Blocked Hash-Tree Status and Entitlement System

Frank W. Sudia
www.fwsudia.com
September, 2000

1. Synopsis

This paper describes a new technology for fast, lightweight, and secure confirmation of information related to digital certificates.

With the continued rapid expansion of Internet and wireless services, there is a growing need for enhanced public key infrastructure (PKI) services, including more efficient ways to confirm certificate revocation status and related entitlements.

In a wireless environment, (a) messages must be small, due to low baud rates, and (b) the computation to verify them must be minimal, due to the reduced computing resources in many hand held devices.

The **blocked hash tree** (BHT) technology uses a Merkle tree (M-tree) to encode a table of attribute values, with the M-tree's root then embedded in a certificate extension.

The attribute table is public, so for each certificate, each value is preceded by a different secret "blocker" nonce value (e.g., 160 bits). Only upon release of the blocker value can a given leaf of the M-tree be verified down to the embedded root value.

To notify potential relying parties regarding certificate **revocation** or suspension, for a given certificate, the attribute table will contain one leaf for each validity period, so the release of that leaf, with its blocker, confirms validity. If suspension service is desired, a second set of leaves can be included, with one for each possible suspension period.

To enable faster verification of attribute certificates for **entitlements** we can compile a similar table of possible user entitlements, also encoded as a blocked M-tree. The leaf attribute value may be formatted as a standard attribute certificate, with the blocker and intermediate hash values formatted as a "signature" using a new algorithm ID. These entitlement "certificates" can be either short lived or revocable.

Since there is no digital signature verification involved, the method is also suitable for revoking web **server and root certificates.** And by integrating (missing) words of assent into the process, we can force a **contracting moment** with the end-user (Tree-Wrap™).

The BHT system is compatible with the Online Certificate Status Protocol (OCSP), which allows a user-defined format for the reply message.

An overview of competing proposals for lightweight revocation has been included, along with some further implementation details for a BHT responder.

2. A Merkle Tree Primer

Merkle tree technology is employed in many cryptographic systems, such as the Surety digital time-stamping system. It can be used to tie together many messages for purposes of later authenticating them against a reliable copy of the root node.

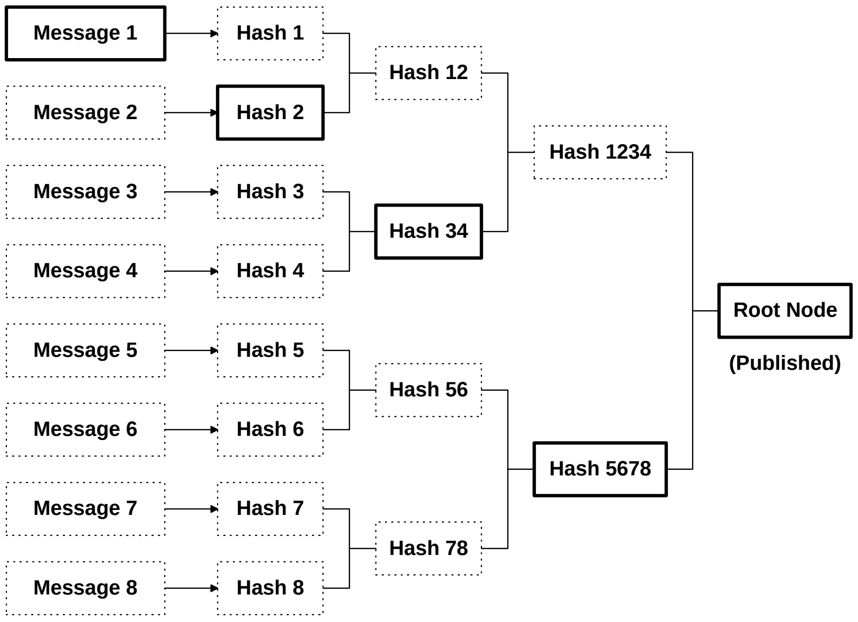

Fig. 1, Merkle Hash-Tree

Figure 1 shows a typical Merkle tree of hash values (usually a binary tree), where the leaf nodes are the data strings to be confirmed.

