Delegated Device Attestation for IoT

Jukka Julku, Jani Suomalainen, Markku Kylänpää
VTT Technical Research Centre of Finland Ltd.
Espoo, Finland
firstname.surname@vtt.fi

***Abstract*—Cybersecurity and trustworthiness of IoT devices are becoming more and more vital. Device attestation is one approach to increase trust through measured evidence of a device's software integrity. However, the lack of integration to different services still limits the adoption of device attestation in the IoT world. This paper describes integration of Device Identity Composition Engine (DICE) -based device attestation with delegated identity and access management. We present device attestation extensions to two existing authorization flows of the widely used OAuth 2.0 framework. These extensions do not alter the original flows, but they do include attestation evidence and result within the original message content, simplifying integration to existing systems. In addition, we describe our prototype implementation of the solution and present early performance results measured using a physical IoT device. Furthermore, we discuss the benefits and challenges of the approach. The goal of this work is to ease the adoption of device attestation by simplifying integration to existing IoT systems.**

***Index Terms*—Cybersecurity, IoT, Remote Attestation, Identity Management, Access Control, OAuth 2.0.**

I. Introduction

Trustworthiness of the Internet of Things (IoT) is becoming more and more vital for society, industry, and business. Different cyber threats — such as discontinued business due to ransomware, network and service unavailability due to botnet disturbances, and accidents due to compromised machinery and vehicles — motivate the adoption and development of stronger security solutions for connected devices and connecting networks. There is a need for technologies that assure trustworthiness (i.e., the high probability that devices behave as expected) and enable trust (i.e., the acceptance of the small risk that devices misbehave). However, challenges for securing IoT are various. The heterogeneous device landscape means that security management and device updates are more challenging. Users often have limited experiences, skills, and resources for security configurations, making security error-prone. Solutions need to be user-friendly, scalable, and suitable for resource-restricted devices. At the same time there is a need for security solutions that support new emerging applications and can address previously unknown security threats.

The security and trustworthiness of an IoT device depends on various factors, including the strength and maturity of applied protocols, security architectures, and standards [1]; security certification and tests [2] during manufacturing and procurement; and integrity or tamper protection of the executed applications by access control and anti-virus solutions. Device attestation — as specified by the Trusted Computing Group in Device Identifier Composition Engine (TCG DICE) [3] — is a security control for assuring the trustworthiness of IoT devices and the software they are running. Device attestation is an IoT adaptation of the remote attestation [4], [5] concept. It proves to service providers that the device is running unmodified whitelisted and certified software. It lowers the risks of unexpected malicious behavior and thus enables servers to make a decision to trust and to provide the service for the device. However, even though many current IoT devices already have capabilities to support attestation, it has not been widely adopted or integrated into IoT systems.

The security of large IoT-based systems depends on the solutions for managing identities, access permissions, and software configuration of fleets of devices. Identity and access management (IAM) is a collection of policies and technologies ensuring that the correct devices have the appropriate access to resources and services in the network. For IoT devices there is currently no single IAM alternative [6] but several alternatives that have been adopted by different vendors and platforms [7]. The most common IAM standard for web and cloud services and, due to interoperability, a prominent alternative also for IoT, is the OAuth 2.0 authorization framework [8]. OAuth 2.0 enables a device to securely obtain limited, delegated access to a server resource on behalf of the resource owner without sharing credentials with the device. OAuth 2.0 enables, for example, that a user who is managing a private network can authorize a new sensor device to access a network service by using a web browser or a mobile phone application, as we demonstrated in a field trial [9]. Some efforts [10], [11] have enhanced IAM with remote attestation but have not focused on the requirements of IoT systems.

Our novel contributions are the following. We combine device attestation with an IAM system for IoT devices. We specify how the existing OAuth 2.0 authorization framework can be enhanced with DICE-based device attestation. The achieved delegation-oriented architecture provides a scalable approach to manage trustworthiness in distributed environments with a large amount of sensitive network resources. We describe a proof-of-concept implementation and performance analysis of device attestation-enhanced identity management for a microcontroller class device. We analyze and discuss opportunities, challenges, and trustworthiness of the resulting architecture for identity, access, and trustworthiness management.

This work was supported by the Business Finland and the consortium partners of the PRIORITY project.

The rest of this paper is structured as follows. First, in Section II we describe and discuss our modifications to incorporate device attestation into the OAuth 2.0 authorization framework. In Section III we present a proof-of-concept implementation of our design and discuss the performance impact of the modifications. Then, in Section IV we critically evaluate our approach and its constraints. We also discuss advantages and potential use cases. In Section V we cover related research and the differences to our work. Finally, in Section VI we conclude our paper and propose directions for future research.

II. Approach

This section describes our approach for managing identities, access, and trustworthiness of IoT devices. The approach combines the OAuth 2.0-based identity management and device attestation, so that verification of device integrity measurements is delegated from individual services to the identity management infrastructure, while the services are given means to use the attestation result for their application-specific needs. We first describe device attestation as a background and then explain our extensions to two existing OAuth 2.0 authorization flows — i.e., means to acquire access tokens — to incorporate remote attestation. The selection of the two protocols is based on their suitability for IoT devices. They use X.509 certificate-based authentication rather than, e.g., passwords or other methods better suited for human users.

