Proceedings of ACM IoT Privacy, Trust & Security - IoTPTS 2016
Xi'an, China, May 2016

Cloud-Based Commissioning of Constrained Devices using Permissioned Blockchains

Thomas Hardjono
MIT Connection Science & Engineering
Cambridge, MA 02139, USA
hardjono@mit.edu

Ned Smith
Intel Corporation
2111 NE 25th Ave
Hillsboro, OR 97124
ned.smith@intel.com

ABSTRACT

In this paper we describe a privacy-preserving method for commissioning an IoT device into a cloud ecosystem. The commissioning consists of the device proving its manufacturing provenance in an anonymous fashion without reliance on a trusted third party, and for the device to be anonymously registered through the use of a blockchain system. We introduce the *ChainAnchor* architecture that provides device commissioning in a privacy-preserving fashion. The goal of ChainAnchor is (i) to support anonymous device commissioning, (ii) to support device-owners being remunerated for selling their device sensor-data to service providers, and (iii) to incentivize device-owners and service providers to share sensor-data in a privacy-preserving manner.

CCS Concepts

•**Security and privacy** → **Public key encryption;**

Keywords

Internet of Things; Security; Privacy; Identity Management; Blockchains

1. INTRODUCTION

The "Internet of Things" (IoT) seeks to interconnect all kinds of devices with the promise of the betterment of society. Within the consumer devices space, there is concern on the part of individuals that device makers and service/cloud providers in their quest for new revenue sources will inadvertently compromise the privacy of individual consumers [15]. Data has more value if it is shared. In this paper we argue that a middle-ground is achievable through the combined use of privacy preserving technologies and anonymizing protocols, with the appropriate incentives model for the individual to share their data. New Internet infrastructures are needed to be developed that would allow service providers – who seek to collect data and obtain revenue from analytics based on the data – to share their revenue with individuals, whilst retaining the privacy of the individual.

In this paper we propose the *ChainAnchor* architecture which seeks to address the following challenges:

- *Anonymous device commissioning*: Support the owner of a constrained device in deploying the device (e.g. into a smart home environment). We define device *commissioning* as consisting of two general steps. First, proving the manufacturing provenance of the device in an anonymous fashion without reliance on a trusted third party. Secondly, anonymously register the device (as now being "activated" and "owned") through the use of a blockchain system.

- *Support device owners "selling" data*: Support individual owners of constrained devices to sell or license their sensor-data to service providers (e.g. data broker entity), in a privacy-preserving manner.

- *Incentivize individuals based on revenue sharing*: Provide a mechanism for service providers to incentivize individuals to share device-data in a privacy-preserving manner, and remunerate individuals based on a revenue-sharing model.

ChainAnchor builds on the *Enhanced Privacy ID* (EPID) scheme of [5] for zero-knowledge proofs, and makes use of the blockchain as a mechanism to anonymously register device commissioning and decommissioning. EPID provides the ability for devices to prove its provenance without relying on the manufacturer or on an external trusted third party. Furthermore, EPID provides a number of revocation capabilities should a device (or its keys) become suspect of being compromised.

EPID is an extension of the *Direct Anonymous Attestation* protocol (DAA) [3] for user privacy in the TPMv1.2 hardware [16]. The EPID protocol can be deployed without *Trusted Platform Module* (TPM) hardware, with the option to add and enable a tamper-resistant TPM at a later stage. This option may be attractive to service providers who may wish to deploy TPM-based infrastructure in a phased approach [11, 12]. When a TPM hardware is deployed, it can be used to provide protected storage for the various keys used in the ChainAnchor system.

The EPID scheme provides a number of advantages over the basic DAA protocol. EPID provides a more flexible key generation and signature creation options together with a number of revocation options. EPID signatures are not only *anonymous* and *untraceable* but also *unlinkable*. The un-

IoTPTS'16, May 30, 2016, Xi'an, China

DOI: http://dx.doi.org/10.1145/2899007.2899012

traceability feature is what distinguishes EPID from group-signature schemes.

EPID is not the only anonymous identity protocol available today. The work of Brickell et al. [3] introduced the first RSA-based DAA protocol in 2004. A related anonymity protocol called *Idemix* [7] employs the same RSA-based anonymous credential scheme as the DAA protocol. However, Idemix cannot be used with the TPMv1.2 hardware (or the new TPMv2.0 hardware). Another related protocol called *U-Prove* [13] can be integrated into the TPM2.0 hardware (see [8]). However, the U-Prove protocol has the drawback that it is not multi-show unlinkable [9], which means that a U-Prove token may only be used once in order to remain unlinkable.

In the next section we describe the ChainAnchor architecture and protocol steps. We do this without reference to any specific blockchain. This is followed in Section 3 by a discussion on blockchain technology in the context of IoT device commissioning.

