SCRaaPS: X.509 Certificate Revocation Using the Blockchain-based Scrybe Secure Provenance System

Sai Medury, Anthony Skjellum
SimCenter and
Dept. of Computer Science
and Engineering
University of Tennessee at Chattanooga
Chattanooga, TN 37403
Email: sai-medury@mocs.utc.edu,
tony-skjellum@utc.edu

Richard R. Brooks, Lu Yu
Electrical and Computer Engineering
Clemson University
Clemson, SC 29631
Email: {rrb,lyu}@g.clemson.edu

Abstract

X.509 certificates underpin the security of the Internet economy, notably secure web servers, and they need to be revoked promptly and reliably once they are compromised. The original revocation method specified in the X.509 standard, to distribute certificate revocation lists (CRLs), is both old and untrustworthy. CRLs are susceptible to attacks such as Man-in-the-Middle and Denial of Service. The newer Online Certificate Status Protocol (OCSP) and OCSP-stapling approaches have well-known drawbacks as well.

The primary contribution of this paper is Secure Revocation as a Peer Service (SCRaaPS). SCRaaPS is an alternative, reliable way to support X.509 certificate revocation via the Scrybe secure provenance system. The blockchain support of Scrybe enables the creation of a durable, reliable revocation service that can withstand Denial-of-Service attacks and ensures non-repudiation of certificates revoked. We provide cross-CA-revocation information and address the additional problem of intermediate-certificate revocation with the knock-on effects on certificates derived thereof.

A Cuckoo filter provides quick, communication-free testing by servers and browsers against our current revocation list (with no false negatives). A further contribution of this work is that the revocation service can fit in as a drop-in replacement for OCSP-stapling with superior performance and coverage both for servers and browsers. Potential revocation indicated by our Cuckoo filter is backed up by rigorous service query to eliminate false positives. Cuckoo filter parameters are also stored in our blockchain to provide open access to this algorithmic option for detection.

We describe the advantages of using a blockchain-based system and, in particular, the approach to distributed ledger technology and lightweight mining enabled by Scrybe, which was designed with secure provenance in mind.

1 Introduction

X.509 certificates need to be revoked rapidly and reliably once compromised. The method specified in the standard, to distribute certificate revocation lists (CRLs), is old and untrustworthy. CRLs are susceptible to attacks such as Man-in-the-Middle and Denial of Service. Newer formulations—OCSP and OCSP-stapling—have well-known drawbacks too.

We describe SCRaaPS, Secure Certificate Revocation as a Peer Service, a reliable way to support X.509 certificate revocation using the Scrybe secure provenance system [7, 26]. The blockchain support of Scrybe enables the creation of a durable, reliable revocation service that can withstand Denial-of-Service attacks while ensuring non-repudiation.

The X.509v3 standard [11] provides specifications to implement the Public Key Infrastructure (PKI) system in which each entity has a public key, a private key, and a certificate that is typically assigned by a Certificate Authority (CA). Each certificate holds a validity period along with other important related information, and the CA may revoke a certificate before it expires for reasons that include change of name, change of association between subject and CA (e.g., an employee terminates employment with an organization), and compromise or suspected compromise of the corresponding private key [11].

The standard defines one method of certificate revocation in which the CA periodically issues a CRL [11]. The CRL is a data structure signed by the CA that is made freely available in a public repository and contains a time-stamped list

978-1-7281-0155-2/18/$31.00 ©2018 IEEE

of serial numbers of all revoked certificates [11]. The process of certificate validation involves checking for the certificate's serial number in the "suitably recent" CRL [11]. A major advantage of this method was that the means to issue CRLs is the same as that of issuing certificates—namely, via untrusted servers and untrusted communication [11].

If a browser does not already have a fresh copy of the CRL, then it has to fetch it during the initial connection, which can be tedious. The problem of scalability arises when the CRL becomes large; for instance, close to 50,000 certificates were revoked after the HeartBleed bug. The browsers had to download large CRLs, which slowed down most website connections [19]. The periodic nature of updating the CRLs opens a window of opportunity for attackers to work with revoked certificates until the next updated CRL becomes available, which can be an hour, a day, a week, or even a month in some cases. Additionally, there may be instances when a CRL server is unable to handle requests from clients, in which case most browsers render it as a "soft fail" [6] and accept the certificate.

