Securing User Identity and Transactions Symbiotically: IoT Meets Blockchain

David W. Kravitz
David.Kravitz@darkmatter.ae

Jason Cooper
Jason.Cooper@darkmatter.ae

***Abstract*—Swarms of embedded devices provide new challenges for privacy and security. We propose Permissioned Blockchains as an effective way to secure and manage these systems of systems. A long view of blockchain technology yields several requirements absent in extant blockchain implementations. Our approach to Permissioned Blockchains meets the fundamental requirements for longevity, agility, and incremental adoption. Distributed Identity Management is an inherent feature of our Permissioned Blockchain and provides for resilient user and device identity and attribute management.**

***Keywords*-IoT; cryptography; security; privacy; permissioned blockchain; blockchain**

I. Introduction

A traditional Unix environment conducts logging and permissions enforcement from a multi-user system perspective. With IoT, we now see a network of devices, where each device has different roles, capabilities, and permissions. The network has become the system and the devices the users. Effective management of both user and device identities and attributes can be realized with the Distributed Identity Management that our Permissioned Blockchain provides.

Blockchain is ultimately a distributed, immutable log of events. When harnessed properly, it greatly facilitates reconciliation of event history among multiple entities. Blockchain enables IoT devices to perform transactions, and to be tracked relative to time and location. The major missing link of the well-known Bitcoin blockchain is the element of "permissioning," i.e., a privacy-preserving, traffic-analysis- resistant methodology that leverages external trust relationships so as to establish an auditable identity- and attributes- management authorization framework. IoT devices suitably provisioned with identity management trust anchors can securely and efficiently transact over permissioned blockchains.

Blockchain technology is attempting to revolutionize an industry in existence for centuries. The integrity of a transaction ledger must remain intact for a decade or more. When contemplating contractual agreements, integrity must be maintained for many decades. In other words, the transactions will outlive the cryptographic primitives and devices that they were originally secured with. End users will need to vouch for transactions long after the original device has been removed from service. Asymmetric keys may be compromised years after they've committed a transaction.

Our design of the Permissioned Blockchain takes these long-term issues into account and boasts fundamental features to handle them gracefully. Flexible, established protocols provide a pathway from existing workflows and trust models to a robust and enduring system of record.

Within the Permissioned Blockchain ecosystem, identity and attribute management is built in for both users and devices. Identities, and their associated attributes, transcend any single device or group of devices.

II. Transaction Longevity

A. Asymmetric Key Rotation

Asymmetric cryptography (private/public key pairs) offers many advantages, especially when agreeing on a shared secret over an insecure channel or addressing non-repudiation by distancing the ability to verify from the ability to sign. However, caution is in order when designing a system with these primitives.

Private key management is essential. In most designs, the private keys become the trust anchor of the system, and the proof of identity. This makes the private keys the prime target of attackers. The longer a private key is held, the higher is the probability of compromise. A multi-decade system design must account for keys in use today being compromised in the future. A strong system will withstand future revelation of private keys.

Therefore, our design prioritizes easy rotation of asymmetric keys. This degree of freedom gives the protocol the ability to adopt resilient primitives that do not exist today. Rotating signature keys causes an adversary to have to re-attack in order to be able to continue to sign, provided that the underlying signature scheme remains robust. Similarly, rotating key agreement keys causes an adversary to have to re-attack in order to be able to recover the plaintext of future transactions.

B. Device Group Membership

When discussing proper key management, there are two basic principles that have been developed over several decades. First, assuming a suitable source of entropy, keys should be created on the device they are being used for. Second, keys should never be moved from device to device.

If Alice creates a transaction today, with her Android device, she needs to be able to prove that it's hers later.

Alice has an implicit group, ”Alice's Devices,” to which all of her devices belong. Now, when she creates her transaction, we record that it was done by one of ”Alice's Devices.” As her keys are rotated, and devices enter and leave her group, she can still prove, with any of her other devices, that she created that transaction. As soon as one of her keys is rotated out, or a device is disposed of, we record the event on the blockchain.

C. Hash Function Agility

In the previous sections, we reduced the importance of any single private key by developing a robust key rotation policy, and the concept of group membership. If the asymmetric keys are no longer our cryptographic trust anchor, what is? The answer resides in the hash functions used to hash each transaction and to relate one transaction to another in the blockchain. We would like to be able to take advantage of the critical role that hash function use plays in formulating a blockchain.

