# Receipt Profiles

Amaury Chamayou

# SCITT Architecture uses COSE Receipts

- This document also defines message structures for signed statements and transparent statements, which embed COSE receipts [I-D.draft-ietf-cose-merkle-tree-proofs], i.e., signed verifiable data structure proofs).
- The signed statements and receipts are based respectively on the COSE_Sign1 specification in Section 4.2 of [STD96] and on COSE receipts [I-D.draft-ietf-cose-merkle-tree-proofs].
- COSE Receipts allow for profiles, Section 4.4 Profiles
  - New verifiable data structures can require the definition of a profile.
  - Requirements in Section 4.4.1 (MUST define encoding, production and verification)

# COSE Receipts defines an RFC9162_SHA256 profile

- Section 5
  - Verifiable Data Structure, Inclusion Proof, Consistency Proof
- Aimed at RFC9162 (a.k.a. CT v2)
- Good example, vendor neutral
- Alas, no known deployed implementation (*)

(*) as far as I know, please correct me if I’m wrong!

# But there are two deployed COSE Receipt implementations!

- Datatrails Forestrie
  - https://github.com/datatrails/go-datatrails-merklelog
- Microsoft SCITT CCF Ledger
  - https://github.com/microsoft/scitt-ccf-ledger/
  - https://azure.microsoft.com/en-us/blog/protecting-azure-infrastructure-from-silicon-to-systems/
  - More details to come soon at Ignite (Nov 18)
- Verification interop demo'ed in Madrid
  - https://github.com/achamayou/scitt-for-phi

# Merkle Mountain Range Profile

- https://github.com/robinbryce/draft-bryce-cose-receipts-mmr-profile/
- Efficient Asynchronous Accumulators for Distributed PKI, by Reyzin and Yakoubov
- Sub trees always balanced
- Insert and merge
- Set of roots

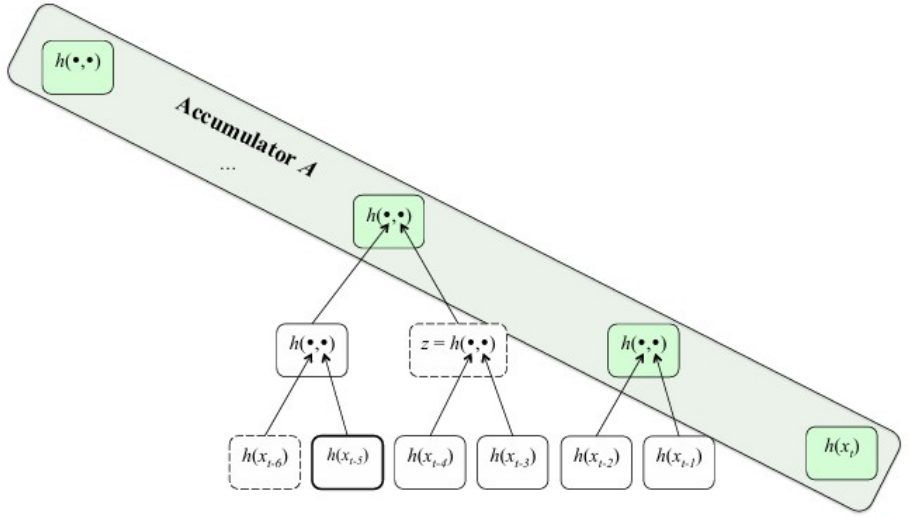


**Fig. 4.** An illustration of our accumulator. The accumulator itself is shaded; the unshaded elements are elements of the Merkle trees which are not actually a part of the accumulator. The elements with dashed outlines belong to the authenticating path for $x_{t-5}$ (which itself has a bold outline). So, the witness for $x_{t-5}$ would be $w^{x_{t-5}} = ((h(x_{t-6}), \mathsf{left}), (z, \mathsf{right}))$.

# CCF Profile

- https://datatracker.ietf.org/doc/draft-birkholz-cose-receipts-ccf-profile/
- Single tree/root, unbalanced
- Confidential Consortium Framework
  - Generic Application Ledger Platform
  - https://github.com/microsoft/CCF
- Sub-leafs
  - Full Transaction Entry – Multiple tables, some confidential
  - Commit Evidence – Persist first, release commit evidence later

```
ccf-leaf = [
     ; Byte string of size
HASH_SIZE(32)
     internal-transaction-hash:
bstr .size 32

     ; Text string of at most
1024 bytes
     internal-evidence:
tstr .size (1..1024)

     ; Byte string of size
HASH_SIZE(32)
     data-hash: bstr .size 32
   ]
```

# Next Steps

- Adopt Profiles in the SCITT WG
- Reviews