A. Device Attestation

In remote attestation [4], [5] an *attester*, e.g., a device, produces *evidence*, i.e., claims of its identity and trustworthiness, for a remote party to support a decision process, e.g., an authorization decision. Most often, this *relying party* does not have the means to assess the evidence directly, but the task is given to a trusted *verifier* that appraises the evidence according to *endorsements* of evidence authenticity and acceptable *reference values*, producing an *attestation result* that the relying party then uses according to its security policies. Typically, the evidence consists of boot time integrity measurements of loaded software components, and the reference values are hashes of known good software for the device. The reference values could be provided, e.g., by a trusted device manufacturer. Similarly, the manufacturer could vouch for the device's capability to securely collect and sign the evidence.

In our approach, remote attestation is based on TCG's DICE hardware root-of-trust and Implicit Identity Based Device Attestation specification [3]. DICE assumes that the device has a statistically unique device-specific secret (UDS) value that is used to deterministically derive two asymmetric key pairs at the device boot. The first key pair, called the *device ID*, acts as the device identity. The private key never leaves the trusted bootloader and cannot be accessed by the device firmware to ensure that the identity cannot be cloned. The device certifies the second key pair, called the *alias key*, using the device identity key pair. The certificate includes the integrity measurement of the device firmware so that it can be used as evidence of device integrity. Typically, an X.509 certificate is used and the measurements are encoded as certificate extensions. If an attacker is able to modify or change the device's firmware, they cannot forge the integrity evidence without the device ID private key. The alias key credentials can be exposed to the device firmware for further use, e.g., as TLS client credentials. Furthermore, a trusted party such as the device manufacturer may certify the device ID public key. The certificate then functions as an endorsement of integrity of both the device's identity and integrity measurement function.

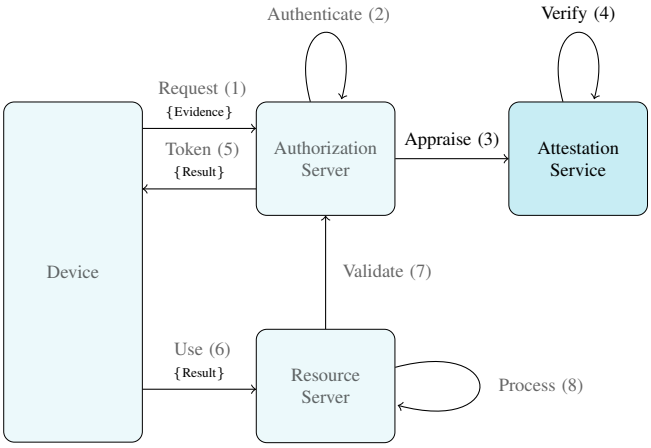

Fig. 1. Mutual-TLS based authorization flow with attestation

B. Device Authorization

In the OAuth 2.0 standard [8], client authentication and authorization decisions are delegated to a centralized *authorization server*. A client that intends to perform an operation on a *resource* hosted on a service, called *resource server*, requests an *access token* from the authorization server and presents it as proof of authorization to the resource server. The standard defines various *authorization flows* that specify concrete steps for different interaction protocols between the client, the resource owner, and the authorization server to obtain the access token.

We expect the device to use X.509 certificates and mutual-TLS authentication-based authorization flow according to RFC-8705 *"OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens"* [12] in order to retrieve the token. We extend this flow by device attestation so that the device provides evidence of its integrity to the authorization server as proof of a device's security state, and the resulting attestation result is included in the access token.

The authorization flow is illustrated in Figure 1. To highlight our modifications, the original elements are expressed in gray whereas the new elements are expressed in darker colors. Also, the new message content is stated for each message explicitly in curly brackets. The modifications are as follows.

When the device requests an access token from the authorization server **(1)**, it uses the DICE alias key credentials as its TLS client credentials. As explained before, the certificate carries evidence of device's integrity. The authorization server

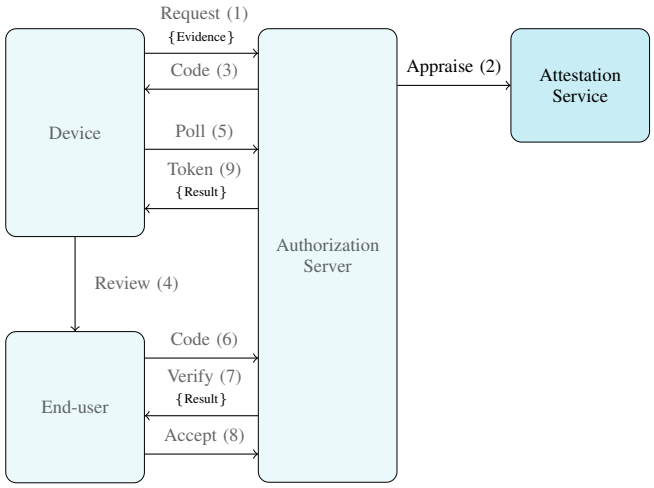

Fig. 2. Device authorization flow with attestation

compares the certificate to its database of registered DICE device ID certificates to authenticate the device during the TLS handshake **(2)**. If the authentication is successful and the DICE certificate extensions are detected, it requests an attestation service to verify the attestation evidence **(3)**. The attestation service verifies device integrity by validating the certificate and comparing embedded integrity measurements against its reference integrity metric database according to a security policy **(4)**. The service issues an attestation token containing the attestation result and returns it to the authorization server. The authorization server issues an OAuth 2.0 access token to the device and embeds the attestation token into it as an additional claim about the client **(5)**.