We must protect the blockchain via hash function agility. This means that we can start off with one hash function, say SHA-2, for hashing transactions. Then, later on, we can compute parallel hashes of existing transactions. Once the parallel hashes, say SHA-3, are accepted by the community, the old hash algorithm can be deprecated.

A first- or second- pre-image or collision- attack may exist against SHA-2 one day, but the parallel hash mechanism gives us the ability to detect such attacks. Even if a signature scheme used for past transactions becomes vulnerable, signing a different message as an attempted replacement will result in a different hash value under the parallel hash regime. Similarly, even if a key agreement scheme used for past transactions becomes vulnerable, confidentiality of these past transactions can be maintained if the current signature scheme is robust and signing by an authorized requester is required. Ciphertext stored off-chain and referenced on the blockchain by its hash can only be requested by an authorized requestor in the current system. This also aids in keeping the blockchain within a reasonable size limit, and is consistent with hashing state into blocks of transactions, where state results from execution of transaction chaincode or acceptance of inputs for external execution.

D. Transaction Expiration

How quickly can we reach old transactions relevant to new ones? How will resource-constrained embedded devices interact with the blockchain?

The key to answering these questions is to have a clearly defined transaction expiration policy. Many government and regulatory bodies require records to be kept for seven years. Following that example, transactions that are a) older than seven years, and b) are only referred to by transactions older than seven years are eligible for expiration. Transactions that may be older than seven years, perhaps a mortgage, but are referred to by younger transactions executing the terms of the contract would not be eligible for deletion.

To do this, we borrow a concept from the distributed version control system, git. In git, each commit contains a reference, by hash, to the previous parent commit. This is very similar to how blockchains function. A commit can also refer to more than one parent, and conversely, more than one commit can refer to a given parent commit. These features enable us to 'tie' together related transactions.

For embedded systems, with little storage space, the parameters may be tweaked. These tiny systems may only need the past 24 hours' worth of transactions, and only those transactions directly pertaining to its job. Multiple references facilitate this use case; it's akin to only checking out a single branch of a git source code repository. If more blockchain history is needed, the device can request the older data from its peers.

III. Overview of our Identity and Attribute Management Approach

”Attributes” can include identities, identifiers, entitlements, etc. User or device Identity and (organizational/device manufacturer) Affiliation can be considered special cases of attributes in that these are embedded into relatively long-term Enrollment Certificates, while these as well as dynamic attributes are incorporated into Transaction Certificates. A user or device may possess multiple Enrollment Certificates concurrently. Enrollment Certificates are used only 'behind the scenes' to prove provenance of attributes, while Transaction Certificates are used within blockchain transactions. These Transaction Certificates replace the raw/uncertified public keys used in Bitcoin addresses.

Our protocols are designed to gain the benefits of inheriting trust relationships that are established off-chain as well as leveraging trust relationships that develop through the history of blockchain transactions. Reciprocally, trust that is built up via the blockchain can be utilized for off-chain services.

Using standardized X.509 PKI and SAML/OAuth/OpenID federated identity management, the system provides trusted assertions of Identities, and other attributes such as Affiliation, user or device qualifications/authorized functionalities, resource entitlements, reputation metrics, etc. Unlike traditional systems, each of these assertions is securely translated into unique proofs-of-possession that are embedded into batches of issued Transaction Certificates. This embedding uses efficient symmetric cryptography in order to hide the information from unauthorized access, while enabling the user or device that owns a Transaction Certificate to privately prove possession of the attribute to intended parties on a selective release basis.

The audit system is further refined, so that appropriate measures can be applied to limit an authorized Auditor's access down to individual users or devices and/or to circumscribed transaction time periods. Such auditability is designed to be revocable. The issuance of Transaction Certificates is also auditable, so as to provably confine the Transaction Certificate-specific subject public keys to those that are derived via the hidden cryptographic expansion function from a long-term subject public key of a legitimate Enrollment Certificate. This inability to get an arbitrary subject public key included within a Transaction Certificate also has ramifications regarding enabling efficient but secure combining of private or public keys that improves transaction processing performance. This is accomplished without enduring the expense to clients or servers of processing individual proofs of possession of subject private keys in order to safely produce batches of Transaction Certificates.