SSL is susceptible to Man-in-the-Middle attack, in which an attacker can insert a CA certificate into the client's root store so that the client may communicate with the attacker's website in a way that seems perfectly secure and legitimate. Furthermore, the "CRL Distribution point" field in the certificate is an optional field and is considered noncritical [11]. An attacker would most definitely not include this field, and the certificate would be deemed valid by the browser. Additionally, an attacker using a revoked certificate with no CRL distribution point mentioned will most certainly have an advantage of not being discovered.

The remainder of this paper is organized as follows. In Section 2, we describe existing approaches to revocations, as well as limitations of these systems, including attacks on revocation. Section 3 defines the concept of secure provenance and relates its properties to the revocation problem. Section 4 explains the secure provenance system, Scrybe, based on distributed ledger (blockchain) technology. Section 5 describes our approach to utilizing Scrybe as a new means for certificate revocation (SCRaaPS). Next, we provide an analysis of why our approach to certificate revocation is superior to existing methods in Section 5. Finally, in Section 6, we offer conclusions and mention future work including our follow-on goal to move from the prototype stage to a sustainable realization of Secure Certificate Revocation as a Peer Service (SCRaaPS) that could be used in production and at scale by a variety of browsers.

2 Current Revocation Methodologies

The Online Certificate Status Protocol (OCSP) was proposed to replace CRL revocation [23]. OCSP enables a client to verify the status of a certificate dynamically during the stage of connection establishment and specifies a client-server architecture in which a client can request the revocation status of a certificate. An OCSP responder will respond with a status of good, revoked, or unknown [23]. The response was intended to be both quick and lightweight and to solve the scalability issue with large CRLs that needed to be downloaded and parsed. The window of vulnerability vanishes since the request/response is dynamic and the client always receives a response from the most updated knowledge of an OCSP responder [23]. The protocol also mentions certain standards for the encryption algorithm to be used in the request, and the responder can reject any request using a weak encryption algorithm [23].

While OCSP solved many problems inherent in CRLs, implementation was not quite up to the mark. As a single point of failure, OCSP responders incurred too many requests and became the bottleneck. They were notorious for being slow to respond [1] and, in addition, were deemed not to have good up-time [1]. Another case of poor implementation arose when a client would attempt to log in on a captive portal while waiting for an OCSP response that could be blocked by the captive portal [1] that would perhaps allow a response only after a user logged in. Web browsers like Chrome and Firefox devised a workaround to enhance user experience by implementing OCSP with a "soft-fail" approach [6] (i.e., if a certificate is within its validity time and an OCSP responder takes too long to respond, then the certificate is accepted to be valid) [1].

Recognizing all these flaws, the IETF proposed a new technique, commonly referred to as OCSP-Stapling, in which the client can request the revocation status of a certificate as part of the TLS handshake [12]. A website that enables this feature must periodically request its Certificate Authority (CA) for updated revocation status [12] and must also send the most updated information as part of the TLS handshake response. This method saves the client the burden of having to make a third-party request for revocation status, therefore resolving the problem of slowing website communication while safeguarding the client from attacks like Denial of Service. This technique would have worked well if the protocol enforced the server to always send an OCSP response, which it did not do [15]. Even though a client sends an extension "`status_request`" as part of TLS handshake (which mentions that the server must include an OCSP response in the handshake), the server may choose not to [15] append an OCSP response and the client will have to accept the certificate as valid. Alternatively, the browser client may choose to request the OCSP responder by itself, but this option would not be without all the inherent vulnerabilities of OCSP. This gap was perhaps not filled because not all CAs were equipped at that time with OCSP response capability; there would be a lot of connection failures if the clients were enforced to fail from connection establishment to a CA that has not yet implemented OCSP response methodology. Later, a modification to the TLS standard was proposed in which a client is forced to fail from establishing a connection to a server that does not respond with an OCSP response [15]; by then most CAs had evidently implemented the capability.

In a subsequent addition to the protocol, an extension “`status_request_v2`” was introduced to carry out OCSP status checking for all the intermediate certificates present in the certificate chain [22]. A client mentioning the “`status_request_v2`” extension must also mention a list data structure containing the list of certificates for which revocation status is requested. The server may respond by “stapling” the list of revocation status for all the certificates. This development was substantial in improving certificate revocation since the revocation checking of intermediate certificates was crucial and client browsers had to rely on CRLs or other methods to achieve it. Client browsers could mention trusted OCSP responders [22] and would accept an OCSP response coming from one of those responders, but they would also accept an OCSP response from an authorized OCSP responder [22]. (An authorized responder is one that has been delegated by a CA that issued the certificate for the website and signed the delegation using the same private key [15] that the CA used to issue the certificate.)