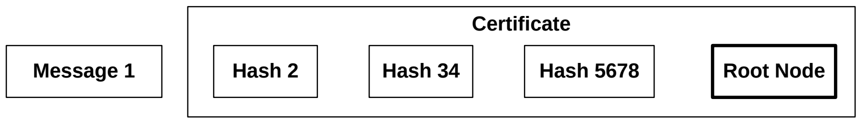

Fig. 2, Hash-Tree Certificate

Fig 2 shows the resulting Merkle tree certificate, which can be used to confirm the authenticity of a single leaf message against the published root.

To verify the certificate, the recipient uses the intermediate hash values to recreate the complete path between the message and the root node, as shown in Figure 3.

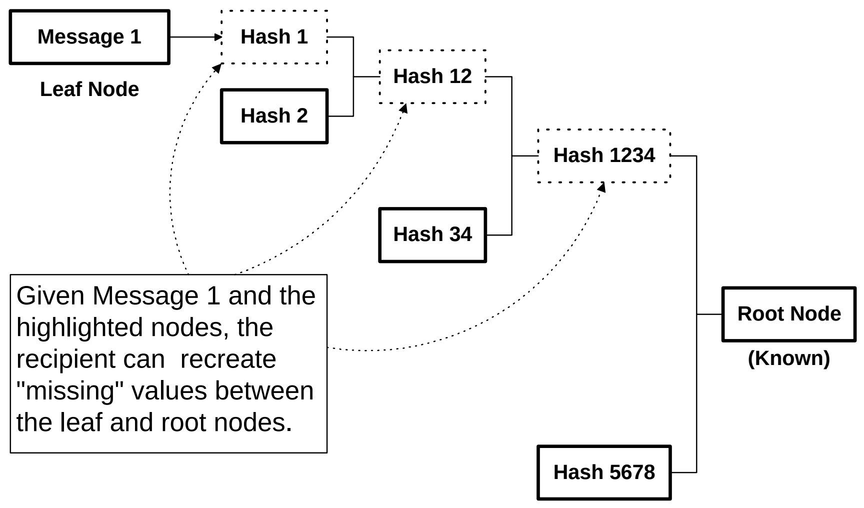
Message 1
Leaf Node
Hash 1
Hash 2
Hash 12
Hash 34
Hash 1234
Root Node
(Known)
Hash 5678
Given Message 1 and the highlighted nodes, the recipient can recreate "missing" values between the leaf and root nodes.

Fig. 3, Verification of Hash-Tree Certificate

3. Confirming Certificate Status

Consider an ordinary user or server identity certificate containing a public key, signed by a certifying authority (CA). We want to re-confirm this certificate on a periodic basis, such as daily, weekly, 2-hourly, etc. And we want this reconfirmation to be as painless as possible, as to creation, communication, and verification of status update messages, while maintaining a high degree of auditable security.

3.1. Validity Period Table as Blocked M-Tree

One way to provide authenticated information pertinent to validity and revocation for a given certificate is as follows.

