The work reported here was sponsored by a SBIR Phase I grant from the US Department of Homeland Security. It does not necessarily reflect the position or policy of the US Government.

New Techniques for Remote Identity Proofing

Presentation at CSUS

Francisco Corella
fcorella@pomcor.com
Karen Lewison
kplewison@pomcor.com

Identity proofing is different and harder than authentication

- In identity proofing there is no prior relationship between subject and verifier
- Authentication gold standard: present three verification factors
  - Something you have: device containing private key
  - Something you know: password
  - Something you are: one or more biometric features
- But in identity proofing, without prior relationship:
  - The subject cannot have previously registered a password with the verifier
  - The subject cannot have previously enrolled a biometric sample with the verifier

Existing solutions for identity proofing over the Web have serious problems

- Knowledge-based verification
  - No longer works: too much PII available to impostors
- Federated login (e.g. with Facebook, Twitter or Google, using OAuth or OpenID Connect or a proprietary protocol)
  - Identity provider observes all identifications
  - Availability and performance requirements hard to meet for authoritative identity sources such as, e.g. a DMV
- Public key certificates
  1. Only one verification factor
  2. Cumbersome validation of certificate chain
  3. No good solution for storing the certificate and its associated private key

We propose solutions to all 3 problems with traditional certificates

1. Rich credentials
   - Enable 3-factor verification (have, know, are) without prior relationship between subject and verifier
2. Method of implementing a PKI on a blockchain or distributed ledger
   - Simplifies the validation of a certificate chain
3. Two methods for storing a private key in the browser
   - Enabled by new web technologies

Rich credentials

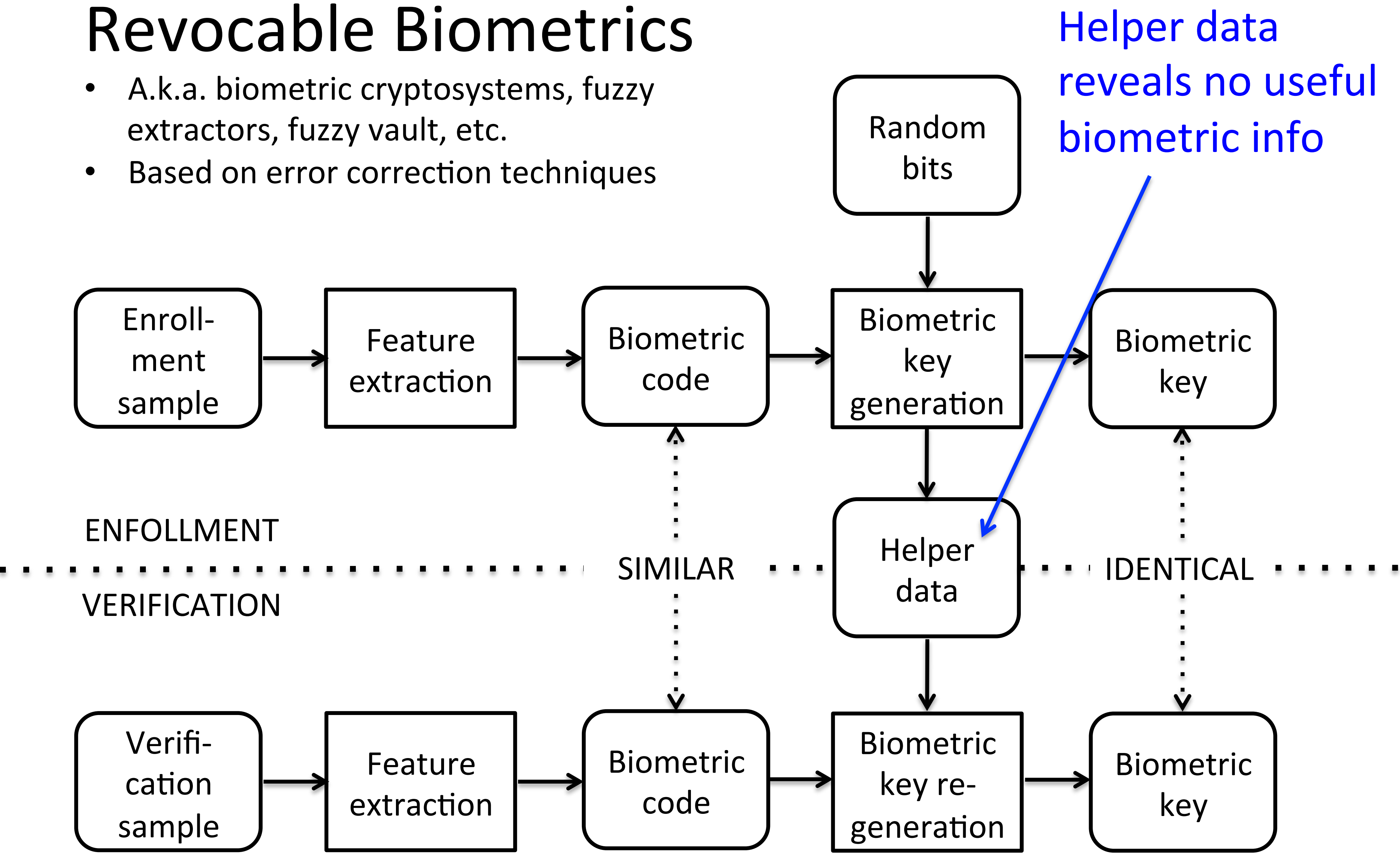
- A rich credential is a cryptographic credential that binds a public key to
  - Identifiers and/or attributes of the subject,
  - Verification data for a password, and/or
  - Verification data for one or more biometric samples
- The verifier can verify a password submitted by the subject against the credential signed by the issuer, without prior registration of the password
- The verifier can verify one or more biometric samples against the credential, without prior enrollment of the samples