Presently, browsers implement various revocation methodologies. Mozilla Firefox moved away from CRL revocation in 2010 [10] and enforced OCSP response with CRL as a fallback. In 2013, the company announced its own methodology called “OneCRL” [14] in which a centralized revocation list containing the revocation status of all the intermediary certificates was maintained and pushed to clients periodically. Later, it also enabled OCSP stapling and currently enforces OCSP must-staple methodology. In 2012, Google Chrome announced that it would stop conducting any standard form of revocation checking like CRL or OCSP [17]. Instead, Google designed its own methodology called CRLSets. The company maintains a comprehensive internal list of crawled CRLs [8], which are mostly obtained from CAs. From this internal list, only those with no reason code or the specific reason codes (Unspecified, KeyCompromise, CACompromise, or AACompromise) [8] are published to clients. CRLs are published periodically every few hours. The implementation ensures that most or all of the intermediate certificates are part of the published CRL [8].

3 Secure Provenance

Here we define secure provenance and indicate how its properties can address the revocation problem.

Data provenance is metadata that can be used to track changes in data [24] over time and to ensure integrity. Secure provenance is achieved in a system where the integrity of provenance data can be maintained and ensured, and the metadata is always available for querying. The metadata collected in secure provenance must always follow the chronological order in which events occurred and must be immutable [3]; that is, once logged, the information must remain read-only and should not be susceptible to falsification. The system enforces accountability and non-repudiation where an entity cannot claim that it was not responsible for a change that occurred and is logged in the system [4]. Further, in case an error occurs, changes can be traced back chronologically to identify when and what triggered the change responsible for the error [4].

Having secure provenance data for revoked certificates will help ensure the integrity and availability of revocation. There is no confidentiality goal for such data. Secure provenance data will incorporate the revocation status of all the intermediate certificates including when, by whom, and why a given certificate was revoked (some can be revoked implicitly by virtue of an antecedent’s revocation). Additionally, chronologically ordered data can be implemented with data structures that provide insight into the hierarchy of the certificate chain, thereby offering convenient revocation of all the certificates that an intermediate certificate provider may have issued.

4 Overview of the Scrybe System

We explain how we verify that Scrybe supports non-repudiation and is also robust against distributed denial of service (DDoS) attacks; in particular, we explain how the Lightweight Mining (LWM) algorithm, a unique feature of Scrybe, proves resilient to such attacks.

4.1 Overview

As illustrated in Figure 1, there are two main components of the Scrybe blockchain: blocks and transactions. A blockchain is simply a sequence of linked blocks, where the current block contains the hash of the previous block.

4.1.1 Blocks

As previously mentioned, each block contains the hash of the previous block, which makes the blockchain *immutable*. Blocks are added to the blockchain by *miners*, entities responsible for maintaining the integrity of the blockchain. Scrybe only allows authorized entities to mine blocks through the secure LWM algorithm (comprising the Scrybe consortium) Miners are responsible for aggregating a list of transactions and calculating the Merkle root. The Merkle root allows other miners to quickly verify that every transaction is actually included in the block. When a