The current paper seeks to be readable to a broad audience, and as such it does not cover in-depth the cryptography behind EPID and DAA. We assume the reader is familiar with public-key cryptography and with the basic operations of the blockchain in the Bitcoin system. In order to assist the more curious reader, we provide a brief summary of the EPID scheme in the Appendix and provide pointers to the relevant equations in the Appendix. The current paper focuses on an RSA-based EPID scheme based on the Camenisch-Lysyanskaya signature scheme [6] and the DAA scheme of Brickell, Camenisch and Chen [3]. Readers are directed to the authoritative papers of [3] and [5] for an in-depth discussion. An EPID scheme using bilinear pairings can be found in [4]. It is based on the Boneh, Boyen and Schacham group signature scheme [1] and the Boneh-Schacham group signature scheme [2].

2. CHAINANCHOR DESIGN

As mentioned previously ChainAnchor follows two integrated steps in performing device commissioning, namely (i) proving the manufacturing provenance of the device in an anonymous fashion without reliance on a trusted third party; (ii) anonymously register the device through the use of a blockchain system. Figure 1 summarizes the ChainAnchor flows.

We refer to the manufacturer as the *Provenance Issuer* or source of provenance. We use the notion of a *Provenance Group* created by a device-manufacturer, to distinguish devices produced by different manufacturers. The same notion of provenance groups can also be used within one manufacturer to distinguish between certain device categories or attributes (e.g. types, batches, etc).

Our proposed ChainAnchor architecture makes use of the zero-knowledge proof protocol of EPID to allow a device to prove to a *Provenance Verifier* entity (and to the device owner) that the device has the correct provenance as coming from a given manufacturer. At the completion of this step, two things occur. First, the device self-generates a public key pair (called the *transaction key pair*) which is to be used later on the blockchain system. Secondly, the device and the Provenance Verifier establishes a *pair-wise shared key* (PSK), which is subsequently used to create a secure channel between the device and the Provenance Verifier. This secure channel is used to deliver a copy of the transaction public-key to the Provenance Verifier. The Provenance Verifier "records" the successful commissioning by recording a transaction to the blockchain addressed to the device's transaction public-key.

The EPID-signatures done by the device as part of the zero-knowledge proof keeps the devices anonymous and untraceable to the Provenance Verifier and to other external entities. On the blockchain, the device is recognizable only through its transaction public-key. The existence of a device's transaction public-key on the blockchain – achieved by the Provenance Verifier recording a transaction to that public-key on the blockchain – signals to data brokers and related service providers that the device is now fully operational under the consent of the owner.

2.1 Entities in the System

- *Device Manufacturer and Provenance Issuer* (DM-PI):
  The device manufacturer is the entity that created the device and seeks to allow external entities to obtain assurance as to the provenance of the devices that it produces. As such, it plays the role of the provenance-issuer.

- *Data Broker and Provenance Verifier* (DB-PV):
  The DB-PV is the provenance verifier entity that performs the anonymous provenance verification of a device. We assume multiple DB-PV entities exist.

- *Data Brokers*:
  We assume that there are multiple data brokers who are not provenance verifiers. In this case they rely on a DB-PV for proof of provenance of a device.

- *Device Owner*:
  The device-owner or "owner" is the person who legally owns the device.

- *On-Boarding Tool & Wallet*:
  The on-boarding tool (OBT) is a software tool that is used by the device-owner to manage the constrained device. The on-boarding tool provides a user-friendly interface to the owner to activate and to commission a device. The OBT is assumed to have "wallet" capability to allow it to interact with a blockchain. It also has access to some of the keys that the device self-generates.

- *Constrained Device*:
  This is the IoT device which is being commissioned by its owner. We assume the device has at the very least the capability to sign transactions. The device plays the role of the *Provenance Attester* when engaging with the DB-PV entity.

- *Blockchain*:
  The blockchain is assumed to be a private permissioned blockchain or the current public permissionless Blockchain (capital "B") used in Bitcoin [14].

2.2 Keys in the System

The ChainAnchor system uses a number of cryptographic keys, some of which are EPID-specific, notably the *Provenance Verification Public Key* and the *Device-Member Private Key*. For our purposes in ChainAchor, it is important to