Verification data in a rich credential

- For verifying a password:
  - Hash of a password (and two salts, as explained below)
- For verifying a biometric sample pertaining to a traditional biometric modality:
  - Enrollment sample or biometric code or template
- For verifying a biometric sample pertaining to a revocable biometric modality:
  - Helper data and hash of biometric key

Revocable Biometrics

- A.k.a. biometric cryptosystems, fuzzy extractors, fuzzy vault, etc.
- Based on error correction techniques



pomcor
pomcor.com

Biometric security

- Biometric security may be based on
  - Biometric secrecy
    - Preserved by revocable biometrics, but
    - Applicable to very few modalities (iris?)
    - Highly vulnerable to adversary acquiring biometric data
  - Presentation attack (a.k.a. spoofing) detection
    - More robust and broadly applicable, but
    - Difficult for remote presentation
    - Arms race between attack and detection techniques
- Biometrics should be used only in combination with other verification factors
  - As enabled by rich credentials

A possible method for presentation attack detection in face verification

- A DMV may want to issue a rich credential using a facial image embedded in the credential
- For presentation attack detection:
  - The facial image is matched against an audiovisual stream of the subject reading prompted text selected at random with high entropy
  - Speech recognition is used to verify that the text being read is the prompted one
  - Audiovisual synchrony is verified by using lip reading to correlate easily distinguishable visemes to corresponding phonemes