The device receives the access token and uses it to request some operation on the resource server **(6)**. The resource server requests the authorization server to verify the validity of the access token, including the embedded attestation token **(7)**. The resource server then processes the device's request **(8)**. It may use the attestation token to make application-specific decisions on the trustworthiness of the device and the data the device provides.

C. Device Registration

Before the mutual-TLS based authentication and authorization flow presented in the previous subsection can be used, a trust relationship must be established between the device and the authorization server. That is, the device's identity and the corresponding credentials, i.e., the device ID certificate, must be registered with the authorization server. There are multiple options on how to achieve this. Here, we discuss one such method for end-user assisted security bootstrapping of devices.

Our method is based on the RFC-8628 *"OAuth 2.0 Device Authorization Grant"* [13] that allows an input-constrained device to request for end-user authorization to access a protected resource. This flow uses an end-user code that the device passes to an end-user as an identifier of a pending authorization request on the authorization server, which the end-user then approves. The device uses the flow for authorization to register its identity with the authorization server. The registration procedure is implemented as a separate resource server, called a registration server. We extend the original flow with device attestation to strengthen the resource owner's trust toward the device that is registered. The flow, including our extensions, is illustrated in Figure 2. The modifications are as follows.

The device requests access from the authorization server using OAuth 2.0 Device Authorization flow **(1)**. It uses the DICE alias key certificate in the same way as in the mutual-TLS based authorization flow. However, since the authorization server doesn't have an established trust relationship with the device yet, that is, it does not possess the corresponding device identity certificate, it cannot authenticate the device at this point. The server proceeds with the integrity verification **(2)** and stores the resulting attestation token for further use. It then issues device and end-user codes and provides them with the end-user verification URI to the device **(3)**.

Following the original flow, the device instructs the end-user using an out-of-band method, e.g., the device's display or near-field communication, to review the authorization request on another device **(4)**, and then starts repeatedly polling **(5)** the authorization server for the access token. The end-user visits the provided verification URL and authenticates to the authorization server with their credentials, **(6)**, e.g., using a mobile device.

The authorization server prompts **(7)** the end-user to accept or decline the authorization request. It retrieves the device's attestation token that was stored earlier and includes it, with any other device information, in the confirmation prompt. The end-user reviews the request and either accepts or declines it **(8)**. In addition to the attestation status, the end-user may, e.g., visually verify that the device matches the information given by the attestation service. If the device is granted access, the authorization server replies with an access token to its next request matching the authorized device code **(9)**. The attestation token is embedded in the access token in the same way as in the mutual-TLS flow.

When the device has retrieved the access token, it uses the token to register its device ID certificate with the registration server. In order to ensure that the device registers the same identity that was attested and presented to the end-user, the server verifies that the certificate matches that recorded in the attestation token. After this step, the device is able to request access tokens independently without end-user assistance.

All communication between the device and the authorization server is protected by TLS, including the end-user code. In addition, the access token is bound to the TLS client certificate and the device identity used during the attestation is encoded in the attestation token embedded in the access token. Thus, assuming that the device is not compromised, and that the end-user can trust that the end-user code provided via an out-of-band method comes from the specific device, an external attacker cannot divert the flow to register some other device instead. Moreover, the registration service may verify that the registered device matches the device approved by the end-user.

III. Evaluation

We implemented a proof-of-concept of the protocol extensions described in Section II on a micro-controller class device. We then measured the communication delays and the amount of data transmitted during the device authorization protocol in order to evaluate the performance overhead due to device attestation.

A. The Proof-of-Concept Implementation

The proof-of-concept IoT device is based on a Nuvoton M2351 class [14] micro-controller with ARM TrustZone-M technology [15] that allows hardware-enforced isolation of the platform into secure and non-secure domains. TCG DICE has been implemented as early boot code in the secure domain and utilizes cryptographic accelerator for hash and asymmetric cryptography calculations. Certificate operations are implemented using the mbedTLS library [16]. The micro-controller does not include device secret based on either eFuse or Physically Unclonable Function, but there is a unique identifier that is used to derive the DICE UDS instead. However, a proper implementation would also require the ability to limit visibility of the secret. Nevertheless, even though these shortcomings impact the security of the proof-of-concept implementation, they do not dilute the value of our main work: the protocol extensions.