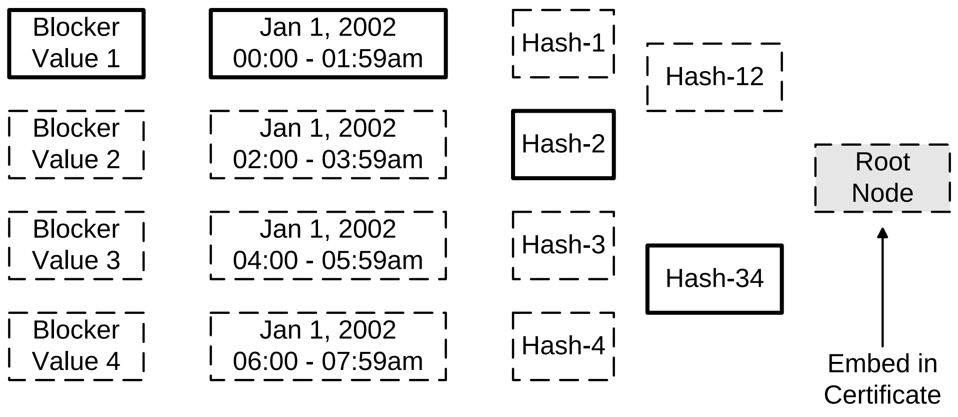

Fig 4, Blocked Hash Tree (BHT) of Certificate Validity Interval Strings

As shown in Fig. 4, a list of data strings representing future validity intervals is prepared, each prefixed by a unique blocker value, which is kept secret by the CA / Issuer. Each blocker value and validity period string combination is hashed to produce the bottom row of leaf nodes. These are then hashed up to a root node, which can be embedded into a user's certificate. The short textual statements denoting each validity interval are known in advance, but only when the CA / Issuer releases the blocking value for each table entry can it be confirmed that the CA / Issuer intended the certificate to be valid for that period.

3.2. Status Confirmation Message

Fig 5 shows the resulting status confirmation message. In theory, the current validity period string need not be provided, because the recipient can predict it. However we prefer to include it with the status message, as the additional communication required is not large, and it will greatly aid in processing and user interpretation.

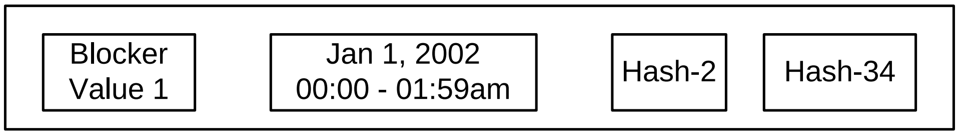

Fig 5, BHT Certificate Status Update Message

This method is distinct from Kocher[1] (hash-tree), Micali (hash-chain), and Aiello. The Kocher hash-tree system creates a separate M-tree with a digitally signed root for each period, whereas Micali and Aiello employ hash chains. In a BHT system, each status message is checked against the root node embedded in the user's certificate extension.

Unlike the Kocher hash-tree method, the responder does not digitally sign each response, but instead merely releases the appropriate secret blocker value, and the recipient need not verify a digital signature on the root node.

3.3. Blocker Value Management

There are many ways to generate and manage the blocker values, but one simple method is as follows. For each user, generate a user secret by hashing the user's certificate serial number with a master secret known only to the issuer.

user_secret = hash(issuer_master_secret | cert_serial_number) ;

where "|" is the string concatenation operator. Then hash the user secret together with the current validity period string to generate the blocker value for the current leaf.

current_block_value = hash(user_secret | current_period_string) ;

[1] A similar system was claimed by Micali in US patent 6,097,811.

Return to the requester:

current_block_value | current_period_string | needed_hash_values

This makes it easy to re-generate the blocker values (and period strings) at will, without needing to store them.

3.4. BHT Status Message Sizes

Table 1 shows expected status messages sizes for a BHT system. Typical periodicities for revocation notification intervals include weekly, daily, and 2-hourly. All hash values are assumed to be 20-byte (160-bit) data values (e.g., SHA-1). Many examples utilize 2-hour periodicity, as this is believed to be the finest granularity needed for high value financial transactions. Most other environments will require less data.

Periodicity	N Yrs	Periods	Nodes	Depth	H Bytes	T Bytes
Weekly	1	52	64	5	100	140
Weekly	2	104	128	6	120	160
Daily	1	365	512	8	160	200
Daily	2	730	1,024	10	200	240
2-Hr Lite	1	3,650	4,096	12	240	280
2-Hr Lite	2	7,300	8,192	13	260	300

Table 1. Typical Revocation Notification Periodicities and Message Sizes

For each periodicity, the table shows the number of periods, the number of binary tree nodes and tree depth, the number of "hash bytes" (H Bytes) and "total bytes" (T Bytes), assuming that the blocker value and period range label are each 20 bytes long.

For economy in the case of 2-hourly periodicity, we can lengthen some periods when the user is assumed to be asleep. For example, we can omit the notifications at 10 PM and 2 AM (user local time), while retaining Midnight (00 AM), giving 10 periods per day. The reduced period count allows better hash tree utilization without impairing quality.