Endorsements and other data relevant to the attributes and reputation (relative to such attributes) of users or devices that are exchanged within transactions can be processed/aggregated and fed back into the system so that future Transaction Certificates reflect such data within the embedded proofs-of-possession of attributes.

The core principles outlined above form the basis of handling Know Your Customer/Anti- Money-Laundering (KYC/AML) for financial services transactions and Know Your Machine (‘KYM’) for IoT devices [1], without sacrificing performance or throughput. The methodology was designed to also be extensible for use in M2M and other IoT transactions that involve severely- resource-constrained devices. Transaction Certificates and the cryptographic protocols are structured so as to avoid the need for long-term storage of per-certificate keys, and to minimize client computation, cross-server synchronization, and off-chain transaction setup communications. Our further exploration into the IoT domain entails extending the capabilities to handle dynamic and static groups of stationary and mobile devices so as to corroborate legitimate assertions (such as those pertaining to location or proximity) made by each, and to aid in timely detection of anomalous device and/or user behavior for the purpose of effective containment of rogue devices and/or users via revocation of Transaction Certificates or Enrollment Certificates. Applications include improved identity fraud management and tuning of authorizations to more securely permission high-value transactions, taking advantage of the immutably ordered and time-stamped ledger to aid in correlation of both on- and off- chain transaction activity.

Our solution is characterized by a number of advantageous properties, as outlined in the remainder of this section.

In response to a challenge by a system Attribute Certificate Authority (ACA), a user or device Client acquires fresh assertions from one or more Analytics Processors (APs) that encapsulate the current status of the user or device with regard to its cumulative ratings relative to one or more attributes (as derived by the AP from individual peer ratings on the blockchain it is privy to).

The attributes themselves are asserted by one or more standard Attribute Authorities (AA), where, as prerequisite, the Client provides such an AA with proofs-of-possession of the attribute(s) for which assertions are being requested. The Client also provides such AA with a challenge acquired from an ACA. Independently of the blockchain system, a device that is securely over-the-air-provisioned with upgraded software/firmware may receive a standard (key-less) attribute certificate [2] that expresses the device’s resultant functionality and references the device’s factory-provisioned certificate. Such attribute certificates are used in conjunction with the factory-provisioned device certificate to prove ownership of these attributes to an AA. Standard means based on established roots of trust are used for this, such as the device using a unique private key with which it had been provisioned (along with certificate on corresponding subject public key) by the device manufacturer in order to sign responses that incorporate AA-issued challenges.

The ACA verifies and collects such AA-issued assertions that reflect the ACA’s challenges into its database encrypted for access only by a Primary Transaction Certificate Authority (TCA). The Primary TCA (which may be split by using threshold- or multi- signatures), in turn, generates Template Transaction Certificates that incorporate sets of uniquely encrypted attributes, and the means for authorized Subordinate TCAs to generate batches of derivative Transaction Certificates (TCerts) that each rekey these encrypted attributes (by using TCert- and attribute- specific attribute encryption keys) and include a TCert-specific subject public key. The keying of the encrypted attributes stems from use of a hierarchical key derivation tree, where each Subordinate TCA is granted access to node key(s) of that tree in accordance with the domain of its jurisdiction. The construction of the TCerts also involves use of a Client-specific (key-derivation) key that is provided alongside the Template Transaction Certificate, encrypted based on the tree so that it is accessible by the Subordinate TCA. The per-TCert subject public keys are generated using an expansion function of the user/device long-term public key that is extracted from an Enrollment Certificate that associates that key with the user/device steady-state attributes, such as userID/deviceID and user Affiliation (or device manufacturer/device type). Batches of TCerts are delivered encrypted for use only by the owner Client (to assure preservation of unlinkability against traffic analysis), along with per-TCert keys (also encrypted) that enable regeneration by the Client of the TCert-specific attribute encryption keys (since the Client does not have access to node keys of the tree). The enrollment private key known only to the Client plays a pivotal role in cryptographically binding the specific Client to its TCerts. Only that Client can take delivery and derive the per-TCert private keys required to sign blockchain transactions that are each accompanied by a one-time-use TCert. Only the enrollment private key and the Client-specific key need be available long-term to the Client. The per-TCert key can be discarded by the Client once that TCert is used in a transaction. The hierarchical key derivation tree is used by TCAs, APs, and Auditors.

