algorithm III) is what we argue needs to be superseded by a stronger authentication scheme, such as FIDO2, to create another symbiotic bond at its own level. In this publication, we will, therefore, outline how this can be done and introduce additional reciprocal options to also show how ZKIE itself could be leveraged to enhance FIDO2.

To conclude the high-level overview of the original ZKIE approach, after the bootstrap step (Original Algorithm III), the actual initial enrolment happens in the back-end / cloud environment as per the Original Algorithm I, which supports the creation of a secure and authenticated Out-of-Band (OOB) channel based on a certain ZK protocol (e.g., SRP) in the initial absence of X.509v3 certificates and, therefore, TLS. Since TLS - being the de-facto data in transit protection standard in (public) networks - is mandated by all cloud-based IoT offerings, the Original Algorithm II utilises the dynamically generated digital device certificates (as part of the Original Algorithm I) to attest to its identity within the owning tenant in order to complete device registration. The device certificates generated in the infrastructure-side HSM are also initially rolled out over the encrypted OOB channel alongside the tenant-specific or use-case-specific firmware.

2.2. FIDO – High-level overview

Original FIDO comprises three distinct specifications:

1. FIDO U2F: *Universal Second Factor* adds an additional strong factor to the user login based on a range of devices attached via USB, NFC, or Bluetooth.
2. FIDO UAF: *Universal Authentication Framework* leverages an authenticator incorporating FIDO UAF for it to be registered with an online service that can instruct the user to engage a variety of local low-level (OS) protected authentication mechanisms such as camera or fingerprint-based biometrics, voice recognition, or the traditional PIN entry.

3. CTAP: *Client Authentication Protocol* provides an application layer for communication between an authen-ticator and another client or platform.

Basically, the core security element of the FIDO ecosystem is reflected in its authenticator. Its main purpose is to protect confidential and critical information such as biometrics and the private key, which is later bound to a specific app operated by the user. The authenticator can be implemented as part of an operating system utilising its low-level protection within a Trusted Execution Environment (TEE), inside a regular application, or embedded in a hardware-based Secure Element. However, it is beyond the responsibility of a FIDO authenticator to also know its user's identity attributes. Also, the *identity binding* as such on the server side is to be *done outside* FIDO.

From the key management standpoint, FIDO authenticators contain two kinds of cryptographic keys: application (caller) specific *Authentication Key(s)* and an *Attestation Key (AK)*, which is pre-defined or injected at the time of manufacturing.

Registration Ceremony [**Figure 1**]: In this overview, we focus only on those technical parameters of the depicted workflow that are relevant to the understanding of its core goal to use public key cryptography and the pre-fabricated AK for the initial registration. There is also a major dependence on TLS to secure FIDO's critical data exchange. Thus, tlsData demarks the TLS session/payload context. The authenticator (1stF eAuthnr) unlocked by the user creates and submits a public key for it to be associated with their account by the corresponding Relying Party (RP). RP initiates this communication by telling the FIDO Client which user can invoke which application (AppID) with the help of which authenticators (policy) plus challenge (to prevent replay attacks) to conduct registration. Upon reception, the client checks the trust status of AppID by verifying the digital signature of the calling application (which is retrieved from the trusted facet list indicated by facetID) and, if successful, uses the Authenticator Specific Module (ASM) as a software driver to communicate with the authenticator defined by the policy. The client then creates a hash (fcp) of all relevant session data, generates the access key (ak) by hashing AppID, CallerID, PersonaID (operating system user account specific) plus a randomly generated ASM secret upon its initial installation (APIKey), and submits that information alongside the user name to the authenticator. The authenticator will now prompt the user for authentication, generate a key pair (k_{pub} and k_{priv}), store it in its secure memory and connect it to the user name + ak. To conclude, the authenticator computes key registration data that includes the session and application-relevant parameters explained above, including its Authenticator's Attestation ID (aaid) and, of course, the application specific k_{pub}. To ensure authenticity, the data block is signed by the authenticator's attestation private key (pre-defined at manufacture) bound to its attestation certificate. The process is finalised by the client forwarding the signed key registration data to RP, where the attestation certificate is validated and the new user authentication data is stored, if successful.

Authentication Ceremony: This process is similar to the registration but rather uses the previously generated application specific key to sign the user authentication object for RP to be verified upon reception with the help of the registered k_{pub}. However, the initial registration is much more critical to the overall security level, making it a focal point of our investigation in this article. Furthermore, in the subsequent discussion section, we will also outline the reasons and areas where we see deficits within the FIDO authentication or so-called assertion phase when we try to combine it with ZKIE for IoT devices, as outlined in the methodology section below. It suffices to point out that k_{pub} belongs to the FIDO authenticator rather than the user who owns it. So, technically, the RP only verifies the identity of the authenticator, which - as we will outline later - is ambiguous in many, if not most, cases.

2.3. Security analysis and related work

At first, we observe that the current FIDO Security Reference[8], while defining FIDO security goals, presenting a catalogue of security measures, and also providing comprehensive threat analysis, unfortunately, does not

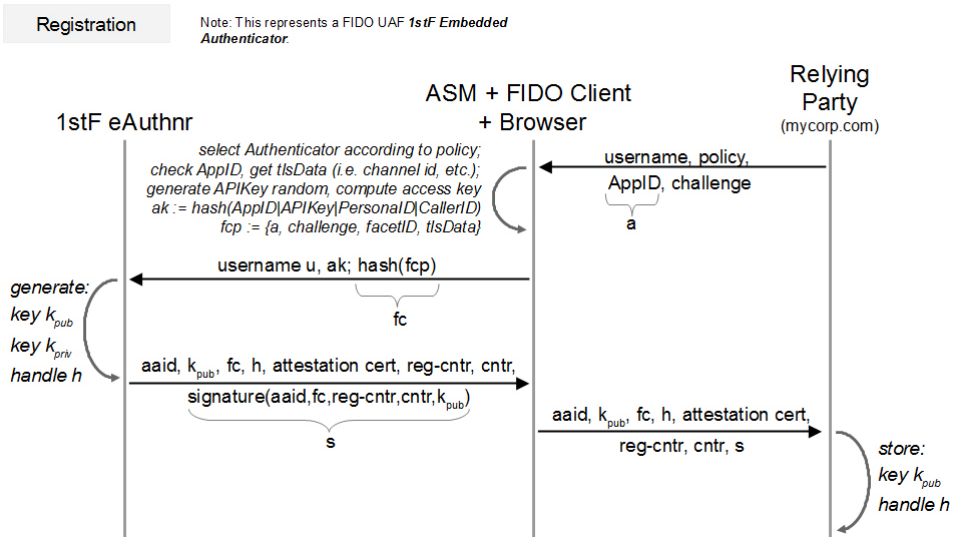

Figure 1. UAF registration cryptographic flow [7].

oblige the implementers to follow its otherwise accepted practice. On top of that, the resulting FIDO certification does not consistently require the vendors to actually disclose the details of their implementation, which is commonly regarded as counter-intuitive and counter-productive in the world of modern cryptographic schemes and protocols: while the specification itself is open to peer review, FIDO implementers are even encouraged to introduce security measures and enhancements of their own that usually remain confidential for competitive reasons.

2.3.1. Attestation types/Static key management

However, from a technical standpoint, the weakest spot of the standard is the underlying authenticator attestation private key, as we have already pointed out, which is injected during the manufacturing stage. This creates a serious trust issue, at least in highly critical environments, since the vendors are in direct control of this step, resulting in their knowledge of said secrets. The issue is further aggravated by another observation that the authenticator AKs do not have to be unique. It is actually the desired behaviour [9] to be able to distinguish a certain make and model rather than an individual device. Many services would even allow "*self-attestation*", also called surrogate attestation, relying on the assumption that the user already has an authenticated session, which, in turn, is secured by TLS. Unfortunately, this often results in a self-signed certificate, which defeats the very purpose of certificate-based authentication. Paired with the long-standing awareness of the limitations of the memories utilised in microcontrollers, non-invasive extraction techniques [10] could have a devastating impact on FIDO (hardware) authenticators (or IoT modules, for that matter): key reuse leads to a complete compromise of the entire product line. Even if the vendor decides to revoke the AK, RPs will be forced to ignore them entirely, seeing how authenticators in production would never receive a fresh private key in a remote fashion.

To combat this Basic Attestation shortcoming, FIDO came up with the notion of a *Direct Anonymous Attestation* based on *Elliptic Curves* or *ECDAA*. The original DAA [11] uses ZK Proof, which is similar to the novel IoT security approach we discuss in the following. However, the scheme itself is highly complex and is also based on the usual Trusted Platform Module (TPM) assumption that only holds true under the existence of a secret key embedded into the processor during its assembly. In addition to that, DAA was originally designed to ensure user anonymity and non-linkability through group signatures as opposed to traditional signatures for authentication, non-repudiation, and integrity. DAA went on to be found insecure, received an update, and got switched from symmetric to asymmetric building blocks (RSA), which was ultimately deemed insecure as

well. However, more importantly, its current and perhaps most efficient form, *ECC-DAA*, has been proven to be prone to a fundamental issue that if one TPM-based signer (i.e., authenticator) is compromised, then authentication fails for the entire group of connected signers[12]. Incidentally, the previous RSA-based version was also susceptible to the same assumption. The authors describe a possible fix for this particular problem, but it is once more dependent on the static nature of the pre-fabricated endorsement key of the TPM, and the fix itself is also incompatible with the current TPM specification.