The recipient never performs more than 13 hash operations to confirm the leaf message against root node from the certificate, and there are NO digital sign or verify operations, by either the responder or the recipient.

3.5. Responder Data Storage

Table 2 shows the data storage requirement per certificate for each periodicity.

Periodicity	Years	Periods	Leaves	Depth	Nodes	Bytes	Pruned
Weekly	1	52	64	5	63	1,260	
Weekly	2	104	128	6	127	2,540	
Daily	1	365	512	8	511	10,220	**320**
Daily	2	730	1,024	10	2,047	40,940	**640**
2-Hr Lite	1	3,650	4,096	12	8,191	163,820	**2,560**
2-Hr Lite	2	7,300	8,192	13	16,383	327,660	**5,200**

Table 2. Data Storage per Certificate (Bytes)

When millions of certificates must be issued and managed, storing the full tree requires too much disk space: over 300 gigabytes (GB) of data per million certificates. However, with no loss in response time, we can “prune” the bottom 5 or 6 rows of the tree, thereby reducing its size by a factor of 32X or 64X. This cuts the required disk space to 300 MB per million certificates for 1 year daily periodicity, and 5 GB per million certificates for 2 year 2-hr lite periodicity.

3.6. Detailed Comparison with Kocher and Micali

This comparison assumes the following:

- All hash values, blocker values, and data payloads are 20-bytes each.
- A digital sign or verify operation requires roughly 10,000 times the computational power of computing one hash value.
- There are 7,300 validity periods (10 x 365 x 2) in two years (using “2-hour lite,” in which we lengthen certain night periods, for a total of 10 periods per day).
- The hash-tree system is assumed to be managing 10 million certificates (depth = 24).
- For the BHT system (and also hash-chain), we can selectively omit many of the response values, to reduce storage requirements.

Note that 2 years of 2-hour periodicity is a “worst-case scenario,” for controlling access to high-value financial transaction systems.

All values estimated. Data sizes are in bytes. Work factors are given as hash-operation equivalent counts.

Revocation System Property	Blocked-Tree	Hash-Tree	Hash-Chain
Status Message Size	300	700	40
Recipient Work Factor (per lookup)	**13**	10,000	1 to 7,300
Certificate Size Increase (bytes)	40	0	40
Responder Storage (per cert, optimized)	4,000	150	4,000
Responder Work Factor (lifetime)	14,600	30,000	14,600

Table 3. Comparison of BHT with Kocher and Micali

As shown in Table 3, the BHT method yields a moderate sized message, due to the need to transmit the intermediate hash values, but the *recipient work factor remains small and fixed*, with a hash-operation count corresponding to the tree depth.

For a system with one-year daily periodicity, there are only 365 periods, yielding a tree depth (and fixed work factor) of 9, and a message size of 200 bytes.

4. Entitlement Data Structures (Auth-Tree™)

The BHT paradigm is also useful for managing entitlements, by constructing trees of user entitlement information. BHT allows many entitlements to be administered, granted and revoked for individual users without reissuing the underlying certificate, eliminates the processing cost to verify signatures on attribute certificates, and helps keep the user's entitlements secret (by not listing them directly in the certificate).

