Certificate Transparency for Domain Name System Security Extensions

draft-zhang-trans-ct-dnssec-00

Dacheng Zhang
Huawei

Background

- This work follows the assumption of DNSSEC and the compromise of key signing keys are out of scope of this work.
- This work assumes the existence of inside attacker. That is, a legal owner of a zone may try to attack or circumvent other zones.
- The owner of a zone is dependent on its parent delegation via the DS record to vouch for its DNSKEY.
- Misbehavior or compromise of the parent zone directly affects the core DNS security of the child zone.
- This work enables publics to monitor the improper delegations operations of parents

Attack Scenario (1)

- Assume a zone (foo.bar.example) and its parent zone (bar.example) are owned by different organizations. Follows are the steps of an example attack that the owner of the parent zone could perform on the child zone.
  - 1. Set up a fake child DNS server
  - 2. It generates a new key signing key X1 and zone signing key X2. It uses the KSK to sign the ZSK. It uses the ZSK to sign its resource records.
  - 3. It generates a DS record for the KSK record it generated in step 2.
  - 4. The parent sign the DS RR with its zone signing key and publishes it
  - 5. Change the IP address of the DNS server of child in the associated RRs to the IP address of the fake DNS server

Attack Scenario (2)

- The child may try to periodically access the DNS server of its parent and monitor the RRs on it . However, there is still a time window between two assessments which can be taken advantage of by the parent to perform a hijacking attack and remove the bogus RRs before the child detects the attack.
- If the owner of parent is forced to publish his operations on the public CT logs, any improper behaviors will be detected eventually. Through checking the log, it is easy detect the improper issuance of RRs of his parent zone.

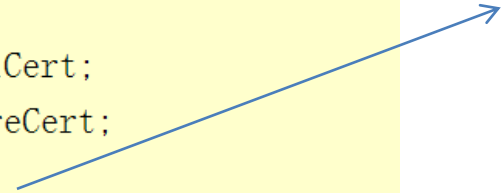
Extensions to Log Entries

```
enum { x509_entry(0), precert_entry(1), DSRR_entry(TBD1),(65535) } LogEntryType;

struct {
    LogEntryType entry_type;
    select (entry_type) {
        case x509_entry: X509ChainEntry;
        case precert_entry: PrecertChainEntry;
        case DSRR_entry:DSRR_Chain_Entry
    } entry;
} LogEntry;

opaque DNSSECRR<1..2^24-1>;

struct {
   DNSSECRR DSRR;
   DNSSECRR DNSSEC_key_chain<0..2^24-1>
} DSRR_Chain_Entry;
```

- "DSRR" is the DS RR submitted for auditing.
- "DNSSEC_key_chain" is a chain of additional DNSSEC RRs required to verify the DS RR.

Extensions to Signed Certificate Timestamp

```
struct {
    Version sct_version;
    LogID id;
    uint64 timestamp;
    CtExtensions extensions;
    digitally-signed struct {
        Version sct_version;
        SignatureType signature_type = DSRR_timestamp;
        uint64 timestamp;
        LogEntryType entry_type;
        select(entry_type) {
            case x509_entry: ASN.1Cert;
            case precert_entry: PreCert;
            case DSRR_entry: DSRR;
        } signed_entry;
       CtExtensions extensions;
    };
} SignedCertificateTimestamp;
```