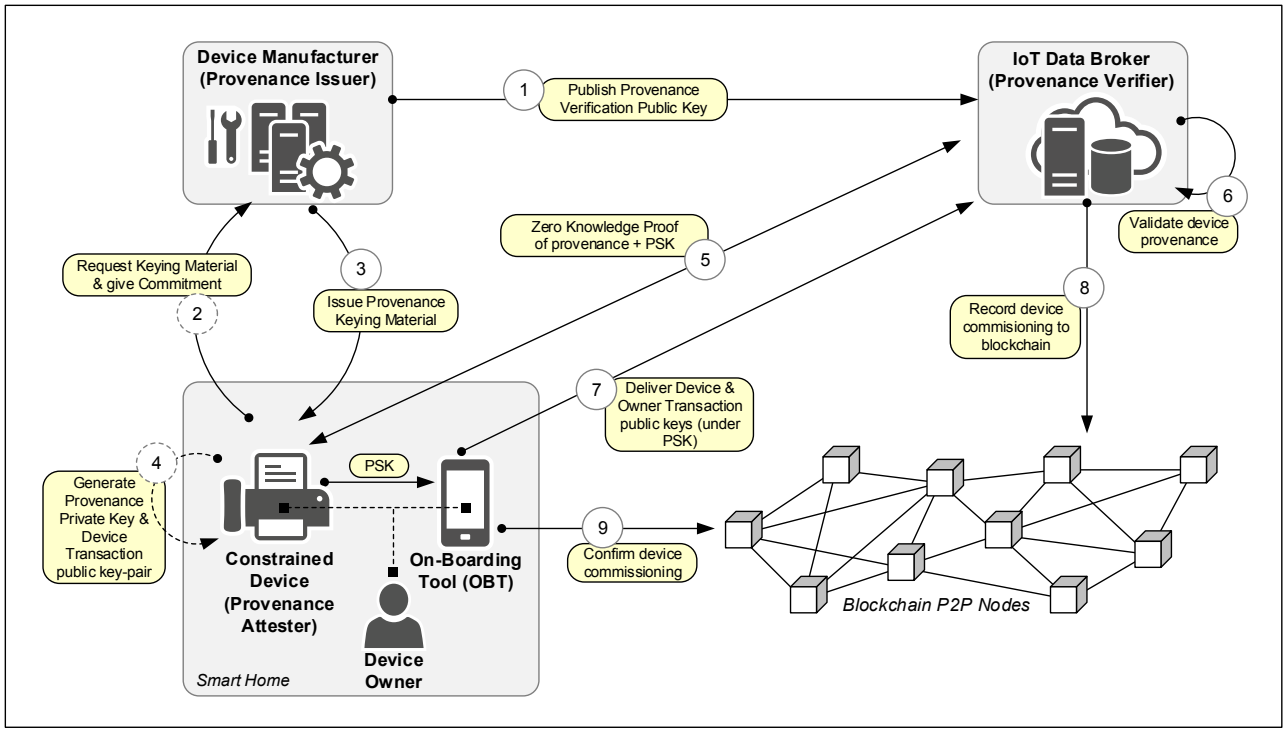

Figure 1: Overview of Device Activation and commissioning in ChainAnchor

note the many-to-1 relationship between device member private keys and the provenance verification public key. That is, for a given provenance group PRG there is one (1) provenance verification public key K_{PRG} held by the DB-PV entity. However, that single public key is used by the DB-PV to validate the manufacturing provenance of multiple (n) devices $D_1, \ldots, D_n$ with a corresponding device member private keys $K^{-1}_{\text{PRG-D}_1}, \ldots, K^{-1}_{\text{PRG-D}_n}$.

These keys in the system are summarized as follows:

- *Provenance Issuing Private Key*:
  This key is denoted as K^{-1}_{PIPK} and is generated by the Manufacturer (DM-PI) for each Provenance Group PRG that the DM-PI establishes. This key is unique for each group and is used by the DM-PI in enrolling or adding new devices to the group during manufacturing. This key is shown in Eq. 2 in the Appendix.

- *Provenance Verification Public Key*:
  This key is denoted as K_{PRG} and is generated by the DM-PI and is delivered over a secure channel to the Provenance Verifier entity (DB-PV). This key is unique for each Provenance Group denoted as PRG. The key allows the DB-PV later to validate the membership of a device (in the corresponding Provenance Group). This key is shown in Eq. 1 in the Appendix.

- *Device-Member Private Key*:
  This key is unique for each device for the Provenance Group to which the device belongs (provisioned). The key is generated by the device and kept secret inside the device. For a given device D_i provisioned for membership in a Provenance Group PRG, the device member private key is denoted as $K^{-1}_{\text{PRG-D}_i}$. This key is shown in Eq. 6 in the Appendix.

- *Device & DB-PV Pairwise Shared Key*:
  As part of proving provenance, the device and the DB-PV will establish a pairwise shared key (PSK). The PSK is a symmetric key.

- *Device Transaction Public-Key Pair*:
  This is the public-key pair used by the device to transact on the blockchain. The key pair is generated by the device or by the TPM hardware if the device has a TPM. We denote this as $(K_{DevTrans}, K^{-1}_{DevTrans})$, with the public key being $K_{DevTrans}$. The OBT has access to this key-pair.

- *Owner Transaction Public-Key Pair*:
  This is the public-key pair used by the Owner to transact on the blockchain. The key pair is generated by the OBT, or by the TPM hardware if the OBT has a TPM on its platform. We denote this key-pair as $(K_{OwnTrans}, K^{-1}_{OwnTrans})$, with the public key being $K_{OwnTrans}$.

- *Provenance Verifier Transaction Public-Key Pair*:
  This is the public-key pair used by the DB-PV to transact on the blockchain. We denote this key-pair as $(K_{PVTrans}, K^{-1}_{PVTrans})$.

- *Manufacturer and Data Broker Certificates*:
  We assume the DM-PI and the DB-PV have published their X.509 certificates in the usual manner.

In the following, we describe the phases of a device's introduction into the environment. We group the steps into phases for ease of understanding and for delineating the responsibilities of entities in the ecosystem.

2.3 Phase I: Device Manufacturing

In this phase, the Manufacturer prepares the device for the market and makes present in the device a number of cryptographic keying material and parameters.