With all that in mind, two particular features seem strikingly critical:

1. the reliance on static pre-configured endorsement keys and
2. the use of group signatures to cater for anonymity and non-linkability.

While we will discuss aspect (1) in the light of a different and more dynamic ZK approach in the next section, aspect (2) may not even be required in the presence of non-interactive technical systems such as IoT. IoT devices are usually deployed in large quantities where all of them belong to the same context of a customer, tenant, or realm. Therefore, the ability to distinguish single nodes from one another can be a desirable requirement that also enables directly targeted reactions to single devices having been compromised to contain potential damage. This assumption is further reinforced by the traditional reasoning for interactive or human users who can have multiple online identities bound to the same authenticator, which is certainly counter-intuitive for technical systems such as IoT nodes. With the benefit of hindsight, giving up anonymity and non-linkability would also greatly simplify the necessary authentication means, thereby naturally reducing the attack surface.

2.3.2. Cryptographic pitfalls

FIDO has largely adopted the fundamentals of that scheme under its own new specification called *ECDAA*[13], which, in turn, has been consumed by FIDO2 and its core component *WebAuthn (Web Authentication)*. While ECDAA itself is still used for device attestation only, WebAuthn incorporates (multi-factor) authentication towards a RP such that RP is usually a website, browsers play the client role and ultimately use a variety of FIDO2 authenticators to assert the user's presence and knowledge of a secret. However, rather than being scrutinised at the level of *ECC-DAA* as outlined above, most criticism has so far been directed at a number of fundamental design or even editorial shortcomings within ECDAA. For example, one essential requirement when working with elliptic curves is point validation to ensure that the respective points really are located on the negotiated curve. However, it is not explicitly mentioned in the specification, or even more importantly, that there are efficient algorithmic ways to ensure correctness of a point-to-curve relationship by using point compression or the compact point representation. To make things worse, ECDAA makes use of Barreto-Naehrig curves that are now known to have reduced equivalent complexity of about 96 rather than the nominal 256 bits, thereby failing to reach the recommended minimum of 128 bits by a count of 32. This deficit amounts to a complexity reduction of four billion combinations of the key space, speeding up brute-force attacks. Last but not least, ECDAA relies on signatures with random(ised) padding, which can lead to padding oracle attacks, similar to the original *Bleichenbacher* attack against *PKCS#1v1.5* padding[14]. Considering the fact that this finding is several decades old, this decision can only be referred to as highly questionable. On top of that, developers do not receive any guidance on the essential topic of (Pseudo) Random Number Generation (PRNG), which can lead to fatal consequences given that platforms vary in their low-level implementations, as does the parametrisation of these PRNGs. To avoid this pitfall and the potential oracle padding threat, a deterministic scheme, such as *RFC 6979*[15], could have easily been incorporated into ECDAA since those do not necessarily need access to a source of high-quality randomness. Incidentally, WebAuthn itself still supports RSA PKCS#1v1.5 padding.

2.3.3. Further criticism

WebAuthn may have also introduced other possible attack angles such as the option not to use ECDAA at all as part of the authentication. In fact, current implementations allow both RPs and users to decide whether an

attestation statement should be sent during the registration or not. This could be exploited as a downgrade attack in such a way that the attacker would negotiate the use of authentication without ECDAA with an RP or similarly target the client browser and its JavaScript Object Signing and Encryption (JOSE) implementation, which, too, allows RSA with PKCS#1v1.5 padding among other likely vulnerable aspects[16]. This could pave the way for a large-scale coordinated attack where adversaries could swap authenticators for malicious client devices (either in hardware or in software) without detection, provided that device attestation has been effectively incapacitated by the preceding downgrade attack.

In addition to our findings, related work, such as[17], shows other potential gaps alongside the security analysis of the various FIDO components, including the FIDO client and the FIDO authenticator we mainly focus on. It also conducts a formal threat analysis listing the consequences.

2.3.4. Motivation for dynamic key management

From the architectural standpoint, FIDO basically attempts to transmute the usual server-side authentication problem into a client-side task by outsourcing the immediate user authentication to a shielded local authenticator, but not without certain risks such as those presented above. One of these risks is the fact that some aspects of the overall workflow are beyond the scope of the specification or optional at best. Therefore, to make our suggested approach more consistent with the original idea of FIDO, we propose the adoption of a recently introduced "*Reference Architecture for Secure Cloud-Based Remote Automation*"[4] (presented in the previous section). In the next section, we will also outline if the scheme can be directly applied to the FIDO IoT realm or if, in fact, certain modifications would be necessary to pave the way for dynamic key management in line with various options for combining FIDO and ZKIE, depending on the use case in question. Most importantly, though, pre-defined or re-used secrets shall no longer be required, and secrets, in general, shall only be known to the respective modules that produced them in a dynamic fashion. In fact, a dynamic key hierarchy would also greatly facilitate the process of device repurposing, e.g., when the same IoT node is sold or migrated to a different tenant. In such a hierarchy, every device would initially create its own undisclosed random master key within its secure enclave, generate a tenant key for the *ZKIE*[4], and encrypt it in its Flash memory with the master key. The former could then be disposed of upon migration to start off with a clean slate and avoid correlations.

2.3.5. Efficient certificate-based dynamic attestation

In line with the previous requirement of dynamic key management and in conjunction with a valid but rarely used option which states that an "*authenticator may be capable of dynamically generating different AKs*" and certificates per origin[18], § 14.4, the subsequent enrolment occurs in a multi-tenant manner [each tenant creates its own CA or Attestation Authority (AA) in case of FIDO] allowing for a uniformly adopted and implemented approach by all vendors, but also supporting strong separation of those within a protected (private) cloud environment. The cloud-based tenant CA can then issue certificates to IoT modules relying on the pre-trust established through carefully set-up PAKE. Because the PAKE connection is also encrypted, the tenant CA can transfer additional sensitive information over that channel such as the use case specific IoT firmware. Contributing to the idea of symbiotic security, IoT modules can use their strong hardware features to create a digital signature of the installed firmware image to ensure its integrity upon every start. Lastly, if one of the modules were to be compromised, its certificate could be easily revoked through its tenant CA without affecting any of the other modules in production. The module could even return to its original state by detecting fraud through its secure boot mechanism, deliberately invoking the initial enrolment protocol to obtain a fresh certificate and legitimate firmware. The necessary logic is compressed into a write-protected bootloader that does not contain any pre-defined secrets and, through its nature and reduced size, naturally minimises the attack surface. This approach also simplifies or deprecates some essential assumptions of the ongoing *FIDO IoT TWG* effort such as the distinction between trusted and untrusted installers or different protection level categories, because it is, in fact, feasible to get away with just a single protection level by ef-

fectively harnessing hardware reinforced cryptography and key management on resource-constrained Ultra Low Power devices such as Cortex M0[19]. FIDO's own IoT specification, however, (*FIDO Device Onboard Specification*[20]) still relies on the root of trust principle, which obviously represents the path of least resistance from the manufacturer's perspective. Nevertheless, to maintain this traditional idea, the required complexity of the underlying schemes, e.g., through various sub-protocols (Transfer Ownership Protocol 0 through 2), routing requirements, ownership vouchers, device attestation certificate chains, and other precautions, grows considerably. A higher complexity may not be necessary at all in the light of other more straightforward and less error-prone alternatives such as the ones we present in this paper. Then, there is also the issue of trust and flexibility: it is hard to argue that root of trust schemes with their pre-fabricated secrets lend less confidence to users and customers than one where the owner completely controls the entire infrastructure and chain of events and, more importantly, the underlying secrets. For this reason, similar technologies, such as the TPM, have long been criticised, even by authorities[21].

On top of that, the notion of a dynamic certificate-based attestation paves the way for additional refinements, such as the more granular separation of Certificate Authorities based on their specific designation and ownership [e.g., Attestation Cas (AttCA) and Assertion CAs (AssCa)], to cater for increased or more stringent authorisation requirements. We highlight this added benefit in Section 3.2 to also exhibit the freedom of being able to distinguish between manufacturer identities (which could, at that level, for instance, attest to the integrity and authenticity of freshly assembled devices) and the ultimate trust established by the owner or buyer of those devices within its controlled environment. In that way, the device owner can choose to rely on the manufacturer's RoT while, at the same time, preserving the ability to create its own trusted root.