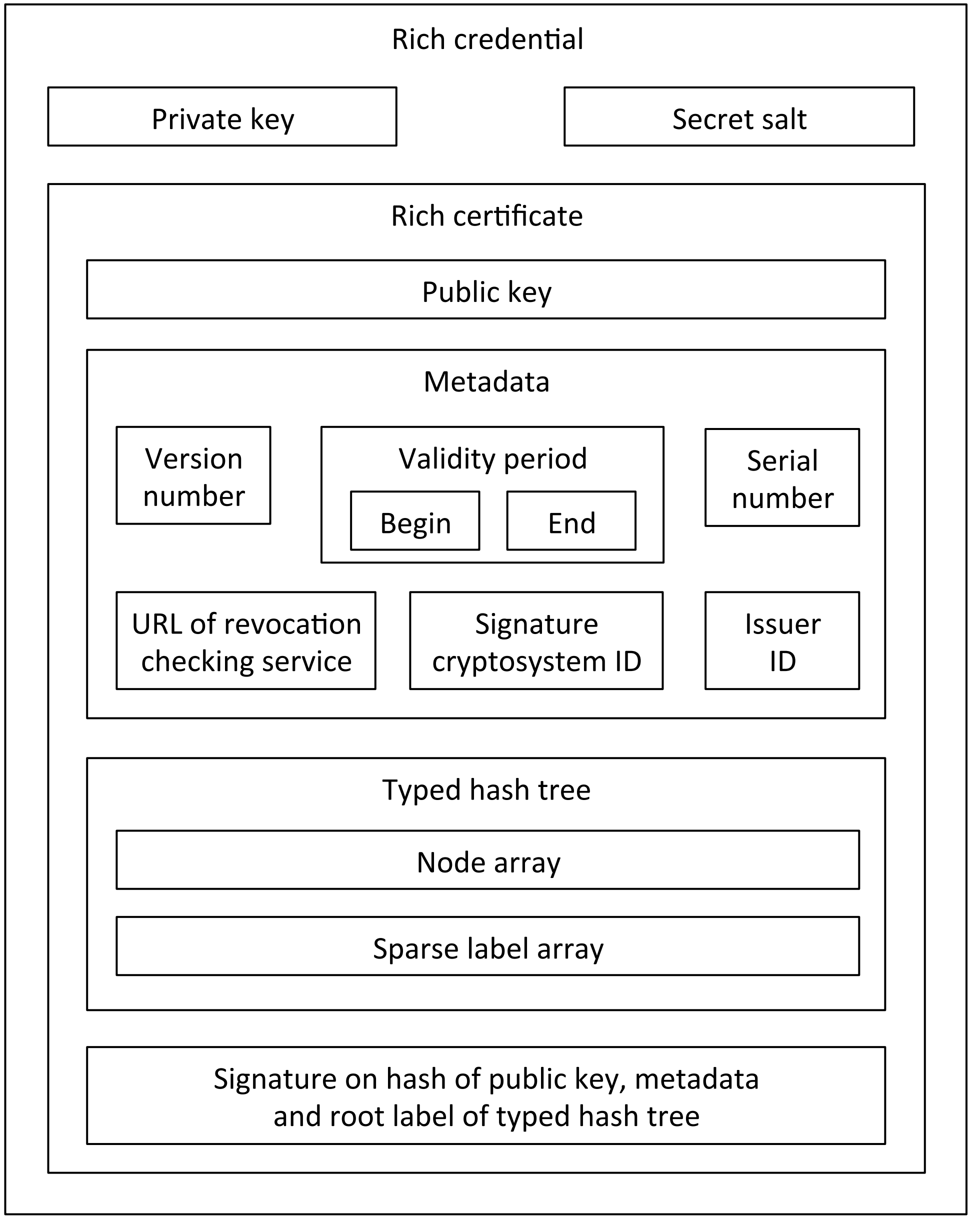
Privacy features of rich credentials

- A rich credential provides selective disclosure of attributes
  - Attributes to be disclosed are negotiated with the verifier, and the subject is asked for permission to disclose
- A rich credential also provides selective presentation of verification factors
  - Factors to presented are also negotiated with the verifier and the subject is asked for permission to present them

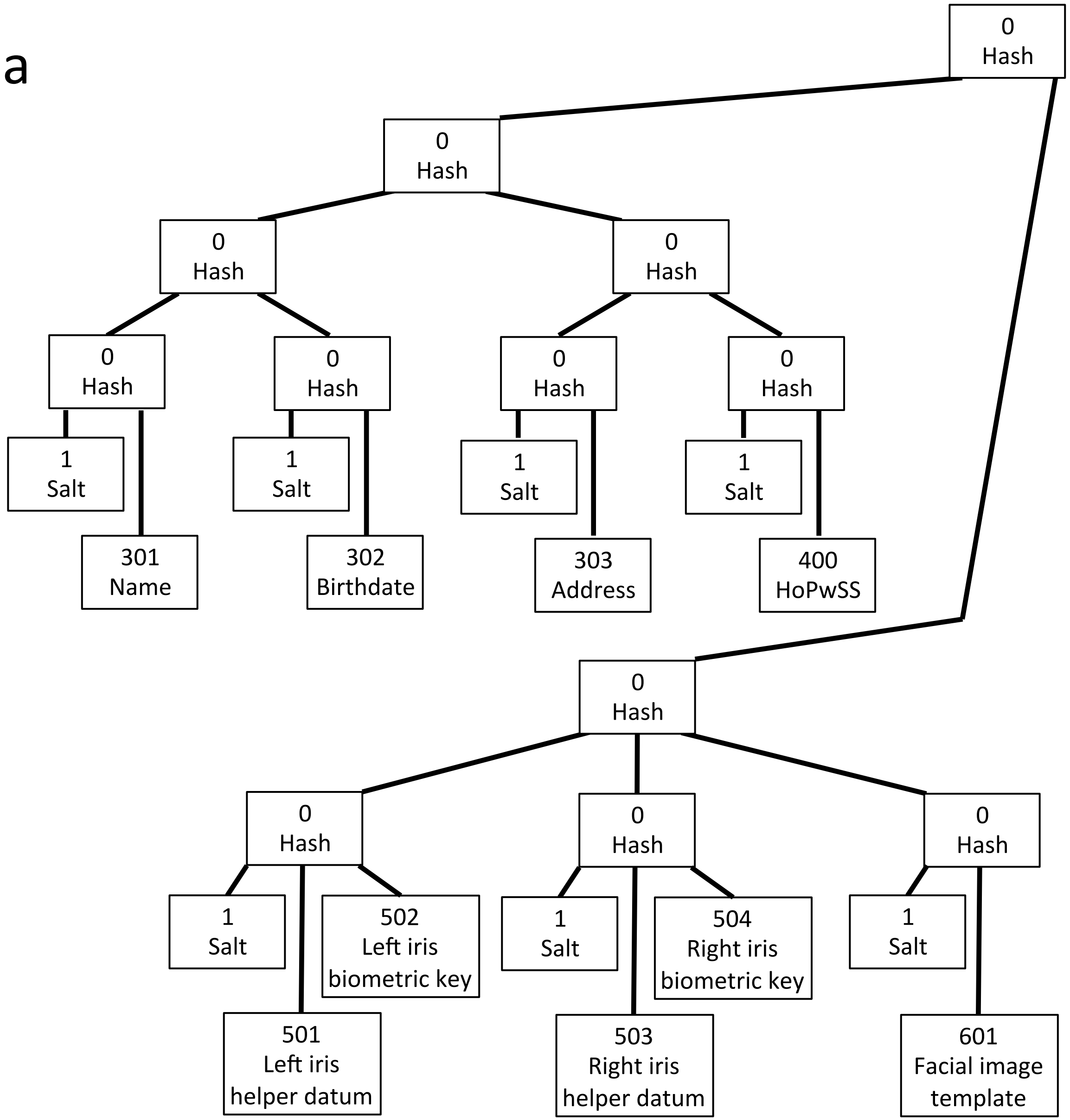
Typed hash trees

- Selective disclosure and selected presentation are achieved using a typed hash tree containing the attributes and the verification data
- The public key is bound to the root label of the tree rather than to the attributes and the verification data
- The root label serves as an omission-tolerant cryptographic checksum of attributes and verification data contained in the tree
  - Attributes and/or verification data can be removed by pruning subtrees, but cannot be added or modified without changing the root label
  - See formal result at the end of this presentation