The device's firmware layer is implemented as an application on top of the Mbed OS [17] operating system, which runs in the non-secure domain. DICE alias key credentials are exposed to the device application over TrustZone-M Non-Secure Callable functions and used as TLS client credentials for all communication. The application performs device registration at the first boot. After the boot it proceeds to periodically sending sensor measurements from a temperature and relative humidity sensor to an IoT service. The device requests an access token separately for each measurement cycle.

The authorization server is implemented on top of the IdentityServer4 [18] OpenID Connect and OAuth 2.0 framework. The framework already provided implementations of both the authorization flows we aimed to extend. The main modifications are as follows:

- A new client store for tracking registered devices, including their device ID certificates, and their access rights.
- A new certificate validation scheme that detects DICE alias key certificates based on their X.509 extensions and validates them against the new client store.
- Hooks for the two authorization flows to trigger verification of integrity evidence and embed the attestation token claim into the access token before signing the token.
- Modifications to store the attestation result during the device authorization flow and present it to the end-user. The end-user view of the attestation information is presented in Figure 3.

We implemented a separate attestation service that provides verification procedures for DICE alias key certificates. It maintains a reference integrity metrics database that records

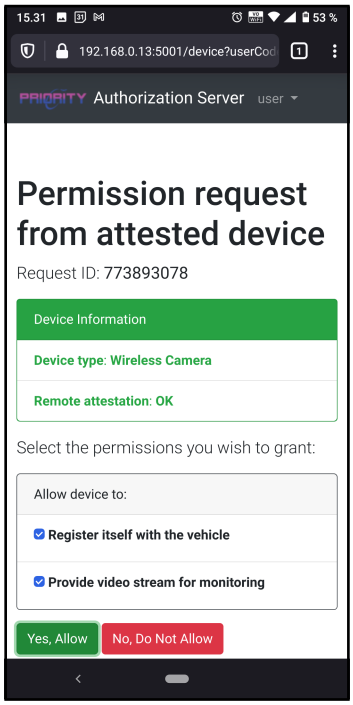

Fig. 3. The mobile UI for authorizing attested devices

information about valid device firmware, such as their target device types, firmware hashes, and versions. Attestation tokens generated by the service use the JSON format and contain information such as the device identifier, an event timestamp, the device type, the firmware version, and the attestation status.

Both the registration service and the IoT service are simple HTTP-based resource servers that are implemented using facilities provided by the IdentityServer4 framework. While the registration service is mainly a simple interface that directly modifies the authorization server's client store, the IoT service maintains a database of measurement events from different sensors. It uses the attestation token to record the security state of the device at the time of each measurement.

B. Performance Evaluation

We measured the communication delay and the amount of bytes transmitted between the device and the services to approximate the performance impact of our approach. Since device registration is expected to be a rare operation compared to device authentication and authorization, in this paper we limit our examination to the latter. The performance impact of the former is expected to be similar.

In the test setting, all the services reside in the same virtual machine and communicate locally. The physical IoT device connects to the services over a Wi-Fi access point. Communication delays were measured, using the network monitoring tool Wireshark [19] in milliseconds between TCP packets belonging to the same operation. The main reason for using a less accurate external measuring tool was the difficulty in integrating the measurement code within the device firmware, especially the TLS library, which limited the accuracy of the

measurements on the device itself. The overhead caused by the device joining and leaving the Wi-Fi network was omitted from the measurements as we wanted to concentrate on the performance impact of the protocol extension itself.

The integrity evidence is delivered as part of a TLS handshake. On the other hand, the verification of the evidence is triggered by the OAuth 2.0 protocol messages exchanged after the TLS connection has been established, i.e., the handshake is completed. According to our measurements, the TLS handshake took on average 11 127 milliseconds between the device and the authorization server and 11 148 milliseconds between the device and the IoT service, respectively. After the connection had been established, the rest of the communication took on average 56 milliseconds and 22 milliseconds for these two cases, respectively. The communication between the authorization server and the attestation service took a total of 28 milliseconds on average. Thus, verification of the evidence took around half of the processing time for the token request.

Based on these measurements, the TLS handshake overhead overwhelmingly dominates the communication delay. We expect this behavior is explained by our low-end device's weak performance on asymmetric cryptography taking place during the handshake, rather than the general processing requirements, e.g., due to the exchanged data. In comparison, the device sent on average of 1433 bytes and received 2093 bytes during the TLS handshakes, while the average number of bytes sent and received during the HTTP message exchange were 740 and 3896 bytes in the first case and 1478 and 283 bytes in the latter case. Hence, the difference in the amount of bytes is considerably smaller than in the communication delay. The integrity measurements add around 158 bytes to the device certificate size and our implementation of the attestation token is about 200 bytes long, resulting in around 267 bytes of extra data within the base64-encoded access token. Thus, compared to the original OAuth 2.0 flows, the impact of our extension to the message size is relatively small.