2.3.6. Putting it all together

In terms of better suitability for resource-constrained devices (e.g., hardware-based FIDO authenticators), it is worth pointing out that FIDO has specifically based the ECDAA scheme discussed above on elliptic curve cryptography to increase its efficiency due to considerably shorter key sizes generally attainable with those at the same security level, however. Incidentally, TLS, as a vital ready-to-use alternative, as also mentioned above and further emphasised below as part of the suggested approach, boasts a sophisticated concept of cipher suites providing fully-fledged EC support, including forward secrecy. From the efficiency standpoint, TLS is on a par with ECDAA and the subsequent signatures both in FIDO registration and authentication steps. Arguably, it has also been the de-facto data-in-transit protection standard for several decades, having stood the test of time by going through multiple versions, specific security enhancements and extensions. Interestingly enough, ZK schemes have also been known for a similarly long time to claim comparable evolution for themselves, not to mention the similarity of the essential security offerings of both TLS and ZK, e.g., strong mutual authentication, session encryption, *etc.* These rather simple observations, including the referenced AttCA profile, which implies the use of TLS to begin with (given that TLS itself is largely based on digital certificates), certainly beg the question of whether TLS (or one of the solid ZK specifications) could have been a more straightforward choice for the FIDO alliance to make. In retrospect, though, it should at least be possible to combine the security features of the existing FIDO options with ZK and TLS such that the collective strength is elevated following the general direction of Symbiotic Security. The next section shall elaborate on all such beneficial combinations.

3. PROPOSED METHODOLOGY

In this section, we describe our proposed approach to managing the key pitfalls identified in the previous section by analysing all beneficial combinations of the aforementioned security schemes. Furthermore, we depict possible implementations in various notations ranging from pseudo-code to UML diagrams to take into account the respective requirements which drive those use cases or combinations accordingly. Those combinations are not necessarily superior or inferior to each other but rather reflect different specific needs

and application priorities. They are, in fact, based on a simple observation that, just as we argued in the introduction, FIDO could enhance authentication and authorisation in ZKIE, and ZKIE could, in turn, be of help when it comes to dynamically managing FIDO AKs.

3.1. Combinations of zero-knowledge initial enrolment with FIDO authentication for the FIDO IoT specification

We see three different yet partially intertwined (with respect to their similarities) ways [Figure 2] to apply ZK Initial Enrolment (ZKIE) to FIDO IoT, vice versa, or both. Please note that green pathways in this concept map denote fully authenticated TLS connections, which are the desired final state and communication relationship between all entities, including the user, in a typical IoT setting. Additionally, in Step 0, the ZKIE-enabled FIDO Server and the IoT Hub have a common trust anchor based on digital certificates and can, therefore, engage in TLS communication. This would be needed if the IoT Hub were to become a RP leveraging the FIDO Server as its Identity Provider (IdP) for session authentication of the user controlling the IoT Node.

1. **Complete substitution** of FIDO's present registration and authentication scheme, simplifying it to certificate-based authentication, where every user would obtain a client certificate following their successful authentication (e.g., biometric) towards the built-in or attached authenticator. The latter would still be the guardian of the underlying private key. Federated or inter-tenant trust would also work through cross-signing or mutually trusting tenant Cas, as we will highlight in the following practical discussion of embedded identities in IoT. In summary, ZKIE could conveniently replace FIDO's AttCA profile introduced in the previous section while also bringing about dynamic AK management and leveraging the bootstrap OOB authentication and the secure firmware roll-out or over-the-air updates. At the same time, ZKIE would play its original role in IoT by solving the initial enrolment issue. The dual use is most evident in Steps 1 and 3 of Figure 2: ZKIE is both invoked by the FIDO Authenticator (proposed use) and the off-the-shelf blank IoT Node (original use). Given the full range of dynamic features, we believe that this would be the most advanced option. However, we must also take into account different priorities and requirements of the various situations we discuss in this paper. The ultimate minimum requirement, though (at least from our perspective and for reasons we have already explained in the previous section), is to reduce static key management while also transmuting the usual RoT into a more owner-friendly and controlled trust domain.
2. Improved dynamic management of the AK where ZKIE would help create unique undisclosed secrets which could subsequently be used for bootstrap authentication in order to obtain the actual AK and certificate signed by the corresponding tenant/attestation CA. In essence, this option would constitute a down-scaled application of ZKIE in FIDO such that only AKs could be made dynamic while still preserving the manufacturer's RoT. At this point, it is worth noting that ZKIE could both improve the general FIDO key management as part of its original mission statement to replace weak human passwords, and ZKIE could also bolster the key management within the RoT of the current FIDO IoT specification [20].
3. In the original reference architecture, ZKIE is used to obtain the IoT device certificates to enable them to communicate with their cloud service (IoT Hub). The FIDO authenticator challenge and response protocol with the FIDO server also requires TLS to establish common communications and authentication ground because, in TLS, certificates are used for strong (mutual) authentication. Once the mutual trust between an IoT device and an interactive user has been established (because their authenticator has obtained a trusted certificate), the user can now (a) directly connect to the device to control it; or (b) use the IoT Hub to relay his commands to the IoT device getting telemetry data in return over the same fully trusted channel. It is worth noting that this would constitute a non-invasive approach from the perspective of a cloud IoT infrastructure as all required elements would be simply added to it. In other words, it should be possible to either keep FIDO user authentication as is to indirectly authorise the user's remote-control session of the IoT Node (thanks to the common trust anchor between the IoT Hub and the FIDO Server) OR thanks to ZKIE roll out a user-specific certificate (see Step 8 in Figure 2) to enable secure end-to-end remote-control sessions between the user and its IoT Node (Step 9).

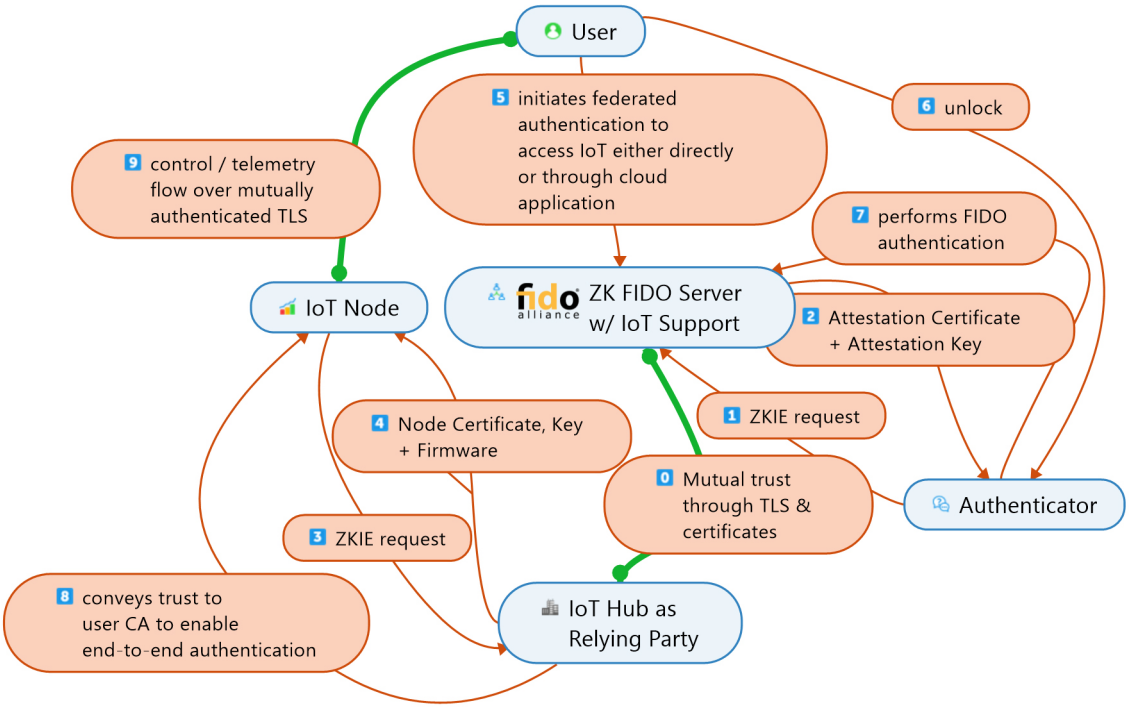

Figure 2. ZK-authenticated FIDO - IoT connection.

To demonstrate one of the combinations, we present a workflow in pseudo-code notation of Algorithm 1, making the following assumptions: a batch of IoT devices is shipped, as outlined in Figure 1 and Figure 2 of Bartsch and Huebner's work[4], and the reference ZKIE cloud-side platform is present alongside a FIDO RP/Server.

In addition to the actors depicted in Figure 2, we also distinguish between a **trusted and untrusted installer**, which entails different actions in the Original Algorithm III[4] *line 1* of the same work. A trusted installer is basically someone who is affiliated with the IoT node owner (i.e., tenant or RP as per FIDO nomenclature) and also has the corresponding entitlement to onboard devices/authorise the enrolment session. An untrusted installer, on the other hand, may just be a contractor or a 3rd party technician who has been tasked with the installation of the IoT device in question. Lacking the necessary authority, this worker requires a different time- or session-limited trigger to initiate onboarding without learning any underlying secrets because we employ ZKIE instead of pre-defined AKs, ensuring that a malicious technician could not simply clone or steal devices to hand. An adversary could still try to install devices under his control into a rogue environment - a threat that certainly requires mitigating procedures at the organisational level (e.g., shipment tracking). Technically, it would be possible to leverage FIDO protocol bindings and authenticator policies to tie certain sessions to specific IP addresses, for instance. This does not provide ultimate protection, but every bit of mitigation counts. This is also something we highlight in Figure 3 and the related practical discussion on embedded identities in IoT to reflect the potential requirement for dual control or segregation of duties as part of a larger supply chain.