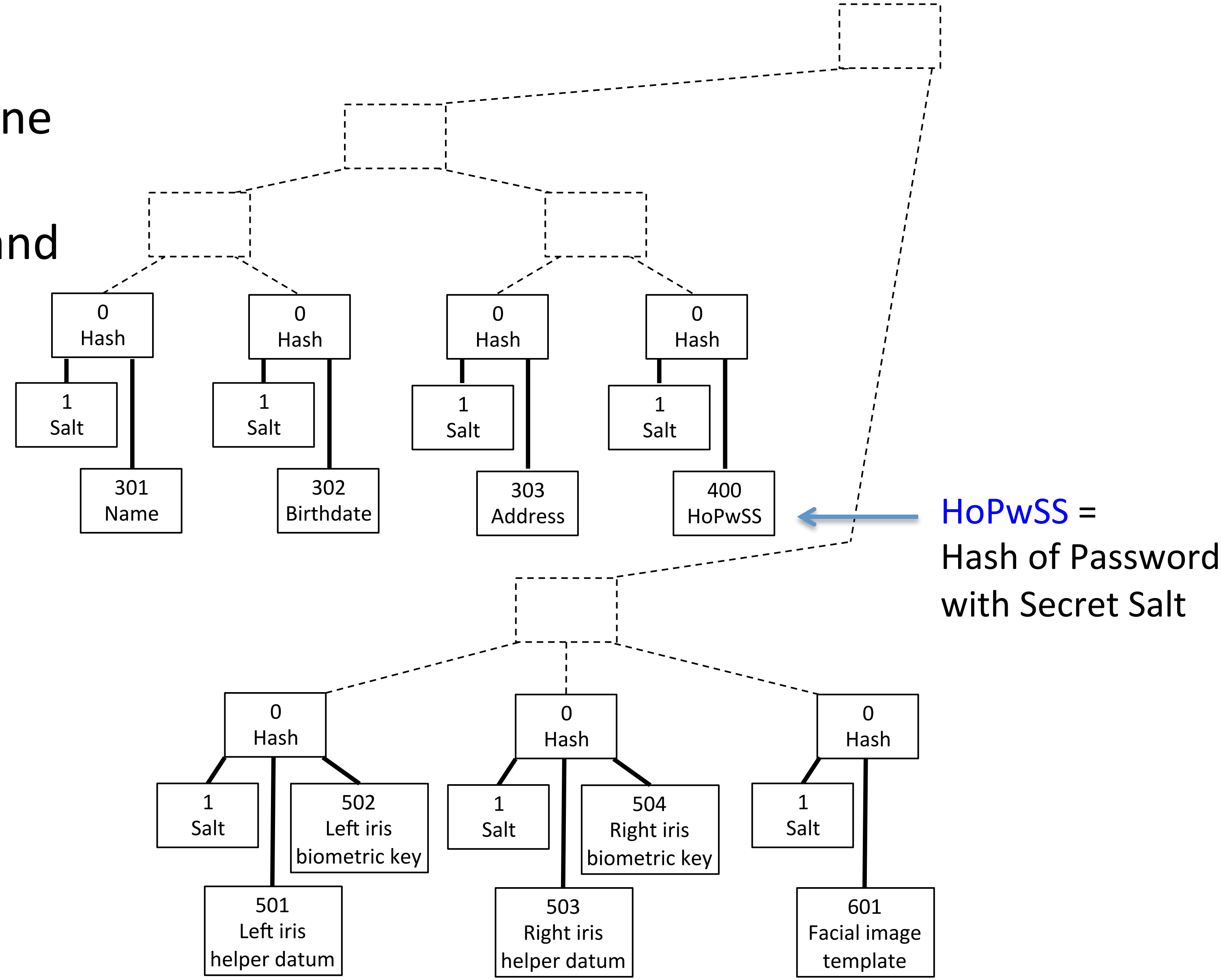
Components of a rich credential



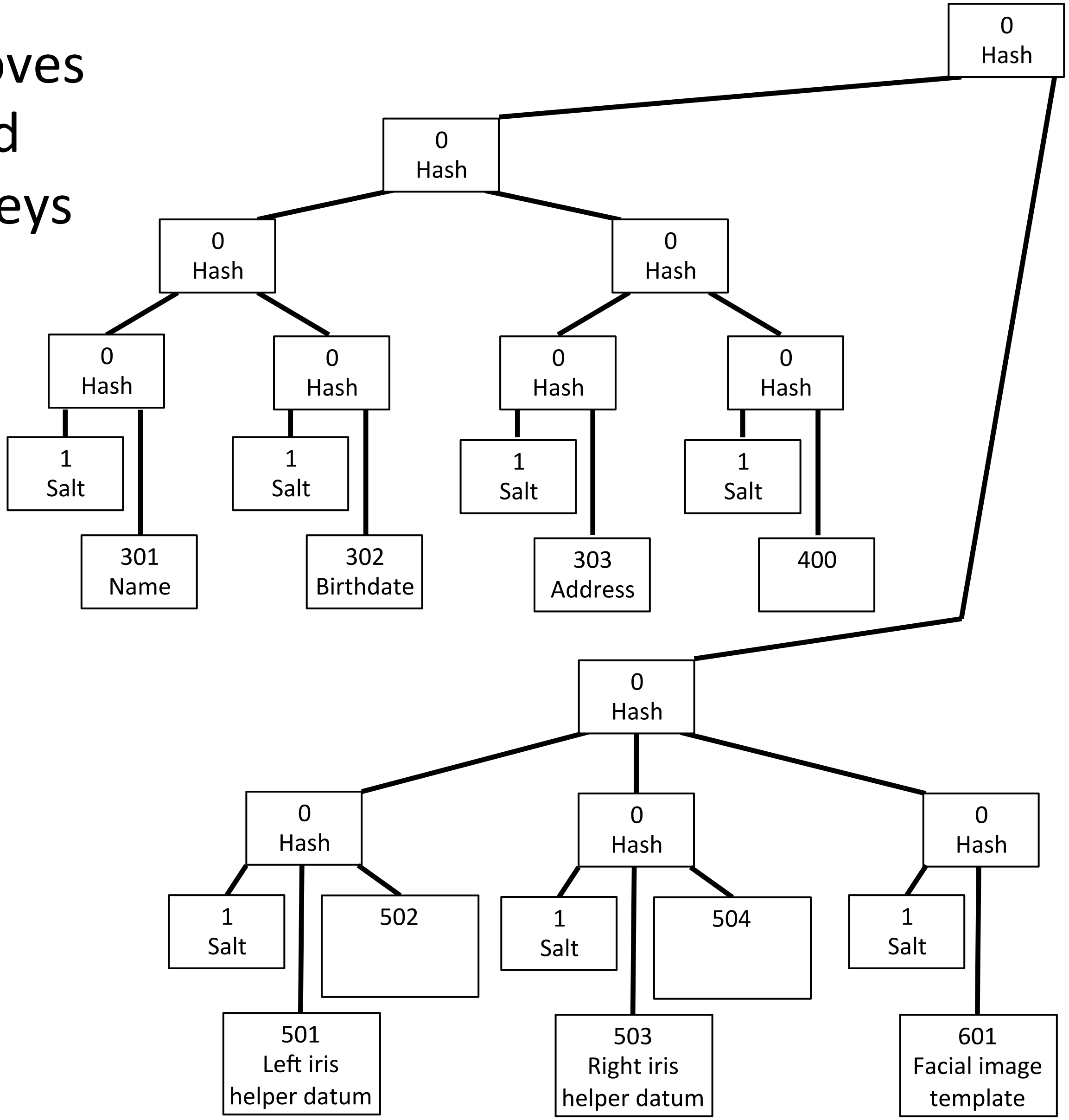
Example of a typed hash tree, in its issuance state