[Step-0] *DM-PI Establishes Provenance Group*:

This step is not shown in Figure 1. Here, the Manufacturer (DM-PI) as the provenance issuer establishes a new Provenance Group for the set of devices the Manufacturer wishes to ship. The Manufacturer creates a group by selecting a group identifier and additional attributes that are specific to the group, and then generates a number of parameters that are unique to the group:

- *Provenance Verification Public Key*: K_{PRG}
  The Manufacturer creates this key for the provenance group of devices. (See Eq. 1 in Appendix A).
- *Provenance Issuing Private Key*: K^{-1}_{PIPK}
  The Manufacturer creates this key in order to issue unique keys to devices. It keeps this key as secret. (See Eq. 2 in Appendix A).
- *Device installed with* K_{PRG}: The Manufacturer makes present in each device (in the group) a copy of the provenance verification public key K_{PRG}. Later during activation each device D_i will generate a random number that is then used to compute $K^{-1}_{\text{PRG-D}_\text{i}}$.

[Step-1] *DM-PI Publishes Verification Key to DB-PV*:

In Step-1 of Figure 1, the manufacturer (DM-PI) provides a copy of the Provenance Verification Public Key (K_{PRG}) to all Provenance Verifiers in the group. Delivery of this key must be over a secure channel.

2.4 Phase II: Device Activation

In this phase, the device has been shipped and is in the hands of the new owner.

[Step-2] *Device Activation & Blinded Commitment*

When the Owner obtains a new device and wishes to add the device to the environment using the OBT, the device activation process initiates a request to the Manufacturer for additional parameters (see Step 2 of Figure 1).

- *Device generates commitment parameters*: The device uses some of the parameters in the Provenance Verification Public Key to create a *commitment* value that "blinds" the device's own secret keying material. (See Eq. 3 and Eq. 4 in Appendix A).
- *Device sends the blinded commitment to the Manufacturer*: The device sends the commitment parameters to the DM-PI, who in-turn must verify that these parameters are formed correctly.

This cryptographic *blinding* (in the commitment values) is done to retain the anonymity of the device in the field from the manufacturer. This is a crucial aspect for the manufacturer, because now the manufacturer can in truth claim that it is unable to track the IoT devices in the field.

[Step-3] *DM-PI returns Group-Member Keying parameters*

In Step-3 of Figure 1 the manufacturer generates a number of parameters for the Member Private Key and sends them to the device. (See Eq. 5 in the Appendix).

[Step-4] *Device Generates Device-Member Private Key & Transaction Key Pair*

In Step-4 of Figure 1 upon receiving the device-specific parameters, the device uses these parameters to generate its own specific *Device-Member Private Key*, denoted as $K^{-1}_{\text{PRG-D}_\text{i}}$. (See Eq. 6 in Appendix A). Additionally, the device also generates its blockchain transaction key-pair $(K_{DevTrans}, K^{-1}_{DevTrans})$.

It is here that the core value of the EPID scheme comes into play. More specifically, if two devices D_1 and D_2 independently present a message with a signature-of-knowledge (see Eq. 8) created using their device member private keys $K^{-1}_{\text{PRG-D}_1}$ and $K^{-1}_{\text{PRG-D}_2}$ respectively, then the Provenance Verifier can verify both using the one verification public key K_{PRG}. However, the Provenance Verifier will not be able to distinguish between devices D_1 and D_2.

2.5 Phase III: Device Commissioning

Device commissioning involves both the device and the onboarding tool (OBT) driven by the owner. First the device must prove its provenance to the DB-PV in a zero-knowledge fashion – namely proving that it knows some secret parameters belonging to the Provenance Group (from Step-0). A successful completion of this task is signified by the establishment a PSK between the device and the DB-PV.

Second, the OBT (representing the human owner) must use the PSK to securely deliver the public keys $K_{DevTrans}$ and $K_{OwnTrans}$ to the DB-PV. The fact that the OBT knows the PSK (which the OBT reads from the device) and is able to exercise the PSK (as proof-of-possession) tells the Provenance Verifier that the human user is participating and has given consent to the commissioning. These two tasks consist of a number of sub-steps shown in Figure 2.

[Step-5] *Proving Provenance*

The sub-steps of this zero knowledge protocol is run between the device and the Provenance Verifier (DB-PV), and is shown in Figure 2:

(a) The device sends a request to Provenance Verifier for device provenance verification.

(b) The Provenance Verifier responds with a challenge message m and a random nonce n_{pv}. (Although not discussed in-depth in this paper, the DB-PV also supplies a signature revocation list (sig-rl) to the device [5]).

(c) Upon receiving the challenge message m and the random nonce n_{pv} from the Provenance Verifier, the device computes an EPID "signature of knowledge" of the commitment parameter that the device previously supplied to the Provenance Issuer (DM-PI) in Step-2. The signature-of-knowledge is denoted as σ. (See Eq. 8 in Appendix A).