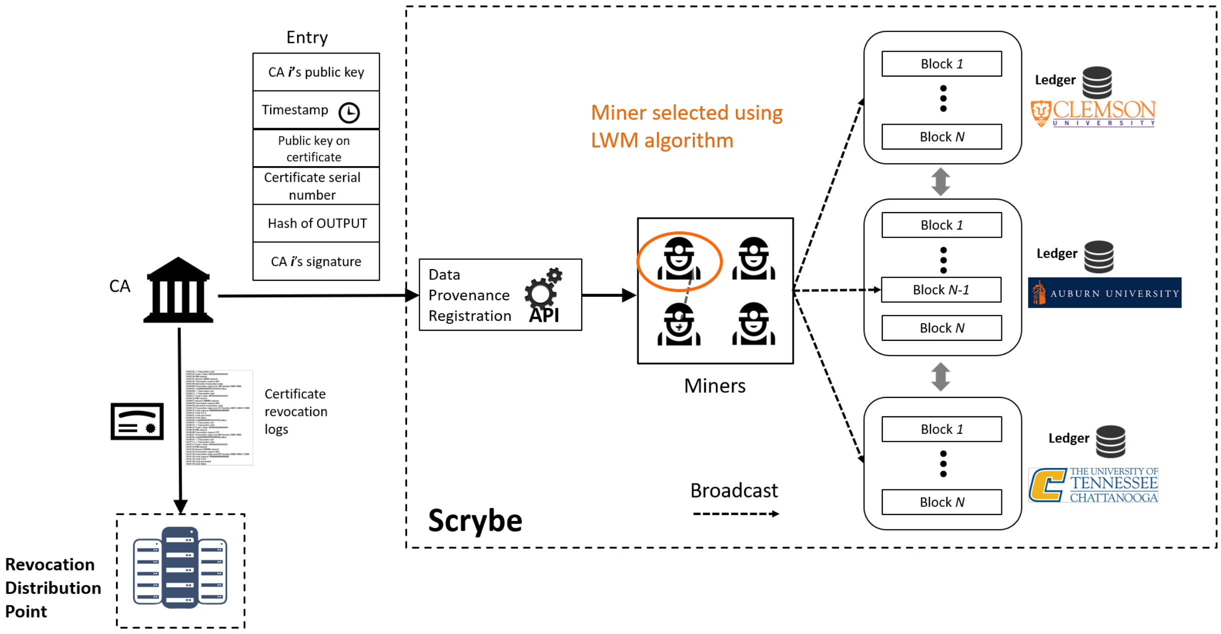
Entry
CA *i*'s public key
Timestamp
Public key on certificate
Certificate serial number
Hash of OUTPUT
CA *i*'s signature
CA
Certificate revocation logs
Revocation Distribution Point
Data Provenance Registration
API
Miner selected using LWM algorithm
Miners
Scrybe
Broadcast
Block *1*
Block *N*
Block *N-1*
Ledger
CLEMSON UNIVERSITY
AUBURN UNIVERSITY
THE UNIVERSITY OF TENNESSEE CHATTANOOGA

Figure 1. Scrybe Architecture showing a CA sending a revocation entry to the Scrybe consortium, miner selection, and the blockchain on mining servers .

miner is selected to add a block to the blockchain, the block is broadcast to all the other miners, and the data are verified (previous hash, Merkle root, and the miner's signature). At this stage, other miners will be able to detect if a transaction is omitted from the block, if an unauthorized miner broadcasts a block, and if the miner's signature is invalid.

4.1.2 Transactions

Transactions are the backbone of provenance. Conceptually, transaction input is categorized as input fields and output fields. A transaction in Scrybe takes input fields, output fields, and submitter's details including the name, public key, and signature as input. The miner adds the timestamp as part of the transaction. Transactions can also be *genesis events*, which register the acquisition of new data. The persistent URLs (PURLs) that point to the data, along with the SHA-3 hash of the data, ensure its validity. Note that all transactions have output fields; genesis events **only** have an output with no input, but normal transactions have both.

By only storing the SHA-3 hash of the transaction instead of the original transaction, we can drastically reduce the size of the blockchain; consequently, there will consequently be no penalty for an extensive number of inputs and outputs in any given transaction. The original transaction will be stored on a *transaction server*, which will be locally maintained, along with the *data server* and the *metadata server*.

Note that for the SCRaaPS application, we have changed the transaction format and scheme compared to the nominal Scrybe format. We store certificate information directly on the blockchain, which we choose to do because there is no need for a separate data or metadata server in this use case referred to by PURLs. Our Entries contain sufficient information to describe certificate revocation without reference either to an external data or metadata server. This simplification is essential because SCRaaPS is working to eliminate DoS/DDoS against revocation information. Furthermore, we store Cuckoo filter coefficients periodically on the blockchain (handled by a single trusted agent in the first implementation, and to be handled by a distributed app on the blockchain itself in the future).

4.1.3 Lightweight Mining

Scrybe introduces a novel way to mine new blocks in the blockchain, which is not a difficult proof-of-work required in cryptocurrency applications. The lightweight mining algorithm (LWM) introduced in Scrybe is presented in the following frame.

Lightweight Mining Algorithm (LWM)

Input: The number of miners N.
Algorithm: For each miner m_i, $0 \leq i < N$,

- *Step 1*: m_i generates a random number r_i;
- *Step 2*: m_i broadcasts the SHA-3 hash of r_i, denoted by $H(r_i)$;

- *Step 3*: Once m_i has collected all N hashes $\{H(r_0), H(r_1), \cdots, H(r_{N-1})\}$, m_i broadcasts r_i.
- *Step 4*: Once m_i has collected all N random numbers $\{r_0, r_1, \cdots, r_{N-1}\}$, m_i calculates $l = \sum_j r_j \bmod N$.
- *Step 5*: m_l is the selected miner to create the next block from the collected transactions. (Without loss of generality, we map $m_i = i, 0 \leq i < N$ as a simple rank ordering for the registered miners.)