Peripheral subtrees, one for each attributre and each verification factor

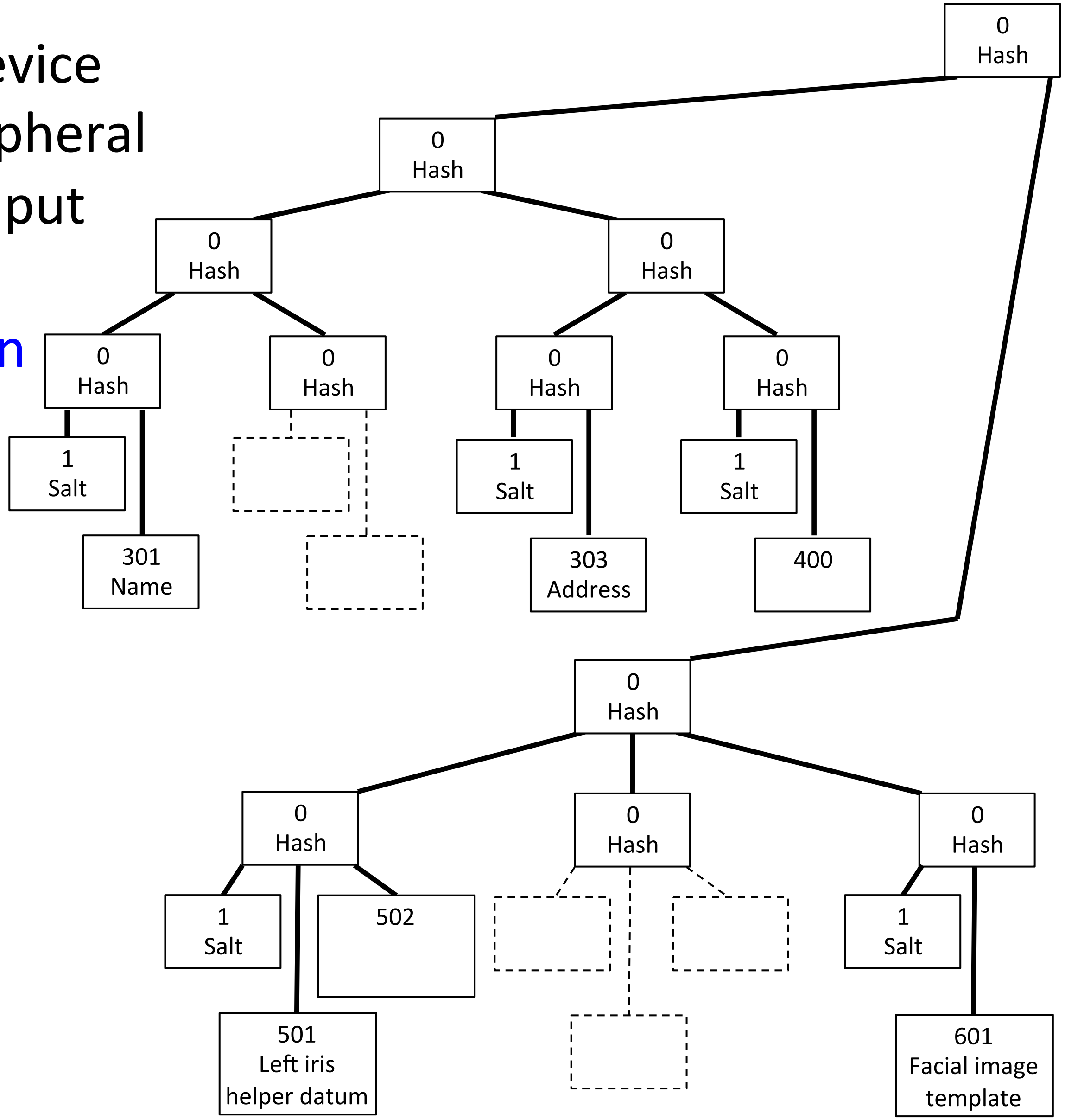


Issuer removes HoPwSS and biometric keys to put tree in its storage state



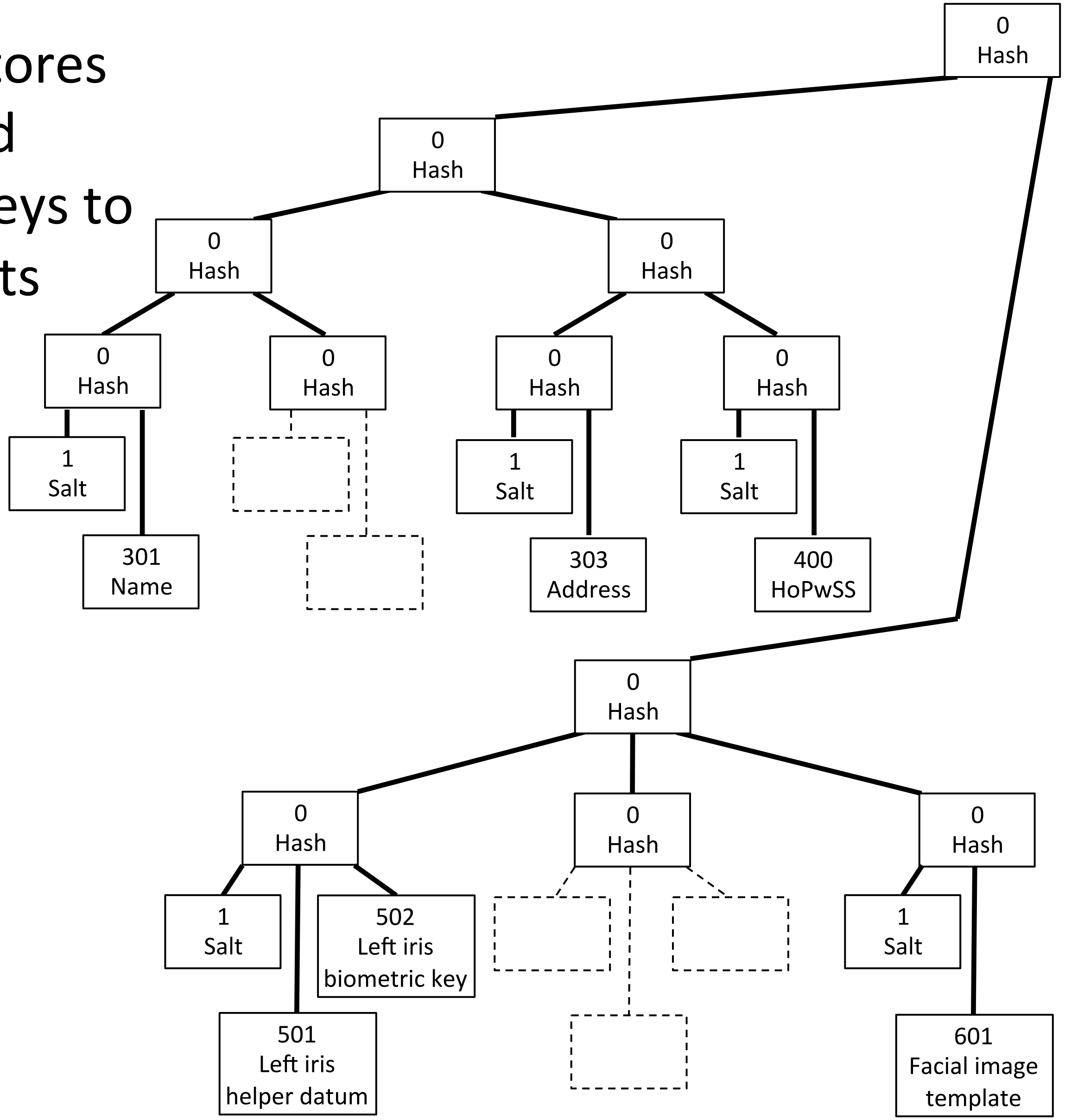
Subject's device prunes peripheral subtrees to put tree in its presentation state