As input into the signature-of-knowledge computation, the devices uses:

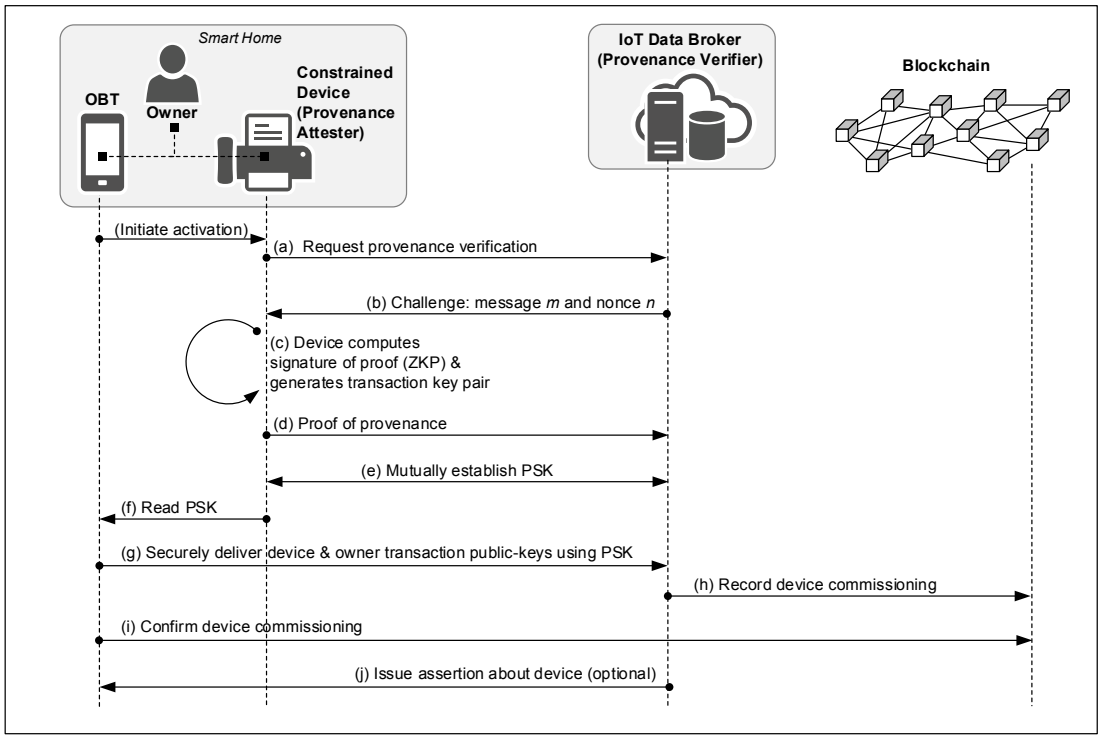

Figure 2: Anonymous Provenance Validation and Commissioning

- The Provenance Verification Public Key (K_{PRG}) – a copy of which was made present by the manufacturer inside the device in Step-0 during manufacturing. (See Eq. 1 in Appendix A).
- The device's own Device-Member Private Key $K^{-1}_{\text{PRG-D}_\text{i}}$ which the device generated in Step-4. (See Eq. 6 in Appendix A).
- The challenge m and the nonce n_{pv} obtained from the Provenance Verifier DB-PV.

(d) The device sends the computed signature-of-knowledge value σ to the Provenance Verifier.

[Step-6] *Provenance Verifier Validates Proof*

In Step-6 of Figure 1, the Provenance Verifier receives the signature-of-knowledge value σ from the device and validates the proof. If the validation is successful, the device and the Provenance Verifier establishes a shared PSK (symmetric key). See sub-step (e) in Figure 2.

Note that in computing the signature-of-knowledge value σ, a further improvement can be made where the device includes the *basename* value in the signature sent to the Provenance Verifier. This allows the owner to choose the Provenance Verifier that she or he trusts, several of which may exist at any one time. This approach is taken by the *DAA-SIGMA* key exchange protocol [17], which embeds DAA within the key agreement flows.

[Step-7] *OBT delivers transactions public keys to DB-PV*

In Step-7 of Figure 1, the owner uses the OBT to securely deliver (under the PSK) a copy of the device and owner's transaction public-keys to the Provenance Verifier. This is shown as sub-step (f) and (g) in Figure 2:

(f) The OBT reads the PSK from the device.

(g) The OBT sends $K_{DevTrans}$ and $K_{OwnTrans}$ to the Provenance Verifier under a secure channel established using the PSK.

[Step-8] *DB-PV Records Commissioning on Blockchain*

In Step-8 of Figure 1, the Provenance Verifier records a transaction to the blockchain addressed to the device's public-key $K_{DevTrans}$. This is shown as sub-step (h) Figure 2.

[Step-9] *OBT Confirms Device Commissioning*

In Step-9 of Figure 1, the OBT (on behalf of the device) confirms the device commissioning by recording a transaction using $K_{DevTrans}$ to the blockchain addressed to the public-key $K_{PVTrans}$ of the Provenance Verifier. This is shown as sub-step (i) Figure 2.

However, prior to this the owner (using the OBT) must first check the blockchain to ensure that the DB-PV has completed the previous Step-8. The owner must look for a transaction originating from the DB-PV. This transaction must be addressed to the device's public-key $K_{DevTrans}$ and must be signed by the DB-PV. If the owner is not able to locate the transaction within any blocks in the blockchain after a reasonable amount of time the owner must not proceed beyond this step. The absence of the correct transaction from the DB-PV means that the commissioning is incomplete (i.e. failed).