The Genesis block contains the information related to initial miners, and the number of miners is fixed before the beginning of every round of miner selection. Each round has a fixed timeout for synchronization. If a miner fails to send the hash within this timeout, then that miner's hash and random number are considered to be $NULL$ for that particular round. Optimal timeout depends on the number of active participants and the desired number of transactions per round.

Considering that the network is a permissioned network, new miners wishing to join are not accepted until after the end of current round. The purpose of LWM is to provide randomization in miner selection. In a Denial of Service (DoS) attack against Scrybe, we assume a malicious miner targets a particular user by excluding the victim's transactions from the block he or she creates. The randomization offered by LWM, coupled with the fact that each miner maintains a local pool of transactions, guarantees the victim's transactions will always be integrated sooner or later, as long as there is at least one honest miner.

The core idea of LWM is "sharing-hash-first." If every miner sends out the random number without sharing the hashes first, a miner can hold his or her own number until he or she has received everyone else's random number. This caveat allows a malicious miner to manipulate miner selection by choosing a number that produces a m_l in favor of a particular miner or deliberately excludes a particular miner. Scrybe takes a naive approach for random number generation in a peer-to-peer service. Each node generates a random number independently before sharing the hash of that random number. This method has been proven to be robust [2]; as long as one node is generating a random number, the $\sum_j r_j N$ remains random.

"Sharing-hash-first" ensures that every miner has to share his or her own number (in the form of the hash) with others before they see others' choices. Since hash values are considered impossible to invert in practice, a miner cannot change the random number after the fact. Further, the hash is signed with the sender's digital signature, which disallows a miner from equivocating. Each miner may broadcast any number they wish, and it is in the interest of each sender to broadcast a random number to avoid predictability and a pattern that can be exploited to reduce the chances of the sender being selected.

Thus, LWM can tolerate up to $N-1$ malicious miners who collude. As long as there is one miner generating a random number, the modulo operation is randomized.

The LWM consensus is one-CPU-one-vote majority, same as PoW [20], but there is no need for the concept of longest chain. In the case of a miner trying to broadcast a block containing a previous hash entry that is not the same as that of the latest block, that block will not be added to the blockchain. Furthermore, there is no chance for a fork in the blockchain for Scrybe since only one miner is chosen in each round to propose the next block.

4.1.4 Servers

Locally maintained transaction servers will hold the transactions comprising the ledger. An additional metadata server can be maintained along with the transaction server wherever it makes sense. The integrity of the database can be verified by generating transaction lists for each block and ensuring that these transactions and corresponding hashes accurately display the state of the database. If there is any discrepancy, the database server is deemed disreputable. The integrity of the data and metadata can be verified by comparing the SHA-3 hash of the data to the SHA-3 hash stored in the transaction: if these hashes differ, the relevant server is considered disreputable. The method for storing data on these servers is configurable and left to the end-user's discretion.

The Certificate Authorities can maintain an optimal minimum number of transaction servers, processing OCSP-staple requests from the web servers that have certificates assigned by that CA. A quorom of all Certificate Authorities can be established with a common blokchain containing the list of all revoked certificates.

5 Secure Provenance Approach to Revocation

This section explains how secure provenance can address the CRL problem, first in general and then specifically with our Scrybe-based system using non-cryptocurrency digital ledger technology.

Scrybe can be configured to provide secure provenance as a service for the revocation lists and can be used to replace OCSP responders in the OCSP Must-staple protocol. Certificate authorities can broadcast the serial number of a certificate to be revoked along with the entity's public key and the CA's public key. This transaction would be validated and added to the blockchain. Web servers may periodically request the OCSP response for its certificate and its certificate chain using the "`status_request_v2`," and SCRaaPS can respond with the revocation status for all