Because we do not necessarily trust the installer and because he is not always part of the tenant enterprise, we use a FIDO2 hardware authenticator (either owned by the contractor or through the previously mentioned device repurposing) pre-registered by the tenant (employing ZKIE to avoid static keys) and configured to work within a specific narrow time frame in which the installation is supposed to take place. In this way, when the shipment of IoT devices arrives at its destination, the (3rd party) operator can fetch his FIDO2 installation enablement authenticator device and use it on the premises to unlock the ZKIE session both for the regular

x.509v3 certificates required for normal cloud IoT operation and subsequently also for the Attestation Certificates required by FIDO/WebAuthn. The full life cycle suitable for larger (e.g., enterprise-level) deployments, including the notion of the (un)trusted installer, is also visually expressed in the UML Activity Diagram Figure 4.

In addition to the high-level pseudo-code notation of the overall process, a working prototype is presented in the supplementary video clip. The footage exhibits a real-life example of an industrial dual-head lighting system that was extended with the help of a secure microprocessor (see Barstch and Huebner's work[19]) to enable Ethernet connection all the way to a cloud-based IoT hub while also providing analogue and digital controls for the underlying electrical circuits (the two separately connected lighting heads). The video then demonstrates all essential steps of the (un)trusted installer use case, starting with a blank device with only a bootloader inside over the complete IoT device registration with the infrastructure back-end up to the enforced secure wipe and over-the-air firmware upgrade. In the following section, we will also "shed more light" on how unique identities can be embedded into IoT devices with FIDO assistance such that these two schemes can be tied even closer together in a practical setting.

3.2. Practical discussion of embedded identities for IoT and additional enhancements

As we observed in the previous section, the proposed approach for embedding identities in IoT relies on the combination of a dynamic scheme of ZK Proof of Possession of a self-generated device-unique secret with FIDO authenticators. The reference implementation/working prototype (supplementary video clip) showcases the typical setting where a human operator leverages a separate hardware-based (biometric) FIDO authenticator to authenticate and authorise the onboarding session. FIDO authenticators could, however, theoretically also be integrated into IoT devices, with the authenticator enrolment being secured either by the ZKIE specific patterns already discussed above or by a classic process (rooted trust), which is usually combined with IoT device deployment. Either way, the main focus is to free the IoT devices (and optionally also the FIDO authenticators) from the long-standing dependency on cryptographic secrets pre-injected by manufacturers to establish a root of trust. Hardware roots-of-trust, such as PUFs, are one such state-of-the-art method to derive hardware-centric "fingerprints" (attestation data)[22]. The term "fingerprint" is quite intriguing in this case, given that FIDO also considers authenticators verifying their users' biometrics to be more secure. To that end, devices will use unique, volatile, yet reproducible cryptographic keys (e.g., symbiotic-active PUFs, which we call *SKMS*[5] to be highlighted by way of parallel work and future extension in the upcoming section) to form an alternative dynamic root of trust leveraging ZK Proof. Therefore, the main intention of our additional work in this field is not only to remove pre-fabricated secrets but also to make them more difficult to extract through direct physical attacks, even if they are dynamic and unique to begin with. While there are other excellent contributions to the combined application of PUFs and IoT such as[23] or[24], our main goal is symbiotic security design, which, with the notion of SKMS, now not only utilises PUFs but improves the resilience against traditional PUF threats themselves (modelling, side-channel, fault injection attacks, *etc.*) through yet another symbiotic relationship rather than simply assuming the PUF unit to be entirely secure on its own. It is important to note the traditional concept of an attacker attempting to compromise one of the hardware or software components of such a system individually. In the case of symbiotic design, an attacker must compromise all components simultaneously and in good time to gain sufficient advantage. It is because of this that we also refer to SKMS as *Symbiotic Unclonable Function* or *SUF*. The reason we firmly believe this approach warrants its own new term (SUF) is that, in contrast to the state-of-the-art PUF design methodology, we break out of the usual limitations of purely physical constructions. In this way, in case of a successful direct attack, the unique device secret can be modified remotely without actually ever learning the new secret to adhere to the overall ZK principle. It is important to outline that the overall architectural security can always be improved, which we intend to continue doing, guided by the symbiotic design rationale. Otherwise, we briefly revisit the subject of PUFs in the next section to conclude our discussion of the potential implications of subsequent or future implementations.

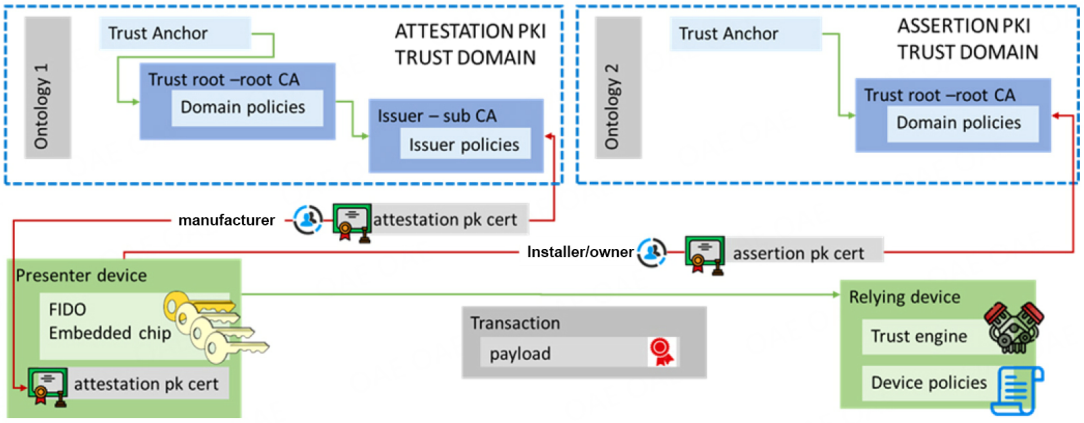

Figure 3. Relationship between installer authoritative trust domain and device trust domain.

As we have already indicated, FIDO can help overcome the usual limitations of "technical" identities since all IoT devices require some sort of local installation and initial set-up (as can clearly be seen in the video footage), which is normally performed by a certain entity. Paired with the aforementioned trust framework based on digital certificates or PKI, we define a generic scheme that conveys trust established within the installer authoritative trust domain (i.e., CA) into the device trust domain. In other words, low-level trust is elevated to a higher level where multiple entities/organisations/domains can fully accept the assurance of the originating entity/organisation/domain. The general outline is depicted in Figure 3.

For this purpose, we adopt the FIDO concept of having two levels of asymmetric cryptographic key pairs:

- Attestation Key (AK) pair, with its private part typically being used to digitally sign properties related to the manufacturing process, the functional capabilities of a FIDO authenticator and new cryptographic keys dynamically created by the FIDO authenticator. We subsequently also use this same terminology when referring to the entire IoT device with an integrated FIDO authenticator because, in this case, both form conceptionally and technically a single entity.
- Assertion Key or Resident Key (RK) pair, with its private part being used to digitally sign properties related to a given business transaction with a RP, e.g., login attempt. As for the AK pair, we subsequently also use this same terminology when referring to the entire IoT device. It is, however, important to note that a FIDO device may produce multiple assertion key pairs with the corresponding public parts being signed by the same attestation private key.

AK, which is traditionally injected together with the firmware into a virgin device when adopting it in a given domain [Figure 3], is elevated by issuing an attestation (for industry compliance, in the form of an X.509v3 public key certificate) for the device that is cryptographically linked to the attestation public key. For this purpose, the RA of the PKI (typically the manufacturer itself) performs a device identity-proving process prior to authorising the certificate issuing process performed by the AttCA. We designate this process as "*device branding*". It comprises the following steps:

1. The virgin device is connected with the PKI on a secure channel (through ZKIE).
2. The AK pair is generated in a secure environment, typically inside a suitably certified HSM.
3. The attestation public key is signed by AttCA, together with other device-specific attributes (e.g., specific type, functions, and other FIDO-specific capabilities) after having proven their validity by out-of-band means (ZKIE), while the signature may comprise the entire firmware supplied by the vendor(s) and additionally (possibly digitally signed) testimonials regarding the firmware development process.
4. The firmware is injected into the device together with the generated AK pair.
5. The issued public key certificate is supplied to the adopting domain that may comprise one or several IoT