Note that the OBT (driven by the owner) is the entity that performs Step-7 and Step-9 instead of the device. There are two reasons for this. Firstly, the human owner must participate in the device commissioning as it signifies to the broader ecosystem that the owner has activated the device and has commissioned the device. Secondly, we cannot assume that a constrained device has wallet capability and can therefore interact directly with the blockchain.

2.6 Phase IV: Decommissioning Device

There are number of cases relating to a current owner decommissioning a device. Two of the most common reasons are (i) device deactivation for permanent decommissioning and (ii) device acquiring a new owner (i.e. sold). In both cases two things need to occur:

- *Current owner decommissioning*: The current owner must indicate that she or he is decommissioning the device out of the cloud ecosystem. This can be done by the owner recording a transaction to the blockchain addressed to the public-key $K_{PVTrans}$ of the Provenance Verifier. This would be the third transaction on the blockchain involving the addresses $K_{PVTrans}$ and $K_{DevTrans}$, and is taken to mean device decommissioning.

- *Provenance Verifier erases relevant keys*: The DB-PV must erase all the relevant keying material and keys pertaining to the owner and the device.

3. BLOCKCHAIN FOR THE IOT INDUSTRY

We believe an industry-wide permissioned blockchain for IoT devices may be more cost effective and more scalable compared to operating a PKI-based service in the cloud. Such an industry-wide blockchain should allow end-users and consumers to verify the state of their IoT devices in a privacy-preserving manner. The blockchain could be owned and operated by a consortium of industry device manufacturers, service providers (e.g. Identity Provider), data brokers and consumer groups. Such a blockchain could assist not only manufacturers in legally proving the change of ownership of devices, but also ownership transfers in the secondary market (e.g. Alice selling her device to Bob).

3.1 Standardizing Transaction Payloads

An industry-wide blockchain for IoT devices would need to provide a standardized payload definition for the transactions in the blockchain. These payload definitions could cover the various tasks related to the lifecycle of an IoT devices. For example, device ownership transfers, device commissioning, firmware updates, device associations, and others.

Figure 3 (a) and (b) attempts to illustrate an example of a blockchain for IoT devices. A linked set of signed assertions shown in (a) capture a log regarding device activation and commissioned status, while the blockchain (b) records the event in an immutable and order-preserving manner.

3.2 Semi-Permissioned Blockchains

We define a *semi-permissioned* blockchain as one that satisfies both of the following: (i) only authorized entities can "write" transactions to it (gated by the *consensus nodes* or "miners") but anyone can read and validate transactions recorded on the blockchain, and (ii) the consensus nodes enforce authorization by processing only transactions whose public-keys are approved to be on the blockchain. A semi-permissioned blockchain can be implemented in several ways, such as using a separate peer-to-peer network, a separate virtualized blockchain (e.g. Ethereum [10]) and other methods.

In the context of ChainAnchor, the enforcement of write-access to a semi-permissioned blockchain could be performed together by the DB-PV and ChainAnchor consensus nodes:

- *Provenance Database*: The DB-PV must maintain a simple *Provenance Database* or list of anonymous transaction public-keys belonging to devices (and owners) who completed the ChainAnchor commissioning.

- *Independent Validation of Transactions*: The DB-PV must independently validate all blocks that have been successfully processed or mined.

- *Source of Reward*: The DB-PV – alone or in conjunction with the DM-PI – rewards the consensus nodes for enforcing the write-access control to the blockchain.

4. REMUNERATION TO DEVICE OWNERS

In ChainAnchor the Provenance Verifier function is merged with the *Data Broker* function for purposes of simplicity. In order to support plain data brokers (who are not Provenance Verifiers) the DB-PV could issue signed assertions (e.g. in SAML2.0) regarding the device and owner transaction public-keys. Alternatively, the data brokers can also lookup on the blockchain.

Here we understand the data broker as having the particular business interest of obtaining accurate sensor-data from only "genuine" IoT devices with the consent of the owner.

ChainAnchor provides a way for IoT devices and device-owners to remain anonymous when interacting with data brokers or other data collection entities. A data broker can identify a legitimate (but anonymous) device through its transaction public-key ($K_{DevTrans}$) being confirmed by the DB-PV on the blockchain or by being listed in the Provenance Database. Furthermore, ChainAnchor provides a way for data brokers to remunerate the anonymous device-owner (i.e. pay for data) through the use of his or her transaction public-key ($K_{OwnTrans}$).

Thus, for example, the DB-PV could simply pay in bitcoins by sending BTC currency to the address or public-key of the owner. This remuneration could be based on the volume of data a device sends to the DB-PV, or based on some other criteria. The precise economic model for this method of remuneration is outside the scope of the current work.