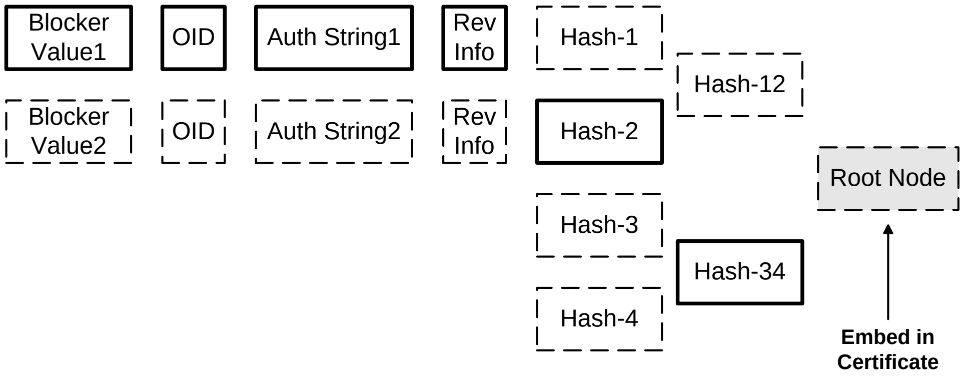

Figure 6, Creating a Simple Auth-Tree System

Figure 6 shows a typical Auth-Tree. The idea is to (a) compile a list of entitlements we may wish to grant to the user during the life of the certificate, (b) create a blocked Merkle tree, and then (c) embed the resulting root node in an extension in the user's certificate.

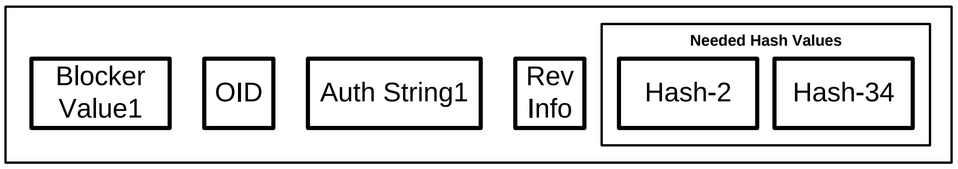

Figure 7. BHT Entitlement String Data Unit

Figure 7 shows the resulting entitlement string data unit. As before, only the elements bounded with solid lines need be sent, because the other hash values can be inferred from the ones that are sent. As shown in the prior art, the hash tree can be very deep. A table of 1,024 elements can be authenticated with a depth of 10 hashes; 1 million elements with a depth of 20; 1 billion with a depth of 30, etc. (See also Table 3, below.)

A major benefit of this approach is that the entitlement "certificate" need not be digitally signed, and the recipient need not incur the work factor of verifying a digital signature.

However, such "certificates" must still be revoked, if necessary, and for this purpose the "rev info" field can contain a serial number, or optionally another embedded BHT root node, for confirming revocation status messages. In addition, entitlements can contain a validity period field, and remain valid for short periods of time.

The BHT revocation status message discussed above is a special case in which we are confirming an entitlement to use the certificate for a given N-hour validity period.

Auth-tree "certificates" can utilize existing attribute certificate formats, as given for example in ANSI X9.45. We can achieve this by formatting the entitlement attributes as a standard attribute certificate, and then define a special algorithm ID for the "signature" field, containing the blocker and necessary hash values. These "certificates" remain unsigned, however, since they cannot be confirmed without knowing the root node from the user's identity certificate.

5. Incorporating Contract Terms (Tree-Wrap™)

In addition to conveying entitlements, we can also create a legal "contracting moment." This is similar to (but distinct from) the "unwrapping" scheme in Sudia, US 5,995,625, "Electronic Cryptographic Packaging." (The term "tree-wrap" is derived from shrink-wrap and click-wrap, which are two well-known end-user e-contracting methods.)

For example, when providing revocation status information we may desire to force the recipient to agree to certain basic e-contract terms regarding the use of the certificate, including limitations of liability and venue for legal actions, etc.

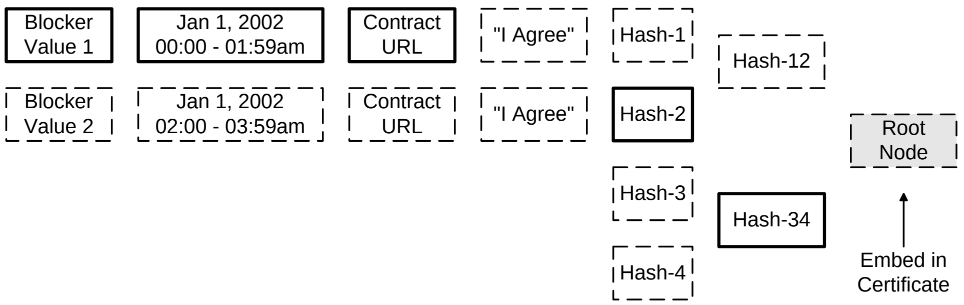

Figure 8, Tree-Wrap Applied to BHT Revocation Status

In Figure 8 we have added a contract URL to the certificate status message. The essence of this feature is that in order to verify the data object, the recipient must, as a step in the verification process, supply the missing string representing words of assent ("I Agree"). Figure 9 shows the resulting revocation status message.