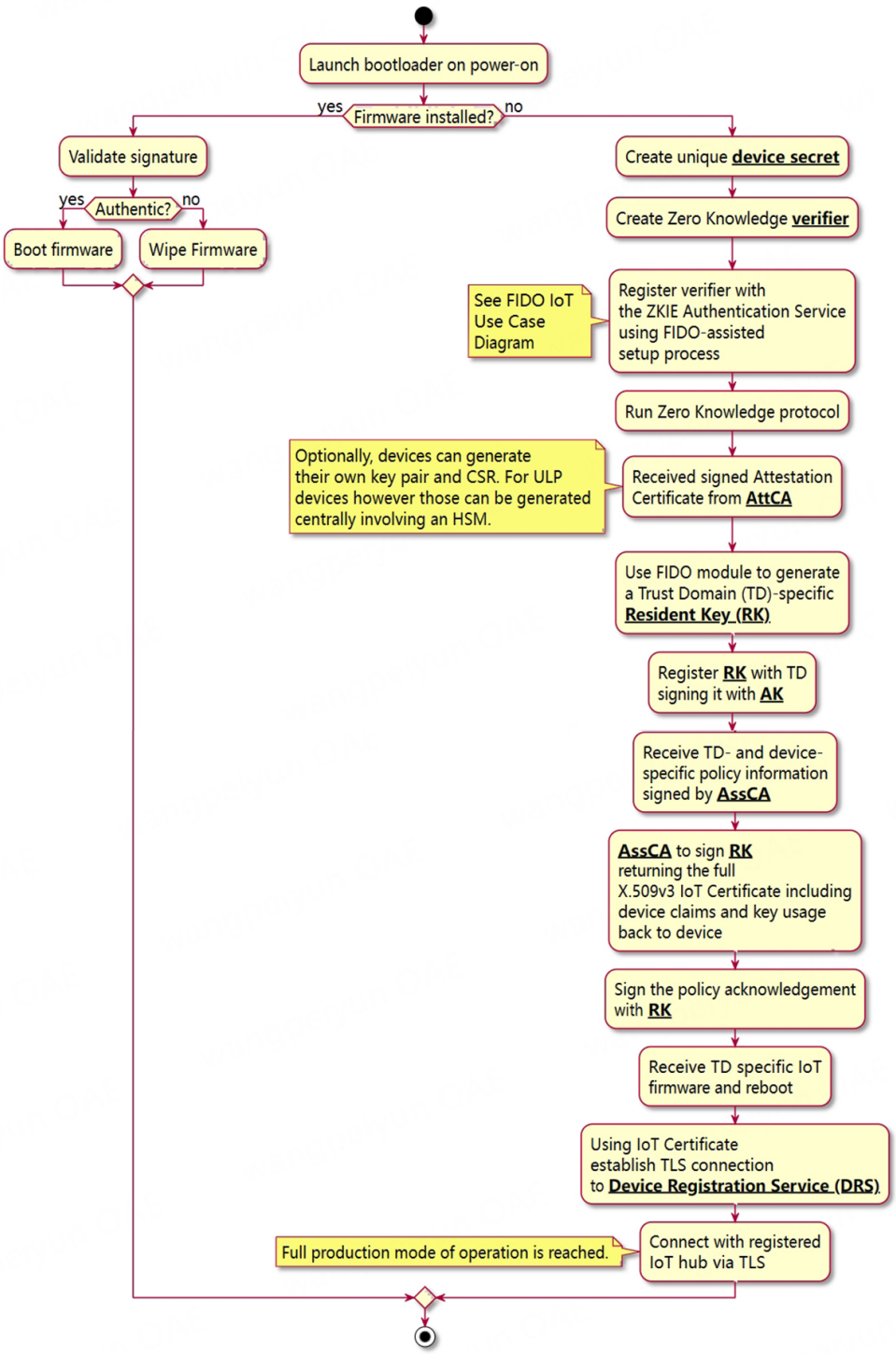

Figure 4. FIDO-assisted IoT onboarding - Activity diagram.

applications using the branded device.

The branding process may be repeated when zeroing the device, e.g., when wiping it for being re-used in a different domain ("*device repurposing*") or when upgrading the firmware (version). The issued certificate allows the device in question to participate in IoT applications by displaying all injected properties and the associated device behaviour. In order to install a branded device into a given IoT application, optionally configuring and registering it with a RP cloud/server follows a different process, subsequently called "*device deployment*". This process comprises the following steps: the RP entity plays the RA role, and the installer acts as a delegated RA

agent.

1. A new RK pair is generated in the device upon an RP request.
2. The RK pk is signed by the device, together with other attributes (originating from the request and from the preceding branding process).
3. The IoT cloud / server application validates the signature and registers the fresh RK pk.
4. The PKI receives the signed data with the RK pk and other (possibly digitally signed) claims from the IoT application in question, such as a trusted installer (acting as an RA agent).
5. After successful verification of the request by the RA, the AssCa issues a public key certificate corresponding to the supplied assertion key (RK).
6. The issued public key certificate is returned to the IoT application, which proceeds to register it.

Basically, this approach adds an additional purpose to the existing FIDO key hierarchy, as displayed in the UML Activity Diagram notation in Figure 4.

1. At first, authenticators must embed an AK, which they then use to sign new RKs for every RP. This is FIDO's root of trust, which, in our case, is a unique, device-specific key to be authenticated through ZKIE and signed by AttCA.
2. Every RP registers a new RK generated by the authenticator in question, which has been signed by the AK. RK's private part is then used in every authentication session with the corresponding RP to sign the assertion of said session.
3. New: the signed RK now becomes a fully-fledged IoT certificate required by today's major IoT platforms such as Azure IoT and AWS IoT.

The AK pk and the associated device certificate originating from the branding process enable RP and the PKI to verify the authenticity and integrity of the freshly generated RK, thus ensuring its binding to the same IoT device and validating a transaction (authentication or payload signature) performed by the IoT device during operation at RP's behest. Suitable secure processor technologies (e.g., ARM Trust Zone or the SKMS scheme briefly introduced in the next section) will be considered to enforce the secure device boot cycle. It is also worth mentioning that in contrast to ordinary FIDO patterns, an embedded identity, besides authenticating an IoT device, can also be used for cryptographic data binding that is generated by such a device (by means of payload signing), which enables appropriate payload protection as a novel technology based on this specific scheme. Being persistent artefacts, payload signatures may also be validated by an arbitrator or a judge in case of a dispute on the grounds of the corresponding device certificate that is tracked by the issuing CA.

Overall, the idea is to leverage FIDO to add a dedicated level of assurance to the initial device setup, which happens to be an omnipresent human intervention factor in the world of IoT (trusted or untrusted installer). Therefore, the UML Use Case diagram [Figure 5] summarises (including the workflow depicted in Figure 4) how FIDO can be used to facilitate both trusted and untrusted installations.

4. IMPLICATIONS OF PROPOSED CONCEPT IMPLEMENTATION

Thus far, we have provided a description of the requirements of ZKIE for FIDO in terms of infrastructural and higher-level information flow requirements. In this section, we outline valuable items for discussion into the broader requirements of the proposed scheme with regard to lateral infrastructural concepts. Based on our previous work[5], we firmly believe those could be used to enable a practical implementation of our scheme. For this, we discuss the needs for benefits of and potential integration of both hardware roots-of-trust and Artificial Intelligence (AI)-assisted techniques to supplement the key issues that would be faced by our scheme, namely in strong hardware identity and device self-recovery with ZK in mind.

In a way and alignment with the symbiotic security design rationale, the research topics presented in the

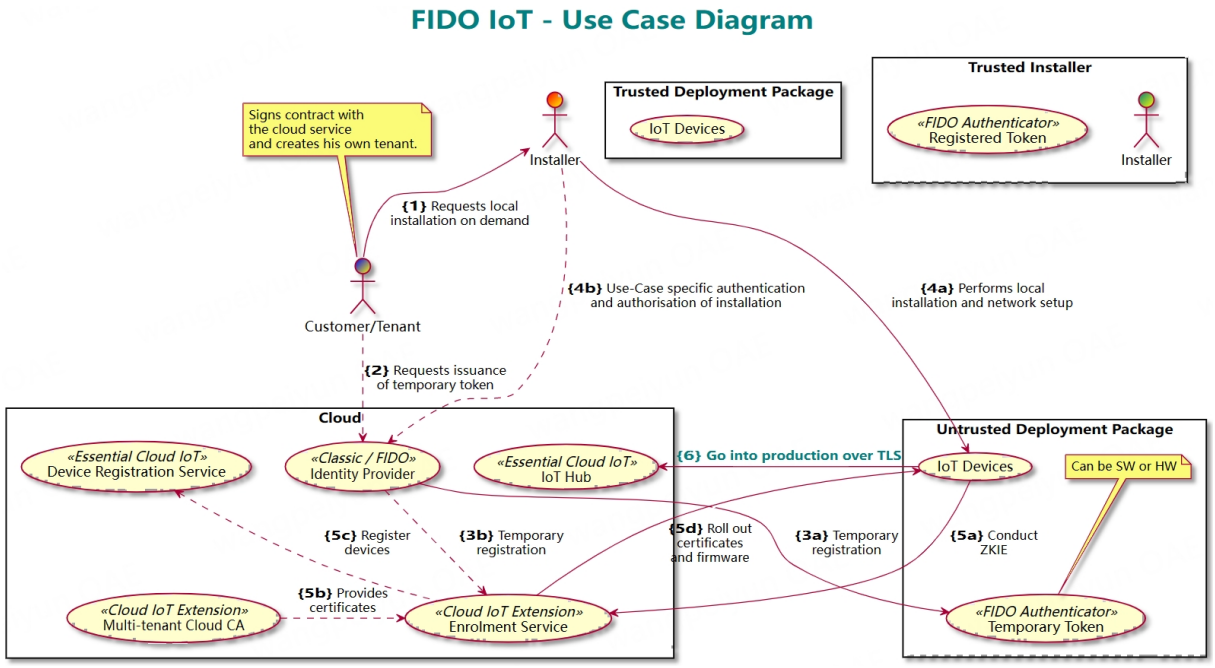

Figure 5. FIDO-assisted IoT onboarding - Use case diagram.

following can be regarded as yet another beneficial extension of the original idea[4]. As a matter of fact, we have already presented such an extension in this paper with a different architectural aspect in mind: an effective combination of ZKIE with another security scheme, i.e., FIDO and the resulting options.