Even though the measured behavior is specific to our choice of device hardware, it indicates that the IoT device's ability to perform a TLS handshake in the first place is most likely the limiting factor for using our approach. However, the handshake already exists in the original protocol and our extensions only slightly increase the amount of data transmitted. The delegation of authentication and authorization to the authorization server means an additional TLS handshake must be performed compared to interacting only directly with the IoT service, effectively doubling the performance cost. Thus, the approach is not suitable for the most constrained devices.

On the other hand, the OAuth 2.0 access tokens are not single-use tokens. If the device is able to store the access token, the communication with the authorization server doesn't need to be performed each time the device needs to access the IoT service. However, lifetime of the access token impacts security. Moreover, device attestation must be performed at least after each device boot as the integrity measurements may change at that point.

IV. Discussion

In this section we first discuss the benefits and challenges of our approach. Then we discuss some potential use cases for the presented solution, and especially the use of the attestation token in the services. Finally, we explore the potential pitfalls of our work and observations presented in this paper.

A. Benefits and Challenges

The approach presented in Section II has various advantages. First of all, the solution delegates the verification of device integrity to the attestation service and thus provides similar benefits through the separation of concerns for remote attestation as the OAuth 2.0 frameworks provides for authorization: simplicity of service implementation and configuration management, as well as scalability, as the mechanisms can be distributed. Moreover, the solution re-uses a mature framework already used in many cloud environments.

IoT applications are often supported by edge computation-based architectures where resources and functions are in the servers, which are in close proximity to the devices. The edge architecture complicates the establishment of end-to-end trust relationship [20] as the services and access management functions may be hosted in different locations. Our delegation-based architecture provides flexibility to the location of services but does not increase trust towards edge resources.

The attestation service does not have to be hosted locally by the identity management infrastructure, but it could also be provided, e.g., by the IoT device manufacturer or another trusted third party. In fact, the authorization server may delegate attestation requests to multiple different attestation services, e.g., hosted by different device manufacturers, based on the information included in the certificate. In this case, the resource server owner must trust not only the identity management provider, but the attestation service provider to which the verification of the attestation evidence is delegated. In this sense, the attestation services provided, e.g., by different device manufacturers, compare to external identity providers of OAuth 2.0 architecture. The device registration phase corresponds to the first synchronization of user identity from the external identity provider.

Embedding the attestation token into the OAuth 2.0 access token offers flexibility of use. The attestation result can be used not only for the authorization decision in the authorization server, but also by the resource servers according to their own security policies or application-specific needs. For example, in some cases attestation could be a mandatory requirement for obtaining an access token for certain security or application critical functions. On the other hand, resource servers could use the attestation token as verified evidence of a device's type.

The approach allows simplified integration into existing systems, as it requires no modifications to the authentication and authorization flow from the client's or the resource server's perspective. On the contrary, both the integrity evidence and the attestation token are embedded in the existing protocol messages, that is, within the X.509 certificates and OAuth 2.0 access token, respectively. Both of these formats support

extensions. The only additional step to the protocols is the communication between the authorization server and the attestation service, which is hidden behind the interface of the authorization server. Thus, the same mechanism can be used with devices, authorization servers, or resource servers that do not support our remote attestation solution. Rather, the solution allows remote attestation-capable devices to provide additional evidence of their trustworthiness to attestation token-aware services. In fact, due to the heterogeneity of IoT devices and the lack of remote attestation support in the existing products, this kind of mixed mode operations is likely.

To enable application-specific exploitation of the benefits of the attestation token, the claims included in the token should be meaningful to different types of resource servers. Our implementation uses a custom format, but for interoperability a well-known encoding format, such as the Entity Attestation Token [21], should be used.

Since one of the main motivations of our approach is to delegate the attestation process to a trusted third party, and relieve the resource servers from the burden of keeping track of the device configurations, the claims should contain rather high-level properties of the device. That is, it should allow the resource servers to utilize property-based attestation [22]. Such properties could include, as an example, security-related claims, such as a high-level integrity state or claims about trustworthiness of the device, or its capability to keep secrets. In addition, the token could be extended to convey, e.g., manufacturer-vouched claims about the data the device produces and how the data should be interpreted, such as the type and precision of measurements or trustworthiness of the measuring process.

As our measurements show, X.509 certificate-based TLS which uses asymmetric cryptography can cause too much overhead for the most constrained devices. A more recent TCG DICE specification [23] introduces symmetric key cryptography and TLS-PSK-based authentication to provide a device attestation mechanism that could be more suitable for these devices. This mechanisms could also be integrated with OAuth 2.0 as part of our protocol extensions.

As another performance issue, the size of the attestation token could also become an obstacle for the most constrained IoT devices. The token should be relatively small to avoid any processing and memory restrictions on the device. The device must be able to forward the access token to a resource server and store it for the lifetime of the token. On the other hand, the token content should be generic enough to support multiple use cases. These two requirements rarely go well hand-in-hand. At minimum, the token could contain the device's identity and an overall claim about its integrity state. It could also contain, e.g., the model and version of the device that the resource servers could then use to look up device information. Verbose information could also be included indirectly, e.g., as a URI to a remote web service hosting the information for the lifetime of the token. Resource servers could then fetch the information from the server as needed, without needing to burden the resource constrained device with the data exchange.