We enable transaction continuity via ”group dynamics”: A device can incorporate into a blockchain transaction the means (using standard key agreement) of privately targeting data/keys to itself and other devices in the group at the time (i.e., ”reflexive” case), as well as to a transaction Validator group and/or an appropriate Auditor (dependent upon policy) and/or intended recipient devices that are not in the group at the time. A non-targeted device that later joins the group may be able to gain access to such data/keys by proving its group membership (as an attribute) to a member of such Validator group. Whether or not such device is entitled to gain access to private data of previous transactions, it may be able to claim association to such transactions within a follow-on transaction by selectively releasing its current membership in the group (by proving ownership of that attribute in a purposely non-transferable way by privately releasing the associated (Transaction Certificate-specific) attribute encryption key within the follow-on transaction that is digitally signed using the Transaction Certificate-unique private key that corresponds to the subject public key of a freshly used Transaction Certificate). Devices can vote other devices into or out of groups to which they belong (similarly to the way ratings are handled). An example application is where a device user proves their current location relative to location of that user at times of previous correlated transactions – e.g., in order to establish patterns for their identity profile or to prove their inability to have feasibly been involved in certain other on- or off- chain location-dependent transactions (such as card-present credit/debit transactions). This can serve as evidence of an impostor using a device associated with that user via group membership.

IV. Cryptographic Constructs

A. Authorization Key Management

We now provide a description of major elements of the cryptographic constructs in some detail, beginning with an overview of authorization key management.

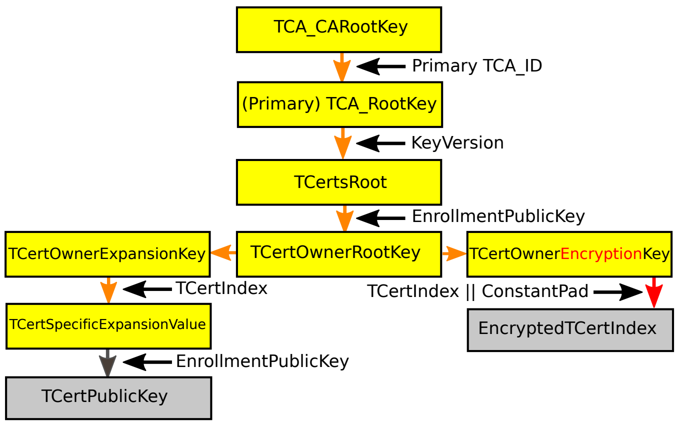

Fig. 1. Represents how authorization is achieved through the generation and use of TCerts

More specifically:

$$\text{TCA\_RootKey}_{384\text{bit}} = \text{HMAC}(\text{TCA\_CARootKey},\ \text{TCA\_ID}) \quad (1)$$

$$\text{TCertsRoot}_{384\text{bit}} = \text{HMAC}(\text{TCA\_RootKey},\ ''1'' \mid\mid \text{KeyVersion}) \quad (2)$$

$$\text{TCertOwnerRootKey}_{384\text{bit}} = \text{HMAC}(\text{TCertsRoot},\ \text{EnrollmentPublicKey}) \quad (3)$$

$$\text{TCertOwnerEncryptionKey}_{256\text{bit}} = \text{HMAC}(\text{TCertOwnerRootKey},\ ''1'') \ \text{(256bit truncation)} \quad (4)$$

$$\text{TCertOwnerExpansionKey}_{384\text{bit}} = \text{HMAC}(\text{TCertOwnerRootKey},\ ''2'') \quad (5)$$

TCertOwnerRootKey is the Client-specific (key-derivation) key.

$$\text{EncryptedTCertIndex} = \text{AES\_CBC\_Encrypt}(\text{TCertOwnerEncryptionKey},\ \text{TCertIndex} \mid\mid \text{ConstantPad}\ ) \quad (6)$$

is included within the TCert, where the ConstantPad provides a check of ownership by the User and/or by an Auditor that is assigned access to TCertOwnerEncryptionKey or the predecessor TCertOwnerRootKey on an individual User basis.