Root label is omission-tolerant cryptogaphic checksum



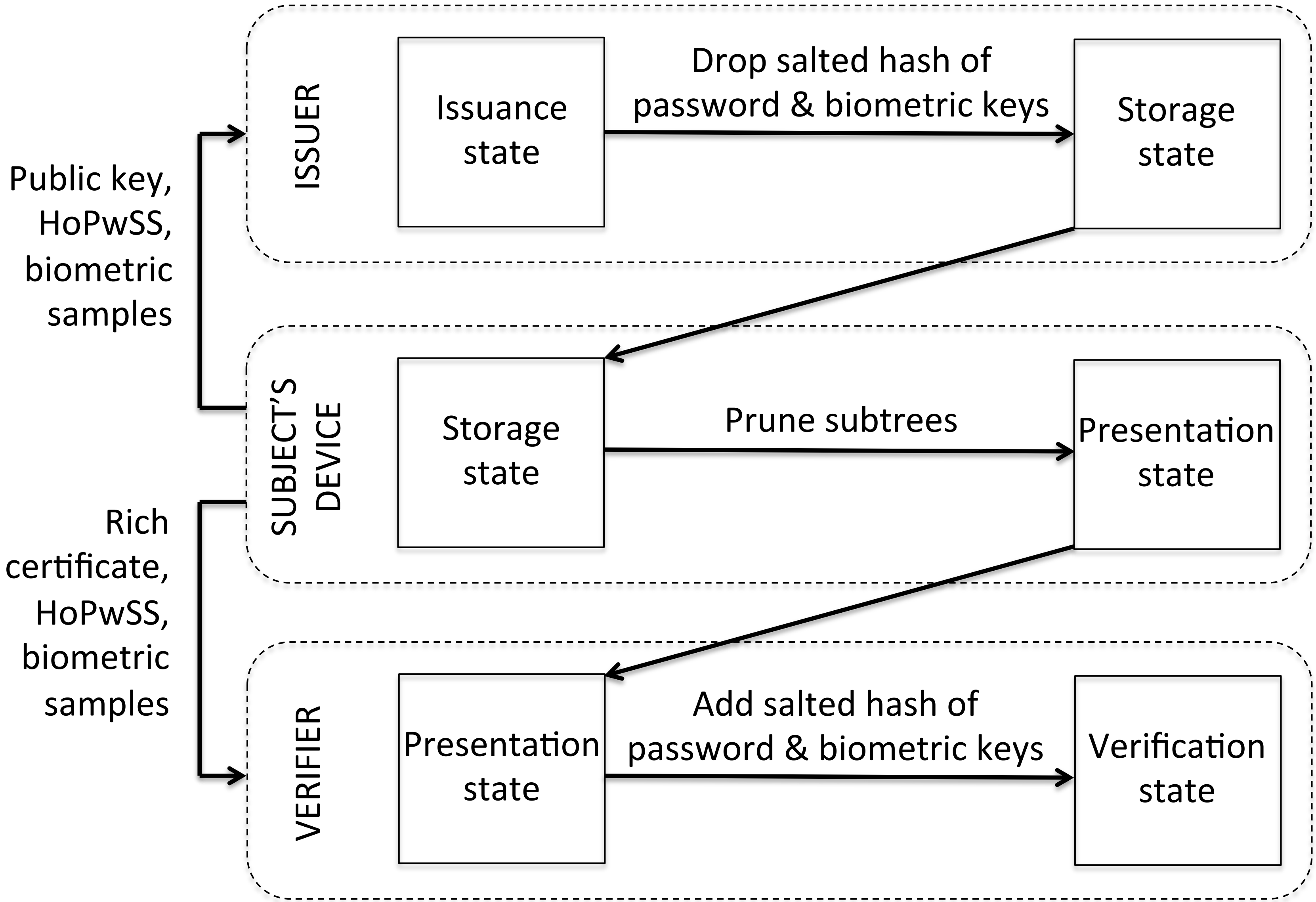
pomcor
pomcor.com

Verifier restores HoPwSS and biometric keys to put tree in its verification state



pomcor
pomcor.com

State transitions of a rich certificate



A rich credential can be backed by a traditional PKI

- The issuer signs the hash of the public key, metadata and root label of the typed hash tree, playing a role analogous to that of a CA
- If the issuer's public key is not well known, the rich credential can be backed by a chain of traditional X.509 CA certificates ending in a certificate signed by a root CA, whose public key is well known
- But validation of the rich credential and CA certificate chain, like validation of a traditional X.509 certificate and CA certificate chain, is cumbersome for the verifier