The attestation token conveys information about the target device, and thus can cause privacy issues if given to less trusted resource servers: how can the user be enabled to control to whom the attestation information is given? A potential solution to this can be implemented in the phase where the access token is retrieved. The client could be given control on what type of attestation information can be embedded into the access token for which resource servers.

Furthermore, even though the attestation token is protected from eavesdroppers by TLS encryption, in older TLS versions, the client certificate might be sent unencrypted during the TLS handshake. In that case, an eavesdropper could extract the integrity evidence and potentially use it to determine information about the device's security state, e.g., if the device is running vulnerable firmware.

B. Use Cases

Device attestation opens up various opportunities for cybersecurity hardening. Potential use cases and services include:

- Attesting the integrity of each connecting thing protects the integrity of the network and its services. We demonstrated [9] our device attestation prototype within a trial for a research project that focused on mission- and business-critical communications, e.g., for public safety and smart farming.
- Knowledge that data has been collected from a trustworthy source can be used as a data quality attribute in a system with mixed-trust devices only some of which have a mechanism to prove their integrity. Data collection from trustworthy sources increases the accuracy and reliability of results and decisions. For instance, an AI could be allowed to make actions without human involvement if the risk of data poisoning is minimized by assuring trustworthiness of data sources. Data received from an attestation-capable device could also be used to assess the plausibility of data from the less trusted devices that are, e.g., physically close to the more trusted device.
- The fewer unverified devices there are in a network, the less likely the network is infected by a botnet and, e.g., will launch a denial-of-service attack or excessively consume user resources. In critical networks, *intelligent routing* decisions can be made using this attestation-based trust: critical communication can be forwarded to network segments and routes that are more trusted. A network could also prioritize data flows from devices that are more trusted.
- Attesting what a device is — e.g. a sensor, camera, or drone — enables simplification of user interactions. Instead of specifying policies for each registered device, authorization policies can be defined only once for the type or role of devices. Hence, instead of authorizing each device to use particular services or making role assignments, new attested devices can be implicitly authorized. Showing and potentially visualizing the type of attached device instead of just showing a random alphanumeric identifier can minimize the number of end-user mistakes.

C. Threats to Validity

In this paper, our aim is to develop easy integration of existing device attestation mechanisms to an identity management framework and our main contributions are the protocol extensions and the ways that the services can use the attestation result. In particular, we do not propose new device attestation or integrity verification mechanisms.

As our measurements show, our choice of identity management framework is not suitable for the most constrained IoT devices due to performance overhead, which means that the proposed approach is not suitable for all IoT use cases.

Moreover, our choice of device hardware for the proof-of-concept implementation is not optimal considering performance measurements. The choice was due to the availability of suitable devices for this research. The device's poor performance on asymmetric cryptography disturbed performance measurements: on the one hand it shows that the proposed protocol is clearly not suitable for the most constrained devices. However, at the same time the results cannot be easily generalized to less constrained devices. New measurements with such a device are needed.

In addition, we measured performance only as the communication delay and the amount of data exchanged between the device and the servers. However, in many IoT applications battery lifetime, that is energy consumption, might be a more relevant performance measure.

V. Related Work

The idea of remote attestation for embedded, resource restricted devices is not new. It has been studied, e.g., in the context of mobile devices [24], wireless sensor networks [25], IoT [26], and device swarms [27]. Existing work often concentrates on fundamental aspects of remote attestation, such as the threat model (assumptions), evidence acquisition (hardware or software), integrity measurement (static or dynamic), and the interaction patterns between the participants (e.g., one-to-one, many-to-many), and scalability. That is, these efforts often concentrate on the parts of the attestation process that are common for multiple use cases — the device identity, integrity measurement, and common primitives for attestation — and leave out the use case or domain specific part — the remote attestation protocol. Our work builds upon this body of work and takes one step toward providing a generic remote attestation protocol for various use cases and simplifying the adoption of remote attestation for existing systems.

Many of the above-mentioned solutions base their evidence acquisition on traditional Trusted Platform Module (TPM) hardware, which is often too expensive a solution for constrained devices, or software-only approaches [28] that impose no or fewer requirements on the device hardware, potentially fitting low-cost IoT devices and legacy devices better, but which cannot offer strong security guarantees. We take the middle road and utilize DICE [3], which is a lightweight hardware-based option already supported by many IoT class micro-controllers.

The concept of DICE has drawn interest in IoT security community in recent years. For instance, Jäger et al. [29] analyze its suitability to IoT devices. DICE* is formally verified DICE implementation integrated into the boot firmware of a microcontroller [30]. DICE is also used in CIDER [31], which introduces Trusted Computing primitive called *dominance* that, in addition to the detection of integrity violation by remote attestation, adds unconditional recovery control for the remote party. Huber et al. [32] build on this work and propose the concept of DICE++, which enables updating the device's trusted computing base without losing devices identity. This research could be combined with our work.