$$\text{TCertSpecificExpansionValue}_{384\text{bit}} = (\text{HMAC}(\text{TCertOwnerExpansionKey},\ \text{TCertIndex}) \mid\mid \text{HMAC}(\text{TCertOwnerExpansionKey},\ \text{TCertIndex} + 1)\ )\ (\text{mod}\ n) \quad (7)$$

where n per FIPS186-4 [5] relative to ECDSA, and TCertIndex is incremented by 2 for subsequent TCert in the batch. The double HMAC is introduced here to reduce exploitable bias that could otherwise jeopardize unlinkability. This issue is raised in [3], which specifies a PoC for the V2V collision-avoidance system that we started with when considering how to achieve unlinkability within a resource-constrained environment. The PoC is based on IEEE 1609.2-2016 Standard for Wireless Access in Vehicular Environments Security Services for Applications and Management Messages, and Crash Avoidance Metrics Partners (CAMP) LLC Vehicle Safety Communications 5 (VSC5).

B. Key expansion: TCert public key and private key derivation (with P-384)

Note that (unlike techniques such as Identity Mixer and U-Prove [4]) the public key expansion is initiated by a TCA rather than by the Client. Also, these key expansion functions are, by construction, independent of attributes (unlike the generation of public keys stemming from identitybased encryption and attribute-based encryption techniques). These properties give us the capabilities of (1) defining a common structure for the two kinds of unlinkable public-key bearing certificates that we require to efficiently manage read- and write- authorizations for use by resource-constrained devices, namely signature TCerts that are delivered directly to their owner, and key agreement TCerts that can be safely and usefully be delivered to requesting entities other than their owner, (2) enabling authorized Auditors to manage risk, gauge compliance, and fulfill regulatory requirements, and (3) enabling analytics processing by APs that results in refining attributes by adding qualifiers such as earned ratings, and that aggregates individual device votes in order to determine whether or not to endow devices with additional attributes such as device group membership.

$$\begin{aligned}&\text{TCertPublicKey}\\&\quad=\text{EnrollmentPublicKey}\\&\qquad+\text{TCertSpecificExpansionValue}*G \qquad (8)\end{aligned}$$

G per FIPS 186-4 [5] relative to ECDSA.

$$\begin{aligned}&\text{TCertPrivateKey}\\&\quad=(\text{EnrollmentPrivateKey}\\&\qquad+\text{TCertSpecificExpansionValue}) \pmod{n} \qquad (9)\end{aligned}$$

n per [5] relative to ECDSA. Note that the TCertOwner need not retain TCertPrivateKey, since it is recomputable from the TCert.

C. Attribute Management

Now we move on to being able to manage attributes within TCerts in a way that is compatible with (1) generation by Subordinate TCAs, with (2) access control relative to authorizing Auditors and APs, and with (3) selective and purposely non-transferable release of proofs of ownership by Clients/TCertOwners:

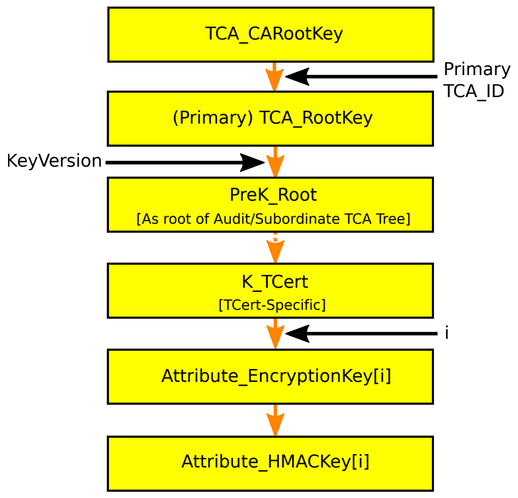

Fig. 2. Audit/Subordinate TCA key management

Let

$$\text{PreK\_Root}=\text{HMAC}(\text{TCA\_RootKey}, \text{"2"} \,||\, \text{KeyVersion}) \qquad (10)$$

Attribute_EncryptionKey[i] is derived from K_TCert, e.g., as HMAC(K_TCert, i).

Derive Attribute_HMACKey[i] deterministically from encryption key, e.g. as HMAC(Attribute_EncryptionKey[i], “1”).