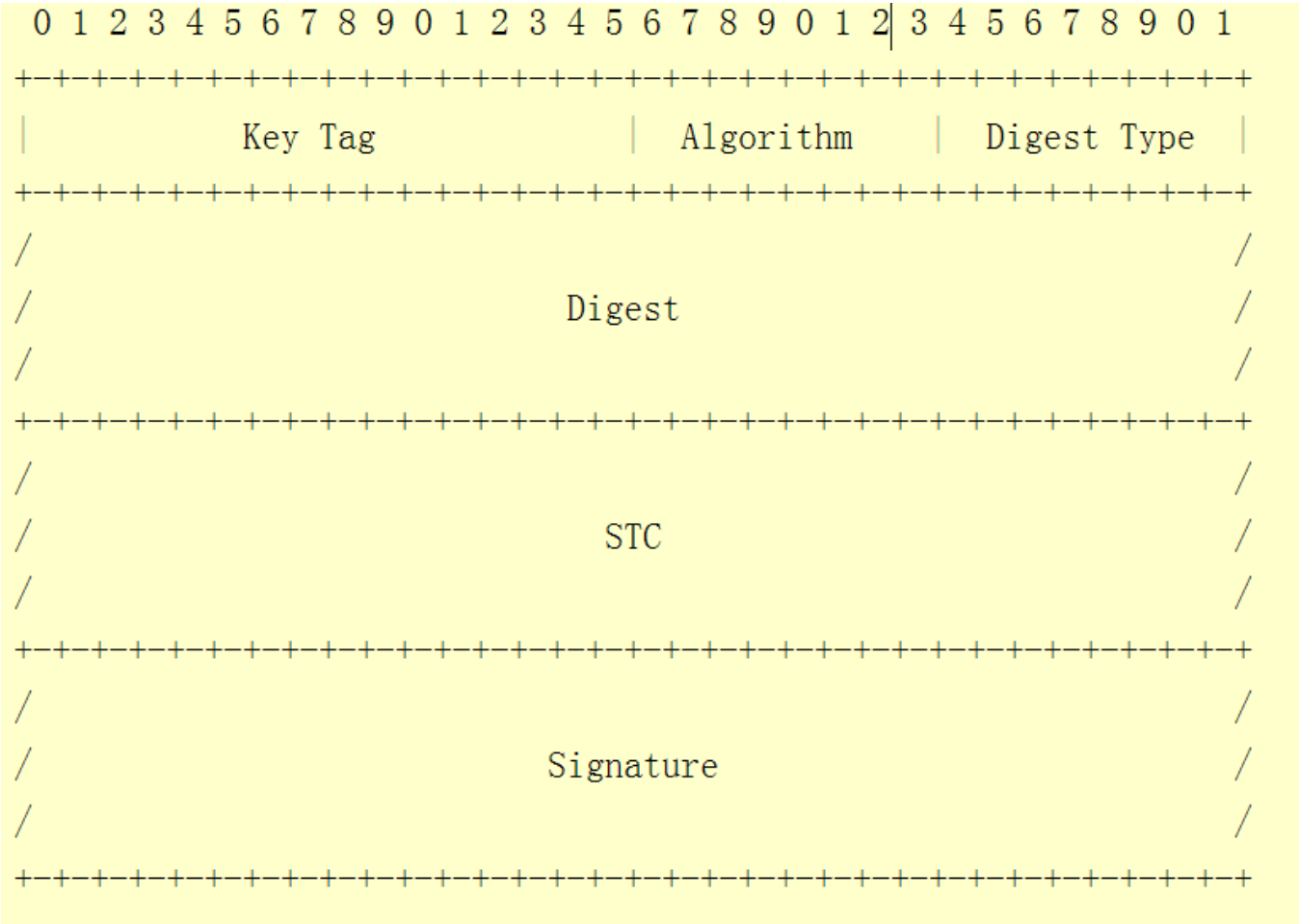
```
struct {
    opaque issuer_key_hash[32];
    C14N_DSRR dsrr;
 } DSRR;
```

Merkle Tree Input

```
struct {
    uint64 timestamp;
    LogEntryType entry_type;
    select(entry_type) {
        case x509_entry: ASN.1Cert;
        case precert_entry: PreCert;
        case DSRR_entry: DSRR;
    } signed_entry;
    CtExtensions extensions;
} TimestampedEntry;

struct {
    LeafVersion version;
    TimestampedEntry timestamped_entry;
} MerkleTreeLeaf;
```

SCT RR

- The Key Tag field includes the key tag of the DS RR referred to by the SCT record.
- The Algorithm field lists the algorithm number of the DS RR referred to by the SCT record.



Log Client Messages (1)

- Add DNSSEC RR Chain to Log
- Inputs:
  - chain: An array of base64-encoded DNS RR. The first element is the submitted DS RR; the second chains to the first and so on to the last, which is a trusted DNSKey RR.
- Outputs:
  - sct_version: The version of the SignedCertificateTimestamp structure, in decimal. A compliant v1 implementation MUST NOT expect this to be 0 (i.e., v1).
  - id: The log ID, base64 encoded.
  - timestamp: The SCT timestamp, in decimal.
  - extensions: An opaque type for future expansion. It is likely that not all participants will need to understand data in this field. Logs should set this to the empty string. Clients should decode the base64-encoded data and include it in the SCT.
  - signature: The SCT signature, base64 encoded.

Log Client Messages (2)

- Retrieve Accepted Root DNSKEY RRs
- Inputs
  - No
- Outputs
  - RRs: An array of base64-encoded DNSKEY RRs that are acceptable to the log.

Comments are welcomed!

Thank You!