5. CONCLUSIONS

In this paper we have introduced the *ChainAnchor* architecture for a cloud-based commissioning of an IoT device into the cloud ecosystem. ChainAnchor provides device commissioning in a privacy-preserving fashion and provides assurance to service providers that the device is a genuine product issued by the manufacturer. The goal of ChainAnchor has been (i) to support anonymous device commissioning, (ii) to support device-owners being remunerated for selling their device sensor-data to service providers, and (iii) to incentivize device-owners and service providers to share sensor-data in a privacy-preserving manner.

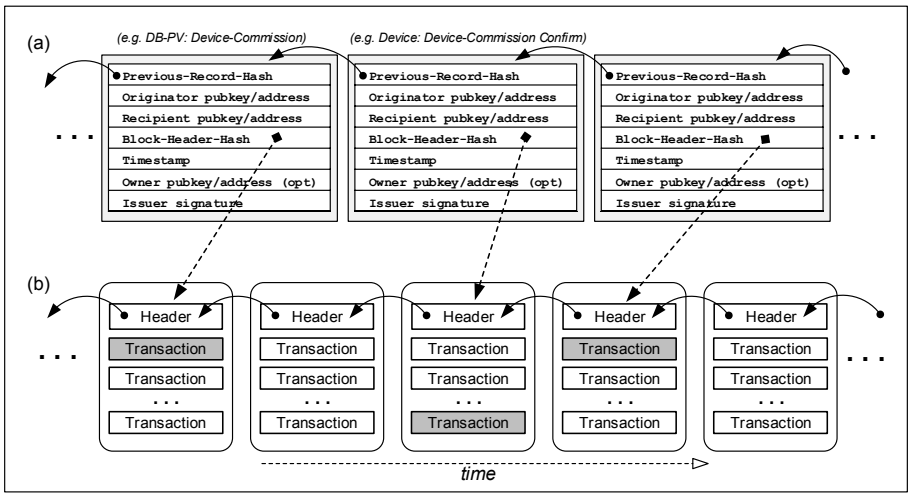

Figure 3: (a) Device commissioning assertions and (b) evidence on the blockchain

We have also discussed the need for an industry-wide permissioned blockchain for IoT devices that maintains the privacy of device-owner, and which may be more cost effective and more scalable for IoT devices compared to PKI-based services.

6. ACKNOWLEDGMENTS

We thank Prof Sandy Pentland (MIT Connection Science & MIT Media Lab) for his support for the current work.

7. REFERENCES

[1] D. Boneh, X. Boyen, and H. Shacham. Short Group Signatures. In *In Advances in Cryptology - Proceedings CRYPTO 2004*, pages 41–55. Springer, 2004.

[2] D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 168–177. ACM, 2004.

[3] E. Brickell, J. Camenisch, and L. Chen. Direct Anonymous Attestation. In *Proceedings of the 11th ACM CCS2004*, pages 132–145. ACM, 2004.

[4] E. Brickell and J. Li. Enhanced Privacy ID from Bilinear Pairing for Hardware Authentication and Attestation. In *Proceedings of the IEEE International Conference on Social Computing (SocialCom 2010)*, pages 768–775. IEEE, 2010.

[5] E. Brickell and J. Li. Enhanced Privacy ID: a Direct Anonymous Attestation Scheme with Enhanced Revocation Capabilities. *IEEE Trans on Dependable and Secure Computing*, 9(3):345–360, 2012.

[6] J. Camenisch and A. Lysyanskaya. A Signature Scheme with Efficient Protocols. In *Security in Communication Networks (SCN2002) (LNCS 2576)*, pages 268–289. Springer, 2002.

[7] J. Camenisch and E. Van Herreweghen. Design and implementation of the Idemix anonymous credential system. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 21–30. ACM, 2002.

[8] L. Chen and J. Li. Flexible and Scalable Digital Signatures in TPM 2.0. In *Proceedings of the 2013 ACM CCS2013*, pages 37–48. ACM, 2013.

[9] L. Chen and R. Urian. DAA-A: Direct Anonymous Attestation with Attributes. In *Proceedings of TRUST 2015 (LNCS 9229)*, pages 228–245. Springer, 2013.

[10] Ethereum. A Next-Generation Smart Contract and Decentralized Application Platform, Nov 2015. https://github.com/ethereum/wiki.

[11] T. Hardjono. Infrastructure for Trusted Computing. In *Proceedings of ACSAC Workshop on Trusted Computing*, December 2004.

[12] T. Hardjono and N. Smith (Eds). TCG Infrastructure Reference Architecture for Interoperability (Part 1) – Specification Version 1.0 Rev 1.0, June 2005. http://www.trustedcomputinggroup.org/ resources.

[13] Microsoft. U-Prove Cryptographic Specification v1.1 (Rev 3). Technical report, Microsoft Corporation, 2014.

[14] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System.

[15] Open Interconnect Consortium. OIC Security Specification Version 1.0.0. OIC Published Specification, OIC/OCF, December 2015.

[16] Trusted Computing Group. TPM Main – Specification Version 1.2. TCG Published Specification, TCG, October 2003.

[17] J. Walker and J. Li. Key Exchange with Anonymous Authentication using DAA-SIGMA Protocol. In *Trusted Systems (INTRUST2010) (LNCS 6802)*, pages 108–127. Springer, 2010.