We are not the first to propose the integration of remote attestation into identity management infrastructure. For example, Ali et al. [10] propose an extension to a federated identity management system where the identity provider vouches for the client's platform integrity in addition to the client's identity. The integrity information is then used for access control decisions. They describe a design and implementation of the concept for Shibboleth single sign-on architecture. Also, the work of Leicher et al. [11] has many similarities to our work. They discuss how integrity verification could be used as an authentication method in OpenID Connect, an authentication layer on top of OAuth 2.0, to bind trustworthiness of the device to an identity. They also mention signaling integrity verification result to the user of the authenticated identity, which resembles our ideas with the attestation token. Neither of these works target IoT devices. Rather, they discuss attestation of more traditional TPM-based platforms and end-user access to resources, e.g., websites. In addition, their solutions require substantial modifications to the original protocol, whereas our approach requires only minimal modifications. Moreover, our focus is more generic than just authentication and authorization, as our aim is to also enable the resource servers to use the attestation results in service-specific ways.

Furthermore, work has been done to integrate remote attestation capabilities into other existing communication protocols to secure embedded devices. For example, Industrial IoT (IIoT) is an area where the use of Trusted Computing platforms as key storage and also as integration to IIoT protocols is emerging. Both Birnstill et al. [33] and Bienhaus et al. [34] propose TPM-based remote attestation enhancement to industrial automation protocol OPC UA. They also use TPM as a cryptoprocessor and secure key storage.

VI. Conclusion

In this paper, we presented extensions to the widely used OAuth 2.0 identity and access management framework that allows IoT devices to prove their integrity using remote attestation as part of a delegated authorization protocol. The verification of the attestation evidence is orchestrated by the authorization server and the integrity evidence, and the attestation result are embedded in the existing protocol messages without the need to change the original protocol flow. As a result, the IoT services only have to worry about using the attestation result for their application-specific needs, but not

about the attestation mechanism. In addition, we described the proof-of-concept implementation and early performance measurements of the proposed approach.

Our solution can co-exist with devices and services that do not support remote attestation, allowing easy integration to existing systems, hopefully simplifying the adoption of attestation of IoT devices as a security practice. Such strong security mechanisms will be needed more and more in the future to cope with the security challenge that our increasing dependence on various types of IoT systems creates.

We discussed some potential ways that a service could use the attestation information, e.g., as data quality attributes in mixed trust systems. Such use cases offer an interesting research direction in enhancing trust to IoT applications and the data they provide.

REFERENCES

[1] E. Lee, Y.-D. Seo, S.-R. Oh, and Y.-G. Kim, "A survey on standards for interoperability and security in the Internet of Things," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1020–1047, 2021.

[2] S. N. Matheu, J. L. Hernández-Ramos, A. F. Skarmeta, and G. Baldini, "A survey of cybersecurity certification for the Internet of Things," *ACM Computing Surveys (CSUR)*, vol. 53, no. 6, pp. 1–36, 2020.

[3] Trusted Computing Group, "Implicit Identity Based Device Attestation," Trusted Computing Group, Reference Version 1.0, revision 0.93, March 5 2018.

[4] G. Coker, J. Guttman, P. Loscocco, A. Herzog, J. Millen, B. O'Hanlon, J. Ramsdell, A. Segall, J. Sheehy, and B. Sniffen, "Principles of Remote Attestation," *Int. J. Inf. Secur.*, vol. 10, no. 2, pp. 63–81, jun 2011.

[5] H. Birkholz, D. Thaler, M. Richardson, N. Smith, and W. Pan, "Remote Attestation Procedures Architecture," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-rats-architecture-11, March 2021, accessed: September 2 2021. [Online]. Available: https://www.ietf.org/internet-drafts/draft-ietf-rats-architecture-11.txt

[6] A. Sfakianakis, C. Douligeris, L. Marinos, M. Lourenço, and O. Raghimi, "ENISA Threat Landscape Report 2018: 15 Top Cyberthreats and Trends," *DOI*, vol. 10, p. 138, 2019.

[7] M. Ammar, G. Russello, and B. Crispo, "Internet of Things: A survey on the security of IoT frameworks," *Journal of Information Security and Applications*, vol. 38, pp. 8–27, 2018.

[8] D. Hardt, "The OAuth 2.0 Authorization Framework," Internet Requests for Comments, IETF, RFC 6749, October 2012.

[9] J. Suomalainen, J. Julku, M. Vehkaperä, and H. Posti, "Securing public safety communications on commercial and tactical 5G networks: A survey and future research directions," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 1590–1615, 2021.

[10] T. Ali, M. Nauman, M. Amin, and M. Alam, "Scalable, privacy-preserving remote attestation in and through federated identity management frameworks," in *2010 International Conference on Information Science and Applications*, 2010, pp. 1–8.

[11] A. Leicher, A. U. Schmidt, Y. Shah, and I. Cha, "Trusted Computing enhanced OpenID," in *2010 International Conference for Internet Technology and Secured Transactions*. IEEE, 2010, pp. 1–8.

[12] B. Campbell, J. Bradley, N. Sakimura, and T. Lodderstedt, "OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens," Internet Requests for Comments, IETF, RFC 8705, February 2020.