4.1. Dynamic hardware roots-of-trust

In order to establish a hardware-centric element for symbiotic attestation in any context, it is essential for a given device to be capable of deriving a device-specific identifier that is intrinsically bound to some phenomena or physical variation which is unique. Given the context of biometric authentication, a fingerprint provides excellent attestation of human presence due to the unique and (mostly) immutable and highly unclonable nature of a fingerprint. Mirroring this authentication context in terms of IoT, strong proof of possession of a unique physical component also provides strong trust in a similar fashion.

With an implementation of FIDO for IoT, a threat model which does not account for an attacker who may gain physical access to devices is insufficient due to the assumption that adversaries can access keys stored in non-volatile memories through side channels. Therefore, before the software layer can be trusted, a trust anchor must be established, ideally dropped directly at the level of device hardware components. Through the use of technologies such as PUFs, it is possible to measure and exploit widely known and measurable variations caused during device manufacturing in order to not only generate keys to authenticate the presence of a device but also the individual components of which a device comprises, such as individual memories (DRAM, SRAM, Flash, *etc.*), processors and Field Programmable Gate Array (FPGA) blocks, *etc.*[25–28]. In this sense, PUFs could be utilised alongside software features, such as secure boot, to provide an initialisation step to FIDO where a device requires a re-enrolment to regain trust in the legitimacy of the components of a device which has become temporarily untrusted. Secondly, PUFs are naturally suitable for dealing with both vulnerable key data and the previously discussed issue of using static AKs across different devices due to the unique property of a sufficiently constructed PUF[29].

With these aspects of PUFs in mind, there lies an open field of research into integrating PUF constructions into enrolment procedures where Zero Knowledge is presumed. In this way, dynamically, the overall PUF function

need not act as a single entity for device authentication; rather, it can be designed to operate dependent on other layers of architectural security (i.e., software secrets) and, therefore, may be renewed over time while retaining device-specific characteristics. This idea is conceivable when operating with memory PUFs, where different data (software) patterns written to the memory can cause unique changes to the PUF output. Therefore, PUFs would likely make an ideal component for constructing a symbiotic security environment for deploying a ZK registration with FIDO.

In summary, on top of the beneficial combinations of FIDO and ZKIE that we have introduced in the previous section (which essentially remove the need for static pre-injected secrets), we also propagate the added advantage of dynamic secrets generated at run time in a secure hardware enclosure such that they also become changeable if compromised thanks to the concept of SKMS in contrast to classical PUFs where secrets are volatile, but static nonetheless. Because of the natural uniqueness of those secrets introduced through PUFs (as part of the SKMS), we refer to them as hardware "fingerprints". As for the actual fingerprints or human biometrics, we highly recommend those in line with the FIDO Certified Authenticator Levels[6] (which is why we leverage a FIDO Level 2 device with fingerprint authentication in our video demonstration). In the next subsection, we will, however, explain why the intuitively stronger addition of human biometrics to FIDO authenticators alone (i.e., without the combined use of FIDO, ZKIE, and ideally also SKMS) does not quite live up to the high-security guarantees expected for machine-to-machine communication (e.g., in IoT environments) and, therefore, their mutual authentication. In essence, human fingerprints are already used to unlock regular FIDO authenticators, while the hardware fingerprints make for naturally unique secrets to reinforce the underlying cryptographic boundary. Still, human fingerprints cater for a stronger bond between the human user and the FIDO authenticator, which is why the FIDO certification level with those devices is always higher. Incidentally, while human fingerprints in FIDO and the hardware fingerprints in SKMS are presently independent of each other and quite 'simply' play their respective role of increasing security at their own level, the symbiotic design paradigm naturally calls for a symbiotic combination of the two as part of our future research.

4.2. Metadata analysis, AI-assisted intrusion detection and authentication

To be able to consistently implement any of the presented schemes aimed at improving the FIDO Basic Attestation profile, every FIDO-capable IoT device obviously needs to receive a unique identity in the process. This is, however, somewhat counter-intuitive from FIDO's perspective because up until now, the alliance has always believed the best way to protect user privacy is by embedding the same key into thousands or millions of devices and having only a loose coupling between the user, the authenticator, and the RP. In other words, with regular FIDO, only the authenticator "knows" the user either through a simple presence test or a more elaborate biometric pattern (e.g., fingerprint). The RP, however, only learns that the user who initiates a particular logon session happens to be using one of the many authenticators of a given manufacturer. The user's actual identity - still disguised by the total number of authenticators from the same manufacturer that share the same AK - remains non-verifiable. This effectively means that a RP must always rely on a certain authenticator's integrity rather than the particular user identity. This is further aggravated in case of the simple or Basic Attestation, because if the shared AK is compromised, there is no way of telling which users' identities are affected in consequence. On top of that, the loss of the AK is irrecoverable because of the static or prefabricated key management we have addressed in this paper. With that in mind, while this may already seem weak for human users, indirect identity verification is *certainly* insufficient for machines or IoT user objects.

Therefore, future work needs to establish a strong positive correlation between the aforementioned entities, as outlined in the previous section. To show that a "strong unique identities" approach does not contravene the principle of privacy protection, further research ought to provide evidence that no (negative) correlation ("breadcrumbs") attacks will be possible because of the protected way of communication and, therefore, transmission of said identities. That, however, can ideally be done by focusing on the encrypted data flow carrying

confidential data the same way an adversary would attempt it in an open network such as the Internet. This additional assessment of potential exposure to traffic analysis in IoT communication shall not only show that privacy leakage will not increase through positive correlation but also mitigate any residual gaps in the relevant ISO/OSI layers, e.g., transport through the application layer, which is clearly beyond the scope of this paper. Furthermore, with the previously discussed possible PUF integrations in mind, AI-based schemes could also be exploited in wider FIDO to scrutinise the hardware-focused elements of the overall attestation process.

The methodology that makes it possible to draw such insightful conclusions is pertinent to the promising research in metadata analysis[30,31], adaptive intrusion detection[32], and Machine Learning (ML)-focused PUF authentication methods[33,34]. Thus, a combination of novel protection mechanisms against information leakage in IoT/FIDO data through ML-enabled non-invasive metadata analysis and, likewise, ML-capable anomaly-based network intrusion detection with concept drift could form a solid self-recovery mechanism in our model. Self-recovery, in turn, has multiple facets to it, such that alongside privacy protection, detection of zero-day vulnerabilities through exposure of inconsistencies in communication protocols or their potentially erroneous implementations could be a highly interesting research angle as well. As a matter of fact, such or similar extensions would inherently follow the symbiotic security design rationale.

4.3. Computational costs and communication overheads

To outline the feasibility of the presented approach and highlight its efficiency, in this section, we will briefly shed light on the computational cost and the communication overhead of the suggested solution. One specific observation that allows us to draw a line and compare FIDO with ZKIE is the use of resource-constrained embedded devices: while FIDO hardware authenticators have no choice but to leverage microcontrollers due to the expected small size factor, ZKIE mainly targets IoT end devices which usually strive for ultra-low power characteristics and the overall small footprint including dimensions for better integration with larger appliances. Thus, the two realms are comparable in this regard to a large extent. In our hardware- and performance-centric discussion on "*Efficient System Design of Scalable Ultra Low Power Architectures with Symbiotic Security*"[19], we have already argued in favour of an LSI (Large Scale Integration) MCU with less power and memory consumption operated by an efficient use-case specific monolithic firmware as opposed to the more common VLSI (*Very* LSI) processors running a fully-fledged operating system. As an example, we had established that even a Raspberry Pi could be an overkill for average IoT applications (e.g., sensors or actuators) with its quad-core 64-bit ARM Cortex-A53 platform (3rd generation of Raspberry Pi). From the cryptographic perspective, we could evidence this assumption by comparing symmetric and asymmetric operations, such as AES-128-CBC and RSA-2048, executed on a RPi3 (using an efficient implementation of OpenSSL and Cortex-A53's cryptographic extension/acceleration) with those computed on a Cortex-M0 directly in hardware with a third option of pure software execution on a Cortex-M3. While the outcome of this test that the hardware-accelerated computation on Cortex-M0 beat the software-executed one on Cortex-M3 by a factor of 30 was to be expected, the results of the race between Cortex-A53 and Cortex-M0 were perhaps a little bit less intuitive: symmetric operations on Cortex-A53 were only 2.15 times faster, while the asymmetric ones had (only) a 6x speed advantage. However, to put things in perspective, even asymmetric operations only take seconds to complete, even on Cortex-M0, which, in real life for the majority of the IoT-relevant applications, is more than enough. From a larger perspective, the core dynamic power consumption of Cortex-M0 is, however, over 19.17 times lower than that of a Cortex-A9, which, in turn, is comparable to Cortex-A53 with respect to their nominal performance DMIPS/MHz value - not to mention that Cortex-M0 does not quite reach the operating frequency of the far bigger brothers (its core area is, for instance, 47.5 smaller compared to Cortex-A9). This clearly goes to show that the moderately increased performance requires a considerably disproportionate increase in power consumption and size.