As an illustrative example of the use of the hierarchical key derivation tree:

$$\begin{aligned}\text{PreK\_Root}&=\text{Primary\_TCAs\_Root\_of\_tree} \qquad (11)\\ \text{PreK\_ABC}&:\ \text{Available to Auditors/sub-TCAs}\\&\quad\text{with jurisdiction} \supseteq \qquad (12)\\&\quad\text{(Automobile, Banking, Construction)}\end{aligned}$$

$$\begin{aligned}\text{PreK\_B}&=\text{HMAC}(\text{PreK\_ABC}, \text{"Banking"}) \qquad (13)\\ \text{PreK\_BofZ}&=\text{HMAC}(\text{PreK\_B}, \text{"BofZ"}): \qquad (14)\\&\quad\text{Bank of Z}, \in \text{(Banking)}\end{aligned}$$

$$\begin{aligned}\text{PreK\_}[\text{BofZ, xyz}]&=\text{HMAC}(\text{PreK\_BofZ},\\&\quad\text{TCertID}=\text{"xyz"}) \qquad (15)\\ \text{K\_TCert}&=\text{PreK\_}[\text{BofZ, xyz}] \qquad (16)\end{aligned}$$

where K_TCert provided with TCert that has TCertID $=$ "xyz" to owner of that TCert, who is required to own ECert with Affiliation $=$ Bank of Z.

D. Basic Fields of a TCert

$$\text{TCertID (probabilistically unique per TCert)} \qquad (17)$$

$$\begin{aligned}&\text{Encrypted\_Attribute}[i]\\&\quad=\text{AES\_CBC\_Encrypt}(\text{Attribute\_EncryptionKey}[i],\\&\qquad\text{Attribute}[i]) \qquad (18)\end{aligned}$$

$$\begin{aligned}&\text{Authenticated\_Attribute}[i]\\&\quad=\text{HMAC}(\text{Attribute\_HMACKey}[i],\\&\qquad\text{Attribute}[i]) \qquad (19)\end{aligned}$$

$$\text{TCert\_Public\_Key (for signature verify)} \qquad (20)$$

$$\begin{aligned}&\text{Encrypted\_TCertIndex}\\&\quad=\text{AES\_CBC\_Encrypt}(\text{TCertOwnerEncryptionKey},\\&\qquad\text{TCertIndex} \,||\, \text{Constant\_Pad}) \qquad (21)\end{aligned}$$

$$\text{KeyVersion (for use by Auditors and TCertOwner)} \qquad (22)$$

E. Extending the Key Agreement Structure to Prevent Circumvention of Audit

Here a Validator group can enforce that a transaction creator has granted policy-stipulated Auditor access. This can be accomplished via verifiable key agreement: The basic construct is such that a validator applies a transaction private key provided securely to the validator by the transaction creator. The validator verifies that this transaction private key corresponds to a transaction public key that is included in the clear within the transaction creator- signed transaction. The validator also verifies that included within the signed transaction is ciphertext that, when decrypted, yields the data required to be transferred to an Auditor per the applicable policy.

F. *Four Kinds of TCerts that can be Requested from a Subordinate TCA by a User U / Device D*

- User U's / Device D's own signature TCerts
- User U's /Device D's own key agreement TCerts (for reflexive case or to distribute to others for future transactions that involve the user U / Device D)
- Auditor/AP key agreement TCerts
- Key agreement TCerts of other users/devices

These "other users/devices" may be identified within the request by userID. Alternatively, these are identified within the request by a dynamic attribute:

E.g., through an in-band transaction or off-chain communication, user U ascertains an account number of another user (and may or may not know that other user's userID).

V. Asset Transfer

We give an example below of an asset transfer use case applicable to financial services, since smart cities involve financial and governmental services as well as cyber-physical systems.

A user draws an asset from an account in order to transfer the asset to another account (that belongs to that user or another user).

Let Key_V denote an AES key derived from a one-pass elliptic curve Diffie-Hellman (ECDH) operation, where the static key pair (VPrivKey, VPubKey) is associated with a Validator group and the ephemeral key pair for the transaction (TxnPrivKey, TxnPubKey) is generated by the transaction creator. Similarly for Key_TC and Key_AT, for Transaction Creator and Asset Transferee.