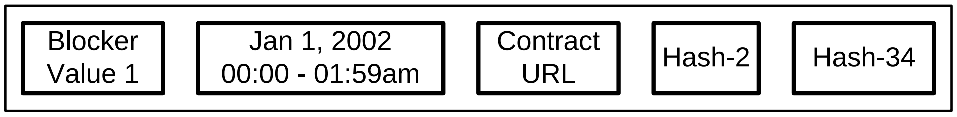

Figure 9, A Tree-Wrapped BHT Revocation Status Message

In practice, the recipient's system would prompt with an "I Agree" button, and then insert the missing text into the verification process. To support contracts in multiple languages, it is sufficient to apply a public XOR value to the hash of the contract to yield a standard value, as this occurrence remains unlikely enough to maintain the presumption of recipient knowledge and consent.

6. Implementation Considerations

6.1. Certificate Extension

The BHT system relies on a custom X.509 version 3 extension in the user certificate containing the root node, plus other information such as notification periodicity.

However, many fielded PKI products, including major browsers, web servers, e-mail clients, etc. do not truly support custom ASN.1 extensions, other than the Netscape extensions and others that achieved early and wide adoption. ASN.1 processing for custom extensions remains "buggy" and can cause fatal program errors. The cleanup of ASN.1 processing by major software vendors, even if started today, could take years to ripple through the huge installed base of client software.

Therefore, we recommend ASCII-armoring the root node extension, since most products can handle textual notice-type extensions. This will hide any nested ASN.1 constructs that could cause trouble for ASN.1 parsers. In the future, when the installed base is more mature, this armoring can be removed to conform to ASN.1 standards.

6.2. Suspension, Revocation Reason

If suspension service is desired, a second set of leaves can be included, one for each possible suspension period. This doubles the number of leaves in the M-tree, but adds only one more layer of depth, so the message size only increases 20 bytes.

We will also add a small set of "revoked" leaves, e.g., one per revocation reason, which can be transmitted to confirm positively that a certificate has been revoked.

6.3. Web Server and Root Certificates

Verifying BHT status messages is fast and does not rely on asymmetric public key operations. Hence it appears suitable for revoking web server and root certificates.

In a simple case, a web merchant can request its own BHT status message for the current period, and transmit it to all clients who request it, who in turn can quickly verify the web server's revocation status. This could be of interest to financial institutions that need to revoke unused web servers.

Similarly, a root certificate authority could elect to publish BHT updates against its root certificate to anyone requesting one. This could provide an elegant way to respond in case of catastrophic compromise, thus improving overall system auditability.

6.4. Responder Security

As with any PKI service, the security of the responder is critical. If attackers obtained the issuer master secret, or otherwise breached security, they could issue false responses. Yet Internet server security remains problematic.

We suggest a physical separation from the Internet, which can be accomplished by storing the master database offline, and copying only the potential responses for the next period to an online directory or responder. This requires 2 such online systems. We take one offline, populate it with the next period, and switch it back to the Internet. We then take the other system offline and repeat the process, alternating for each period.

7. Appendix: Alternative Proposals

For those unfamiliar with this specialized field, two competing technical proposals for fast revocation systems are briefly summarized below. (Terminology used here differs from the original reference documents.)

7.1. Kocher Hash-Tree (Valicert VA)

The Kocher hash tree model has been implemented and marketed by Valicert in their "Validation Authority" (VA) product.