In this context, we must also not forget the power of software optimisation, as can clearly be seen in the example of our own self-developed TLS middleware library for resource-constrained embedded devices called

Benchmark	pbTLS[2]	SRP
App. Data Usage (excl. TCP/IP overhead)	(minimum)[1] 4389 bytes	1440 bytes
Calculation Time (incl. average network transmission overhead)	300 ms	1740 ms
Power Consumption – Main Power Domain (96 MHz/3.3 V/ 25 °C)	300 ms x 16.72 mA = 5.02 As x 3.3 V = 16.5 Ws	1740 ms x 16.72 mA = 29.09 As x 3.3 V = 87.3 Ws
Power Consumption – Always On Power Domain (96 MHz/3.3 V/25 °C)	300 ms x 0.131 mA = 0.04 As x 3.3 V = 0.13 Ws	1740 ms x 0.131 mA = 0.228 As x 3.3 V = 0.752 Ws

1: Depending on the mode of operation, i.e. client or server, one-way or two-way TLS as well as the length of the certificate chain. Here: Server-side certificates only (chain length = 2), IoT module operates in client mode.
2: Tested Cipher Suite TLS_RSA_WITH_AES_256_CBC_SHA256 (0x003d)

Figure 6. Elapsed time, data usage, and power consumption in comparison for TLS and SRP.

pbTLS[35]: the initial reference implementation on a Cortex-M0 MCU consumed 30 s (which is not practical) to establish a TLS session based on the *TLS_RSA_WITH_AES_256_CBC_SHA256* cipher suite. Through various optimisation techniques, including dynamic memory management and efficient implementation of the underlying cryptographic primitives, we were then able to reduce the TLS handshake time to 500 ms. The low footprint of the pbTLS library of about 50 kByte in Flash memory and 500 bytes up to 5 kByte of RAM consumption alongside approximately 100 kByte of demo application (showcased in the video clip provided in the supplementary material) supports the claim that it is possible to achieve solid and efficient security on even the smallest devices.

With respect to the communication overheads, we depicted our results and measurements in Figure 6 for the sake of better overview and completeness. The results are based on the approach and methodology presented in our previous work[19]. The results show that the underlying SRP communication overhead (as part of the ZKIE data flow) is much less than that of a TLS session, which also makes it suitable for highly bandwidth-limited networks and carriers such as LoRaWAN, for instance. The average network transmission overhead (due to the necessary computation) is, however, higher compared to TLS, which also explains higher power consumption values. Ultimately, though, the findings do not indicate unbearable total costs when put in perspective of a usual IoT device communication bandwidth requirement in terms of its payload data and also considering that ZKIE primarily secures the bootstrap phase of IoT/FIDO device registration (as discussed in the previous sections) or the occasional over-the-air firmware update, while the bulk of the ensuing communication is secured through TLS.

5. CONCLUSIONS

This study sheds light on the critical challenges faced by PKI in ensuring the security of current FIDO specifications, particularly in terms of the initial enrolment of secure identities and keys. Through a comprehensive examination of the limitations inherent in the existing FIDO assumptions, we have presented several use case-driven combinations of a novel solution known as ZKIE and FIDO. This innovative approach offers a promising framework for managing the dynamic enrolment, including re-enrolment, of devices with their AKs in the context of the IoT.

To be able to do so, our research explored the practicality of the concept of *Symbiotic Security* within this domain, emphasising the potential for leveraging key aspects across all layers of implementation, such as hardware, software, network characteristics, and human operators, to enhance the ZKIE approach. To illustrate the feasibility and effectiveness of this concept, we have provided examples of noteworthy works from related fields

that could be integrated into this framework, thereby reinforcing its potential to achieve enhanced security in IoT environments.

Overall, this study contributes to the advancement of secure identity and key management in PKI and FIDO systems, highlighting the importance of addressing the challenges associated with the initial enrolment. By introducing the combined ZKIE-FIDO approach and exploring the concept of *Symbiotic Security*, we aim to stimulate further research and innovation in this area, fostering the development of more robust and resilient security mechanisms for the ever-expanding IoT landscape.

DECLARATIONS

Authors' contributions

Made substantial contributions to the conception and design of the suggested approach, conducted the security analysis and comparison with related work, co-authored the FIDO Device Onboard Specification [20] specification, co-developed the working prototype exhibited in the attached video clip, co-authored the SKMS scheme: Bartsch W

Performed a substantial review of the suggested approach and this paper, conducted editorial work, identified and described future development, co-authored the SKMS scheme, and provided administrative support: Millwood O, Kavun EB

Availability of data and materials

The video footage of the working prototype is made available as part of the supplementary material for this article.

Financial support and sponsorship

None.

Conflicts of interest

All authors declared that there are no conflicts of interest.

Ethical approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

REFERENCES

1. Thomas DR, Beresford AR. Better authentication: password revolution by evolution. In: Christianson B, Malcolm J, Matyáš V, Švenda P, Stajano F, Anderson J, editors. Security Protocols XXII. Springer International Publishing; 2014. pp. 130-45. DOI
2. Houshmand S, Aggarwal S. Building better passwords using probabilistic techniques. In: Proceedings of the 28th Annual Computer Security Applications Conference; New York, USA. ACM Press; 2012. pp. 109-18. DOI
3. Federal Office for Information Security. Handling passwords securely: a step-by-step guide. Available from: https://www.bsi.bund.de/EN/Themen/Verbraucherinnen-und-Verbraucher/Informationen-und-Empfehlungen/Cyber-Sicherheitsempfehlungen/Accountschutz/Sichere-Passwoerter-erstellen/Umgang-mit-Passwoertern/umgang-mit-passwoertern_node.html. [Last accessed on 8 Dec 2023]
4. Bartsch W, Hüebner M. Reference architecture for secure cloud based remote automation - Zero-knowledge initial enrolment of resource-constrained IoT with symbiotic security. *atp Magazin* 2019;61:72-82. DOI