VPubKey (or the identifier of the Validator group or identifier of a certificate that includes VPubKey) may be included within the Template Transaction Certificate by Primary TCA (and hence in the derivative TCerts generated by a Subordinate TCA) if Primary TCA knows which Validator group to designate based on the particular collection of Attributes from which the Template Transaction Certificate generation process draws. This field may be included within TCerts as an encrypted attribute, in order to not expose this information. This measure could be used to enforce/audit compliance by the user as Transaction Creator (or to detect non-compliance).

$$\begin{aligned}&\mathrm{BasicTransaction}\\&\quad = (\mathrm{Fresh\_TCert\_of\_Transaction\_Creator} \,||\\&\qquad \mathrm{Transaction\_Signature\_over\_Text}^{1} \,||\\&\qquad \mathrm{Text})\end{aligned} \tag{23}$$

for:

$$\begin{aligned}&\mathrm{Text}\\&\quad = (\mathrm{TxnPublicKey} \,||\, \mathrm{KeyAgreement\_TCert\_AT} \,||\\&\qquad \mathrm{KeyAgreement\_TCert\_TC} \,||\, \mathrm{V\_Ciphertext} \,||\\&\qquad \mathrm{AT\_Ciphertext} \,||\, \mathrm{TC\_Ciphertext}\,)\end{aligned} \tag{24}$$

[1] Generated using Signature_TCert_Private_Key

where the ciphertexts are:

$$\begin{aligned}&\mathrm{V\_Ciphertext} = \mathrm{AES\_Encrypt}(\mathrm{Key\_V},\\&\quad (\mathrm{Attribute\_Encryption\_Key(s)\_of\_Txn\_Creator} \,||\\&\quad\ \mathrm{Asset} \,||\, \mathrm{AccountID\_of\_Asset\_Transferee})\,)\end{aligned} \tag{25}$$

$$\begin{aligned}&\mathrm{AT\_Ciphertext} = \mathrm{AES\_Encrypt}(\mathrm{Key\_AT},\\&\quad (\mathrm{Asset} \,||\, \mathrm{AccountID\_of\_Asset\_Transferee})\,)\end{aligned} \tag{26}$$

$$\begin{aligned}&\mathrm{TC\_Ciphertext} = \mathrm{AES\_Encrypt}(\mathrm{Key\_TC},\\&\quad \mathrm{TxnPrivateKey})\end{aligned} \tag{27}$$

State is checked as indicating that accountID of Transaction Creator does indeed show Asset is available (such as available funds if Asset is a dollar amount). Upon successful validation and consensus, state is updated to show the Asset has been transferred from accountID of Transaction Creator to accountID of Asset Transferee.

VI. Conclusion

The use of identity (albeit in weak form such as username and password) and devices (e.g., one-time codes in SMS messages for multi-factor authentication) to secure transactions is a common practice. We have shown in some detail how this can be made substantially more secure through the use of permissioned blockchain technology. Furthermore, we have also demonstrated how to apply our techniques for the other direction, i.e., to improve the robustness of user identity against fraud by privately referencing the user's blockchain transactions. Another area we explored is securing swarms of devices that fulfill ad hoc or regularly scheduled tasks, via a privacy-preserving rating system for early detection of anomalous device behavior that warrants potential device revocation. The work in this paper presents and extends (with an emphasis on IoT) that which the first author briefed to the Linux Foundation Hyperledger Project community [6]; [7] and at Blockchain Protocol Analysis and Security Engineering 2017 [8].

References

[1] Pindar Wong, keynote abstract, http://wfiot2016.ieee-wf-iot.org/program/
[2] https://tools.ietf.org/html/rfc5755
[3] http://www.its.dot.gov/pilots/pdf/SCMS_POC_EE_Requirements.pdf
[4] http://neven.org/papers/ieeecreds.html
[5] http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
[6] https://github.com/hyperledger/hyperledger/blob/master/2016-06-23_Hyperledger Membership Services Presentation.pdf
[7] https://github.com/hyperledger/hyperledger/blob/master/2016-07-13_MembershipServicesInHyperledgerFabric_Part2.pdf
[8] "Permissioning Your Blockchain: How to Overlay Hyperledger Fabric with a Fully Workable System Tapestry": https://cyber.stanford.edu/blockchainconf