Using a blockchain or distributed ledger to simplify validation

- Distributed ledger:
  - List of digitally signed transactions
  - Replicated across nodes using a P2P network
  - Consensus achieved by a byzantine fault-tolerant algorithm
- Blockchain similar to distributed ledger, but:
  - Transactions grouped into blocks
  - Tentative consensus achieved by proof of work or proof of stake

On-ledger or on-chain storage

- Language for describing transactions includes instruction for storing data in named location
- As storage transaction propagates each node executes it in its own copy of the named location
- Ethereum has on-chain storage, the Bitcoin blockchain does not
- Most distributed ledgers implement general-purpose state machine replication, and as such provide on-ledger storage

Implementing a PKI using a blockchain or distributed ledger with on-chain/on-ledger storage

- A blockchain or distributed ledger with on-chain or on-ledger storage can be used to implement a PKI for rich credentials or ordinary X.509 end-user certificates
- To issue a rich credential, the issuer stores the hash of the public key, metadata and root label in an on-chain or on-ledger storage location that it controls, instead of signing it
- To revoke a rich credential, the issuer stores the hash in another such storage location
- To issue an X.509 end-user or CA certificate, the issuer stores the hash of the public key, metadata and attributes in a storage location that it controls
- To revoke an X.509 end-user or CA certificate, the issuer stores the hash in another storage location
- To validate a rich credential or X.509 end-user certificate and a CA certificate chain, the verifier only needs to look up a hash in its local copies of distributed storage locations

Advantages over a traditional PKI

- Advantages over signed credentials;
  - Shorter credential (no signature)
  - Signature verification step is eliminated
- Advantages over revocation with CRL
  - Verifier does not have to download and verify CRL updates
  - Issuer does not have to set up a CRL distribution service
  - Verifier does not have to rely on the availability of a CRL distribution service
  - Lag between revocation and issuance of CRL update is eliminated
- Advantages over OCSP
  - Issuer does not have to set up an OCSP service
  - Verifier does not have to rely on the availability of an OCSP service
  - Network latency is entirely eliminated

Two methods for storing a private key in the browser

- Two methods made possible by new web technologies
- Method 1
  - Relies on HTML5 localStorage and the Service Worker API
  - Relatively simple, but issuer has access to private key and can extract it and use it somewhere else
- Method 2
  - Relies on the IndexedDB API, the CryptoKeyPair object of the Web Cryptography API, and the Service Worker API
  - More complicated, but issuer does not have access to private key and cannot extract it and use it elsewhere

The root label of a typed hash tree is an omission-tolerant cryptographic checksum of its contents

- Definition of a typed hash tree:
  - Each node has a type and a label
  - An internal node has a distinguished type *d* (e.g. 0) and a label equal to the hash of a prelabel, which is an injective encoding of the sequence of types and labels of its children
  - Pruning a subtree causes its root to become a leaf node having the distinguished type *d*, which is called a dangling node
  - A proper leaf node is a leaf node with an undistinguished type, i.e. that is not a dangling node
  - An unpruned tree is a tree without dangling nodes
- A typed hash tree can be used to represent a collection (more precisely, a multiset) of key-value pairs
  - Each proper leaf node represents a key-value pair, where the key is the type and the value is the label

- Theorem:
  - Let X and Y be two typed hash trees with the same root label. If X is unpruned, then either Y is a pruned derivative of X, or a node of Y has the same label as a node of X but a different prelabel
- Proof summary:
  - Consider the computation C of the root label of Y from its leaf types and labels
  - Consider the earliest stage E of the computation C that coincides with a stage of the computation of the root label of a pruned derivative of X
  - Among the pruned derivatives of X that have E as a computation stage, consider a minimally pruned one X' and let C' be its computation
  - Show that stage E is reached in C and C' by hash steps that hash different prelabels to the same label

- Pruning a typed hash tree removes key value pairs
- Different prelabels that are hashed to the same label constitute a hash function collision

=> Corollary:

  - Let X and Y be two typed hash trees representing multisets M(X) and M(Y) of key-value pairs. If X is unpruned and Y has the same root label as X, then either M(Y) is a submultiset of M(X) or X and Y exhibit a hash function collision

Thank you for your attention!

For more information:
pomcor.com
pomcor.com/blog/
https://pomcor.com/techreports/RichCredentials.pdf
https://pomcor.com/techreports/BlockchainPKI.pdf

Francisco Corella
fcorella@pomcor.com
Karen Lewison
kplewison@pomcor.com

Any questions?