5. Bartsch W, Gope P, Kavun EB, et al. Design rationale for symbiotically secure key management systems in IoT and beyond. In: Proceedings of the 9th International Conference on Information Systems Security and Privacy - ICISSP. SciTePress; 2023. pp. 583-91. DOI
6. FIDO Alliance. Certified authenticator levels. Available from: https://fidoalliance.org/certification/authenticator-certification-levels/. [Last accessed on 8 Dec 2023]
7. FIDO Alliance. FIDO UAF protocol specification: FIDO alliance review draft 28 November 2017. Available from: https://fidoalliance.org/specs/fido-uaf-v1.2-rd-20171128/fido-uaf-protocol-v1.2-rd-20171128.html. [Last accessed on 8 Dec 2023]
8. FIDO Alliance. FIDO security reference: FIDO alliance review draft 27 September 2017. Available from: https://fidoalliance.org/specs/fido-v2.0-rd-20170927/fido-security-ref-v2.0-rd-20170927.html. [Last accessed on 8 Dec 2023]
9. FIDO Alliance. FIDO TechNotes: the truth about attestation. Available from: https://fidoalliance.org/fido-technotes-the-truth-about-attestation/. [Last accessed on 8 Dec 2023]
10. Skorobogatov SP. Copy protection in modern microcontrollers. Available from: https://www.cl.cam.ac.uk/~sps32/mcu_lock.html. [Last accessed on 8 Dec 2023]
11. Brickell E, Camenisch J, Chen L. Direct anonymous attestation. In: Proceedings of the 11th ACM Conference on Computer and Communications Security. ACM; 2004. pp. 132-45. DOI
12. Whitefield J, Chen L, Sasse R, Schneider S, Treharne H, Wesemeyer S. A symbolic analysis of ECC-based direct anonymous attestation. In: 2019 IEEE European Symposium on Security and Privacy (Euro S&P); 2019 Jun 17-19; Stockholm, Sweden. IEEE; 2019. pp. 127-41. DOI
13. FIDO Alliance. FIDO ECDAA Algorithm. Available from: https://fidoalliance.org/specs/fido-v2.0-id-20180227/fido-ecdaa-algorithm-v2.0-id-20180227.html#dfn-ecdaa-issuer. [Last accessed on 8 Dec 2023]
14. Bleichenbacher D. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In: Annual International Cryptology Conference (CRYPTO); 1998. pp. 1-12. DOI
15. Pornin T. RFC 6979: Deterministic usage of the digital signature algorithm (DSA) and elliptic curve digital signature algorithm (ECDSA). DOI
16. Arciszewski S. No Way, JOSE! Javascript object signing and encryption is a bad standard that everyone should avoid. Available from: https://paragonie.com/blog/2017/03/jwt-json-web-tokens-is-bad-standard-that-everyone-should-avoid. [Last accessed on 8 Dec 2023]
17. Panos C, Malliaros S, Ntantogian C, Panou A, Xenakis C. A security evaluation of FIDO's UAF protocol in mobile and embedded devices. In: International Tyrrhenian Workshop on Digital Communication: Digital Communication. Towards a Smart and Secure Future Internet; 2017. pp. 127-42. Available from: https://link.springer.com/chapter/10.1007/978-3-319-67639-5_11#citeas. [Last accessed on 8 Dec 2023]
18. W3C. Web authentication: an API for accessing public key credentials Level 1. Available from: https://www.w3.org/TR/webauthn-1/. [Last accessed on 8 Dec 2023]
19. Bartsch W, Huebner M. Efficient system design of scalable ultra low power architectures with symbiotic security. *IEEE VLSI Circuits Syst Lett* 2019;5:4-11. Available from: https://ieeecs-media.computer.org/media/technical-activities/tcvlsi/newsletters/2019/ieee_vcal_vol_5_issue_4_nov_2019.pdf. [Last accessed on 8 Dec 2023]
20. Cooper G, Behm B, Chakraborty A, Kommalapati H, Mandyam G, Tschofenig H. FIDO device onboard specification. Available from: https://fidoalliance.org/specs/FDO/FIDO-Device-Onboard-RD-v1.0-20201202.html. [Last accessed on 8 Dec 2023]
21. German Federal Office for Information Security. Report on Microsoft Windows 8 and TPM. Available from: https://web.archive.org/web/20160304004000/https://www.bsi.bund.de/DE/Presse/Pressemitteilungen/Presse2013/Windows_TPM_Pl_21082013.html. [Last accessed on 8 Dec 2023]
22. Gassend B, Clarke D, van Dijk M, Devadas S. Silicon physical random functions. In: Proceedings of the 9th ACM Conference on Computer and Communications Security; New York, USA. 2002. pp. 148-60. DOI
23. Ángel Prada-Delgado M, Baturone I, Dittmann G, Jelitto J, Kind A. PUF-derived IoT identities in a zero-knowledge protocol for blockchain. *Internet Things* 2020;9:100057. DOI
24. Román R, Baturone I. A quantum-resistant and fast secure boot for IoT devices using hash-based signatures and SRAM PUFs. In: Nayyar A, Paul A, Tanwar S, editors. The Fifth International Conference on Safety and Security with IoT. Cham: Springer International Publishing; 2023. pp. 121-36. DOI
25. Gu C, Chang CH, Liu W, Hanley N, Miskelly J, O'Neill M. A large scale comprehensive evaluation of single-slice ring oscillator and PicoPUF bit cells on 28nm xilinx FPGAs. In: Proceedings of the 3rd ACM Workshop on Attacks and Solutions in Hardware Security Workshop; New York, USA. 2019. pp. 101-6. DOI
26. Tsiokanos I, Miskelly J, Gu C, O'neill M, Karakonstantis G. DTA-PUF: dynamic timing-aware physical unclonable function for resource-constrained devices. *J Emerg Technol Comput Syst* 2021;17:1-24. DOI
27. Sakib S, Rahman MT, Milenković A, Ray B. Flash memory based physical unclonable function. In: 2019 SoutheastCon; 2019 Apr 11-14; Huntsville, USA. IEEE; 2019. p. 1-6. DOI
28. Holcomb DE, Burleson WP, Fu K. Power-up SRAM state as an identifying fingerprint and source of true random numbers. *IEEE Trans Comput* 2009;58:1198-210. DOI
29. Armknecht F, Moriyama D, Sadeghi AR, Yung M. Towards a unified security model for physically unclonable functions. Available from: https://ia.cr/2016/033. [Last accessed on 8 Dec 2023]
30. Panchenko A. On the impact of cross-layer information leakage on anonymity in crowds. In: Proceedings of the 11th ACM Symposium on QoS and Security for Wireless and Mobile Networks; New York, USA. Association for Computing Machinery; 2015. pp. 35-42. DOI
31. Pennekamp J, Henze M, Hohlfeld O, Panchenko A. Hi doppelgäNger: towards detecting manipulation in news comments. In: Companion

Proceedings of The 2019 World Wide Web Conference; New York, USA. Association for Computing Machinery; 2019. pp. 197-205. DOI
32. Pasikhani AM, Clark AJ, Gope P. Adversarial RL-based IDS for evolving data environment in 6LoWPAN. *IEEE TIFS* 2022;17:3831-46. DOI
33. Najafi F, Kaveh M, Martín D, Reza Mosavi M. Deep PUF: a highly reliable DRAM PUF-based authentication for IoT networks using deep convolutional neural networks. *Sensors* 2021;21:2009. DOI
34. Millwood O, Miskelly J, Yang B, Gope P, Kavun EB, Lin C. PUF-phenotype: a robust and noise-resilient approach to aid group-based authentication with DRAM-PUFs using machine learning. *IEEE T Inf Foren Sec* 2023;18:2451-65. DOI
35. Bartsch W, Wuelbeck J. pbTLS Documentation. Available from: https://documentation.pointblank.de/. [Last accessed on 8 Dec 2023]

APPENDIX

This section contains additional material for ease of access, readability, and completeness, mainly consisting of the essential building blocks originally discussed in Bartsch *et al.*'s work[4].

Original Algorithm I Initial Enrolment in Secure Factory[4]

for $i \leftarrow 1$ to n = delivered modules **do**
    IoT_Node :: Deploy and establish connectivity;
    $(MatchingFirmware, CertKeyPair) \leftarrow$
        ↪ EnrolmentVM(NodeSRPAuthParams);
    IoT_Node :: Install $MatchingFirmware$, write $Cert$ to memory, write $KeyPair$ into secure memory;
end for
function EnrolmentVM(SRPParams)
    $AKVSessionID \leftarrow$
        ↪ AzureKeyVault(NodeSRPAuthParams, KeyID);
    Look up $NodeSRPAuthParams.NodeID$, $NodeSRPAuthParams.Salt$ and $NodeSRPAuthParams.Verifier$ in $SecureStorage$;
    SRP($NodeSRPAuthParams$);
    CA_VM($NodeSRPAuthParams.NodeID$, $AKVSessionID$);
    Open $SecureStorage$ using $AKVSessionID$;
    $CertKeyPair \leftarrow$
        ↪ (SecureStorage.NodeID.sk, SecureStorage.NodeID.pk, SecureStorage.NodeID.NodeCert);
    $MatchingFirmware \leftarrow$
        ↪ SecureStorage.NodeID.Firmware;
    IoT_Hub(NodeID, RootCACert, NodeCert);
    return $enc_{SecureChannelSessionKey}($
        ↪ MatchingFirmware, CertKeyPair);
end function
function AzureKeyVault(AuthParams, KeyID)
    Authenticate $AuthParams$;
    Open key vault $KeyID$;
    return $SessionID$;
end function
function SRP(NodeSRPAuthParams) ▷ Secure Remote Password Protocol
    Do SRP using $NodeSRPAuthParams$;
    return $SuccessFlag$, $SecureChannelSessionKey$;
end function
procedure CA_VM(NodeID, AKVSessionID) ▷ Certificate Authority
    Generate key pair $(NodeID.sk, NodeID.pk)$;
    Open $SecureStorage$ using $AKVSessionID$;
    $SecureStorage \leftarrow (NodeID.sk, NodeID.pk)$;
    Generate Certificate Signing Request; ▷ CSR
    $rootCASecretKey \leftarrow$
        ↪ $SecureStorage.Customer[NodeID]$.
        ↪ $rootCASecretKey$;
    $NodeCert \leftarrow$ (sign CSR with $rootCASecretKey$);
    $SecureStorage \leftarrow NodeCert$;
    Erase $rootCASecretKey$ from memory;
    Close $SecureStorage$;
end procedure
procedure IoT_Hub(NodeID, RootCACert, NodeCert)
    Install and register $(NodeID, RootCACert, NodeCert)$;
end procedure

Original Algorithm II Device Registration with the Azure Cloud[4]

Ensure: Successful initial enrolment
Require: Installed ($MatchingFirmware, NodeCert$)
1: Open two-way TLS channel using $NodeCert$;
2: DRS(RegistrationRequest, $NodeCert$);
3: **procedure** DRS(RegistrationRequest, NodeCert) ▹ Device Registration Service
4: Do device attestation using $NodeCert$;
5: Register device with $IoT\_Hub[RegistrationRequest.CustomerID]$;
6: **end procedure**
7: Close TLS channel;

Original Algorithm III Customer Device Bootstrap[4]

Ensure: Internet connectivity, utility installed, IoT module attached via UART/USB, registered customer
1: Log into $EnrolmentVM\_WebService$ through local utility via https and username/password (admin or developer access)
2: **for** $i \leftarrow 1$ to n = delivered modules **do**
3: Power on module
4: $NodeSRPAuthParams \leftarrow$ BootloaderSRP(createNodeSRPAuthParams); ▹ $NodeSRPAuthParams$ = SerialNumber, Salt, Verifier
5: $SecureStorage.NodeSRPAuthParams[i] \leftarrow \hookrightarrow NodeSRPAuthParams$;
6: BootloaderSRP(doSRP);
7: **end for**