APPENDIX

A. SUMMARY OF EPID

In the following we summarize the RSA-based EPID scheme as defined in [5].

A.1 Issuer Setup

In order to create a group membership verification instance, the Issuer must choose a *Group Public Key* and compute a corresponding *Group-Issuing Private Key*.

For the Group-Issuing Private Key the Issuer chooses an RSA modulus $N = p_N q_N$ where $p_N = 2p\prime_N + 1$ and $q_N = 2q\prime_N + 1$ and where p_N, p_N, $p\prime_N$ and $q\prime_N$ are all prime.

The Group Public Key for the particular group instance will be:

$$(N, g\prime, g, h, R, S, Z, p, q, u) \tag{1}$$

The Group Issuing Private Key (corresponding to the Group Public Key) is denoted as:

$$(p\prime_N, q\prime_N) \tag{2}$$

which the Issuer keeps secret).

In order to communicate securely with a User, the Issuer is assumed to possess the usual long-term public key pair denoted as $(K_I, {K_I}^{-1})$, where K_I is publicly know in the ecosystem.

Any User who has a copy of the Group Public Key can verify this public key by checking the following:

- Verify the proof that $g, h \in \langle g\prime \rangle$ and $R, S, Z \in \langle h \rangle$.
- Check whether p and q are primes, and check that $q \mid (p-1)$, $q \nmid \frac{(p-1)}{q}$ and $u^q \equiv 1 \pmod{p}$
- Check whether all group public key parameters have the required length.

A.2 Join Protocol: User and Issuer

In the join protocol, a given User seeks to send to the Issuer the pair (K, U) which are computed as follows.

- The User chooses a secret f and seeks to convey to the Issuer a *commitment* to f in the form of the value U.
- The value U is computed as

$$U = R^f S^{v\prime} \tag{3}$$

  where $v\prime$ is chosen randomly by the User for the purpose of *blinding* the chosen f.
- Next the User computes

$$K = {B_I}^f \pmod{p} \tag{4}$$

  where B_I is derived from the *basename* of the Issuer (denoted as bsn_I).

The goal here is for the User to send (K, U) to the Issuer and to convince the Issuer that the values K and U are formed correctly.

In the above Equation 4, a User chooses a base value B and then uses it to compute K. The purpose of the (B, K) pair is for a revocation check. We refer to B the *base* and K as the *pseudonym*. To sign an EPID-signature, the User needs to both prove that it has a valid membership credential and also prove that it had constructed the (B, K) pair correctly, all in zero-knowledge. In EPID and DAA, there are two (2) options to compute the base B:

- *Random base*: Here B is chosen randomly each time by the User. A different base used every time the EPID-signature is performed. Under the decisional Diffie-Hellman assumption, no Verifier entity will be able to link two EPID-signatures using the (B, K) pairs in the signatures.
- *Named base*: Here B is derived from the Verifier's basename. That is, a deterministic function of the name of the verifier is used as a base. For example, B could be a hash of the Verifier's basename. In this named-base option, the value K becomes a "pseudonym" of the User with regard to the Verifier's basename. The User will always use the same K in the EPID-signature to the Verifier.

A.3 Issuer generates User's Membership Private Key

In response, the Issuer performs the following steps:

- The Issuer chooses a random integer $v\prime\prime$ and a random prime e.
- The Issuer computes A such that

$$A^e U S^{v\prime\prime} \equiv Z \pmod{p}$$

- The Issuer sends the User the values

$$(A, e, v\prime\prime) \tag{5}$$

Note that the CL-signature [6] on the value f is $(A, e, v := v\prime + v\prime\prime)$. As such, the User then sets his/her Membership Private Key as:

$$(A, e, f, v) \tag{6}$$

where $v := v\prime + v\prime\prime$. Recall that f is the secret chosen by the User at the start of the Join protocol.

A.4 User proving valid membership

When a User seeks to prove that he or she is a group member, the User interacts with the Verifier entity. This is performed using the Camenisch-Lysyanskaya (CL) signature [6] on some value f.

This can be done using a zero-knowledge proof of knowledge of the values f, A, e, and v such that

$$A^e R^f S^v \equiv Z \pmod{N} \tag{7}$$

The User also needs to perform the following:

- The User computes $K = B^f \pmod{p}$ where B is a random base (chosen by the User).
- The User reveals B and K to the Verifier.
- The User proves to the Verifier that the value $\log_B K$ is the same as in his/her private key (see Equation 4).

In proving membership to the Verifier, the User as the prover needs to send the Verifier the value

$$\sigma = (\sigma_1, \sigma_2, \sigma_3) \tag{8}$$

where each of the values are as follows:

- σ_1: The value σ_1 is a "signature of knowledge" regarding the User's commitment to the User's private key and that K was computed using the User's secret value f.
- σ_2: The value σ_2 is a "signature of knowledge" that the User's private key has not been revoked by the Verifier (i.e. not present in the signature revocation list sig-RL).
- σ_3: The value σ_3 is a "signature of knowledge" that the User's private key has not been revoked by the Issuer (i.e. not present in the issuer revocation list Issuer-RL).