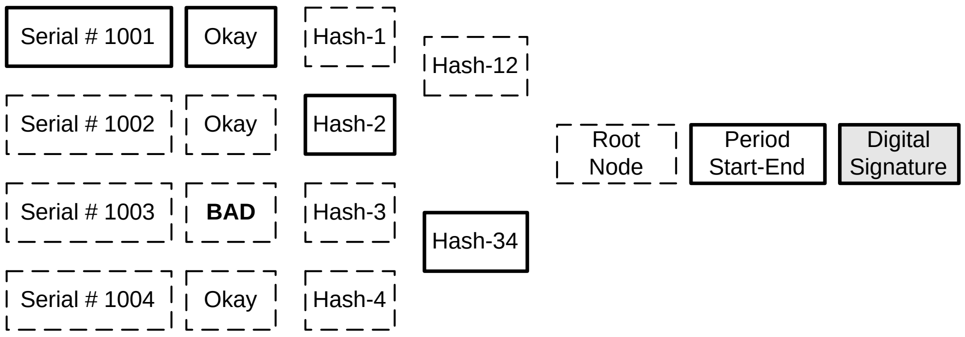

Figure 10, Kocher Revocation Tree

As shown in Figure 10, an M-tree is created from a list of all outstanding certificates and their current status. There is one table per period for the entire system, which is digitally signed. The response message is shown in Figure 11.

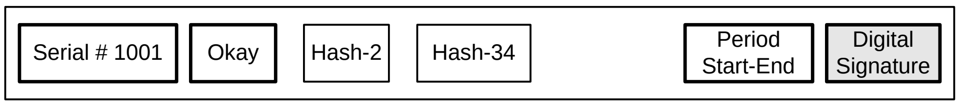

Figure 11, Kocher Response Message

Then, during the entire period, the server need not apply a separate digital signature to each response. Rather it just sends the same digital signature with every response, plus the revocation status data and necessary hash values. This is a solid advance over OCSP, because on a high volume server, the digital signing overhead would be crippling.

If the recipient is a casual user, it incurs the overhead of verifying the digital signature. However, if it is a high volume web server, which makes many requests during a given validity period, then it need only verify the root digital signature once per period. Then all subsequent requests during the period can be swiftly verified by merely hashing down each response to see if it matches the root node value. Similarly the root signature (which adds 128+ bytes to the message size) must be sent for each period, although it could be omitted from subsequent in-period responses.

The Kocher certificate table lists all possible certificates, and can readily grow to handle millions, or even billions of certificates. Each additional level of hashing depth doubles the possible size of the table.

N of Certs	Hash Depth	Hash Bytes
1,000	10	200
10,000	14	280
100,000	17	340
1 million	20	400
10 million	24	480
100 million	27	540
1 billion	30	600
10 billion	34	680

Table 4, Kocher Tree Depth by Number of Certificates

As shown in Table 4, the Kocher system can handle many outstanding certificates, while maintaining modest message sizes. The messages become longer, which may exact some price in a wireless environment. However, once a server has obtained and verified the root node for the current period, subsequent validations are cheap.

BHT status messages do not become longer as the user population grows, because their size depends only on the number of periods for that certificate. And they never incur the added length of the root signature, or its verification overhead.

7.2. Micali Hash-Chain (CRS)

The Micali "Certificate Revocation System" (CRS), which uses a hash-chain model, has not been built, but is covered by several published patents.

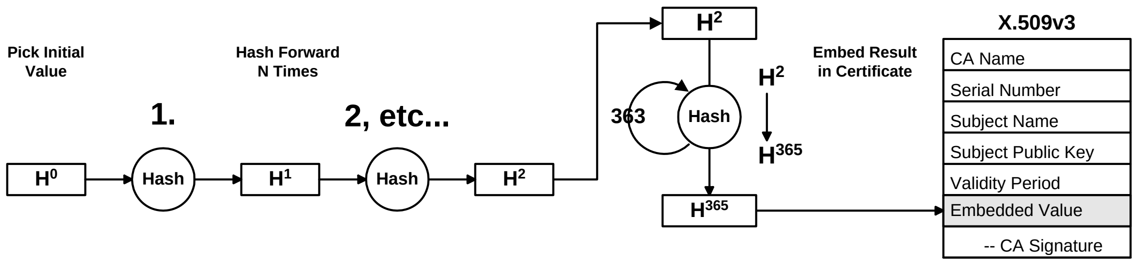

Figure 12, Creating a CRS Hash Chain (1 year, daily)

Figure 12 shows the setup process for one year of daily periodicity. One selects an initial random value (H^0), hashes it forward N times, where N equals the number of validity periods over the certificate's life, and embeds the final period value (in this case, H^{365}) into the user's digital certificate, in a custom extension.