[13] W. Denniss, J. Bradley, M. Jones, and H. Tschofenig, "OAuth 2.0 Device Authorization Grant," Internet Requests for Comments, IETF, RFC 8628, August 2019.

[14] Nuvoton Technology Corporation, *NuMaker-PFM-M2351 User Manual*, Nuvoton Technology Corporation, 2018.

[15] Arm Limited, "Arm® TrustZone Technology for the Armv8-M Architecture," Arm Limited, Cambridge, England, Tech. Rep. Version 2.1, October 2018.

[16] Linaro Limited. Mbed TLS - Trusted Firmware. Web-page. Accessed: August 23 2021. [Online]. Available: https://www.trustedfirmware.org/projects/mbed-tls/

[17] Arm Limited. Mbed OS | Mbed. Web-page. Accessed: December 1 2020. [Online]. Available: https://os.mbed.com/mbed-os/

[18] IdentityServer. IdentityServer. Web-page. Accessed: December 1 2020. [Online]. Available: https://identityserver.io/

[19] The Wireshark team. Wireshark · Go Deep. Web-page. Accessed: August 23 2021. [Online]. Available: https://www.wireshark.org/

[20] R. T. Tiburski, C. R. Moratelli, S. F. Johann, M. V. Neves, E. de Matos, L. A. Amaral, and F. Hessel, "Lightweight security architecture based on embedded virtualization and trust mechanisms for IoT edge devices," *IEEE Communications Magazine*, vol. 57, no. 2, pp. 67–73, 2019.

[21] G. Mandyam, L. Lundblade, M. Ballesteros, and J. O'Donoghue, "The Entity Attestation Token (EAT)," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-rats-eat-09, March 2021, accessed: September 2 2021. [Online]. Available: https://www.ietf.org/internet-drafts/draft-ietf-rats-eat-09.txt

[22] A.-R. Sadeghi and C. Stüble, "Property-based attestation for computing platforms: Caring about properties, not mechanisms," in *Proceedings of the 2004 Workshop on New Security Paradigms*, ser. NSPW '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 67–77.

[23] Trusted Computing Group, "Symmetric Identity Based Device Attestation," Trusted Computing Group, Reference Version 1.0, revision 0.95, January 7 2020.

[24] N. Asokan, J.-E. Ekberg, K. Kostiainen, A. Rajan, C. Rozas, A.-R. Sadeghi, S. Schulz, and C. Wachsmann, "Mobile trusted computing," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1189–1206, 2014.

[25] R. V. Steiner and E. Lupu, "Attestation in wireless sensor networks: A survey," *ACM Comput. Surv.*, vol. 49, no. 3, Sep. 2016.

[26] T. Abera, N. Asokan, L. Davi, F. Koushanfar, A. Paverd, A.-R. Sadeghi, and G. Tsudik, "Invited: Things, trouble, trust: On building trust in IoT systems," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2016, pp. 1–6.

[27] M. Ambrosin, M. Conti, R. Lazzeretti, M. M. Rabbani, and S. Ranise, "Collective remote attestation at the Internet of Things scale: State-of-the-art and future challenges," *IEEE Communications Surveys Tutorials*, vol. 22, no. 4, pp. 2447–2461, 2020.

[28] S. F. J. J. Ankergård, E. Dushku, and N. Dragoni, "State-of-the-art software-based remote attestation: Opportunities and open issues for Internet of Things," *Sensors*, vol. 21, no. 5, 2021.

[29] L. Jäger, R. Petri, and A. Fuchs, "Rolling DICE: Lightweight remote attestation for COTS IoT hardware," in *Proceedings of the 12th International Conference on Availability, Reliability and Security*, ser. ARES '17. New York, NY, USA: Association for Computing Machinery, 2017.

[30] Z. Tao, A. Rastogi, N. Gupta, K. Vaswani, and A. V. Thakur, "DICE*: A Formally Verified Implementation of DICE Measured Boot," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 1091–1107.

[31] M. Xu, M. Huber, Z. Sun, P. England, M. Peinado, S. Lee, A. Marochko, D. Mattoon, R. Spiger, and S. Thom, "Dominance as a new trusted computing primitive for the Internet of Things," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 1415–1430.

[32] M. Huber, S. Hristozov, S. Ott, V. Sarafov, and M. Peinado, "The Lazarus effect: Healing compromised devices in the Internet of Small Things," in *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, ser. ASIA CCS '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 6–19.

[33] P. Birnstill, C. Haas, D. Hassler, and J. Beyerer, "Introducing Remote Attestation and Hardware-based Cryptography to OPC UA," in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2017, pp. 1–8.

[34] D. Bienhaus, L. Jäger, R. Rieke, and C. Krauß, "Gateway for industrial cyber-physical systems with hardware-based trust anchors," in *Intelligent Distributed Computing XIII*, I. Kotenko, C. Badica, V. Desnitsky, D. El Baz, and M. Ivanovic, Eds. Springer International Publishing, 2020, pp. 521–528.