In response to status requests, the responder releases the intermediate hash value (in reverse order) corresponding to the current period number, starting with H^{364}, working backwards one iteration per validity period.

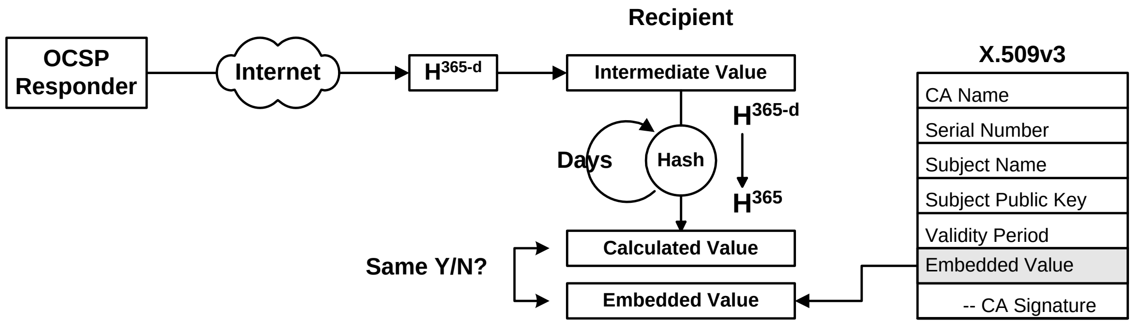

Figure 13, Checking the CRS Intermediate Value

To confirm the validity of a certificate, the recipient requests the intermediate value for the current period, hashes it forward N times (where N equals 365-D), and checks to see if it matches the embedded value.

To revoke a certificate, the responder ceases publication of the "valid" values, and may instead release a "NO" value that hashes to a different final value, also embedded in the certificate (thus increasing the size of the extension.)

CRS yields a smaller message size, since it is just one hash plus optional information to identify the period and certificate number. The number of hash operations needed to confirm the message can vary widely, starting with few at first, and growing by one each period. Among parties who transact frequently, the recipient can optimize by caching a recent prior value.

The work factor to verify a BHT status message remains small and constant, without the added complexity of caching, because the hash depth is fixed for the life of the certificate.

BHT can add many more signaling modes without increasing the extension size.

CRS cannot send authenticated text data, so the paradigm cannot be enhanced to support a parallel entitlement scheme.

8. Bibliography

Aiello et al., “Fast Digital Identity Revocation,” Proceedings of Advances in Cryptology (CRYPTO-98) Springer-Verlag Lecture Notes in Computer Science

American Bankers Association, ANSI X9.45, “Enhanced Management Controls Using Attribute Certificates”

IETF, RFC 2560, “Online Certificate Status Protocol - OCSP"

Fischer, US 5,214,702, “Public key/signature cryptosystem with enhanced digital signature certification” (entitlements)

Kocher, US 5,903,651, “Apparatus and Method for Demonstrating and Confirming the Status of Digital Certificates and Other Data” (hash-tree)

Merkle, US 4,881,264, “Digital Signature System and Method Based on a Conventional Encryption Function”

Merkle, US 4,309,569, “Method of Providing Digital Signatures”

Micali, US 6,097,811, “Tree-Based Certificate Revocation System” (sim. to Kocher)

Micali, US 5,666,421 and US 5,960.083, “Certificate Revocation System” (hash-chain)

Micali, US 5,016,274, “On-Line/Off-Line Digital Signing” (Note: claims all forms of cryptographic pre-computation.)

Sudia, US 5,659,616 and WO 96/02993, “Method for securely using digital signatures in a cryptographic system” (entitlements)

Sudia, US 5,995,625, “Electronic Cryptographic Packaging” (e-contracting)