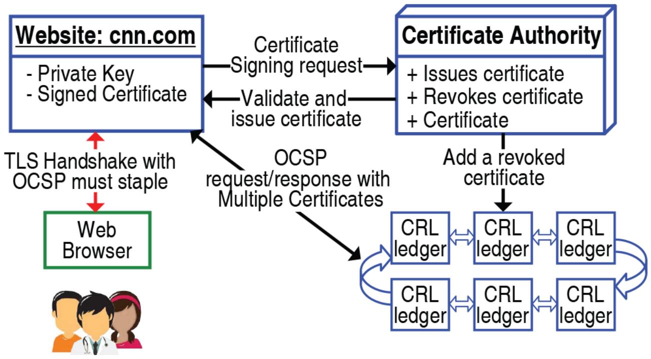

Figure 2. Revocation using Scrybe distributed CRL ledger

these certificates. The servers can staple this revocation status during connection establishment using the TLS handshake. Furthermore, Scrybe can act as a public bank of revocation statuses for any client that may wish to check revocation for any number of certificates.

The distributed nature of the Scrybe blockchain safeguards revocation data from Denial-of-Service attacks. Assuming that each web server has a list of an optimal minimum number of SCRaaPS servers' IP addresses, if one Scrybe server does not respond within a reasonable timeout, then the next Scrybe server in the list may be contacted for revocation status. An attacker should never be able to take down all the servers once there are a sufficient number, and there is no way to know for sure which server will respond to a given client's request. In OCSP revocation, the client request would enumerate a list of OCSP responders [23] that the client trusts, and an attacker may choose to launch a Denial of Service attack on these responders. Such a scenario does not arise with SCRaaPS, provided many Scrybe miners form the consortium that implements the blockchain in a production deployment.

The LWM algorithm enables building the blockchain quickly: all the mining servers will have the most updated blockchain within a reasonable delay. With Scrybe, the latency in updating the CRL is low, and the window of opportunity for an attacker to make use of a revoked certificate consequently reduces substantially.

To disseminate revocation information in a more scalable and low-latency manner, a set membership data structure can be utilized for quick lookup.

A Bloom Filter is an m-bits-long binary data structure indicating whether an element a_i is a part of a list [25] A=$\{a_1$,a_2,...$a_n\}$. An element is fed into k hash functions to determine which bits to set in the Bloom Filter data structure. False negatives are impossible, but false positives may occur. The probability of a false positive [25] is as follows:

$$(1 - e^{kn/m})^k \tag{1}$$

A cuckoo filter, like Bloom Filter, is a probabilistic data structure that can be used to look up set membership [9]. The cuckoo filter supports removing members and a bounded false positive rate for practical applications while maintaining similar space complexity as Bloom Filter [13]. It is a compact variant of cuckoo hash table [21] that stores fingerprints of member elements. With dynamic relocation of elements and delete capability [13], cuckoo filter is the perfect probabilistic data structure for SCRaaPS. Optimal parameters must be evaluated for negligible false positive rate.

Utilizing a set membership data structure for querying revocation status can significantly reduce the time taken to retrieve the response. The coefficients can be stored on the blockchain and updated periodically and can be cached by clients (web servers and web browsers). The test indicates with certainty if a certificate is **not** part of the revocation list but may have a false positive (key properties of Cuckoo filters). The data structure bits may be downloaded on the client browser, further speeding up the process. Also, in case a certificate returns positive, the browser may fetch the updated response from the cuckoo filter on Scrybe to eliminate a false-positive outcome.

Web browsers will be able to utilize the same process to check the revocation for intermediate certificates instead of their self-created database of intermediate certificates. For example, Mozilla can utilize the SCRaaPS service provided by Scrybe instead of CCADB (a repository of information about intermediate CAs and root certificates [18]) for Firefox, and Google could use SCRaaPS instead of CRLSet for Chrome. Instead of competing to provide revocation, Mozilla and Firefox could contribute and support to a common service with better overall revocation coverage.

Our plan is to update the cuckoo filter coefficients on an event-driven basis after every ten certificates are revoked. Typically [16], there are 200-300 certificates revoked per day. More frequent updates could also be possible. But, we consider that in addition to using the Cuckoo filter, an extremely high security client could download all blocks that have come after the most recent filter data block and allow a linear search of up to nine more recent revocations (in some cases, we may have more than one entry per block, but that doesn't detract from this concept). Those blocks can be discarded each time a new set of Cuckoo filter coefficients is updated.

In addition to creating and validating blocks, each SCRaaPs server will be responsible for the creation of this Cuckoo filter. Organizations may choose to add edge servers as pruned nodes (a node that does not participate in mining) to store the most updated part of the blockchain

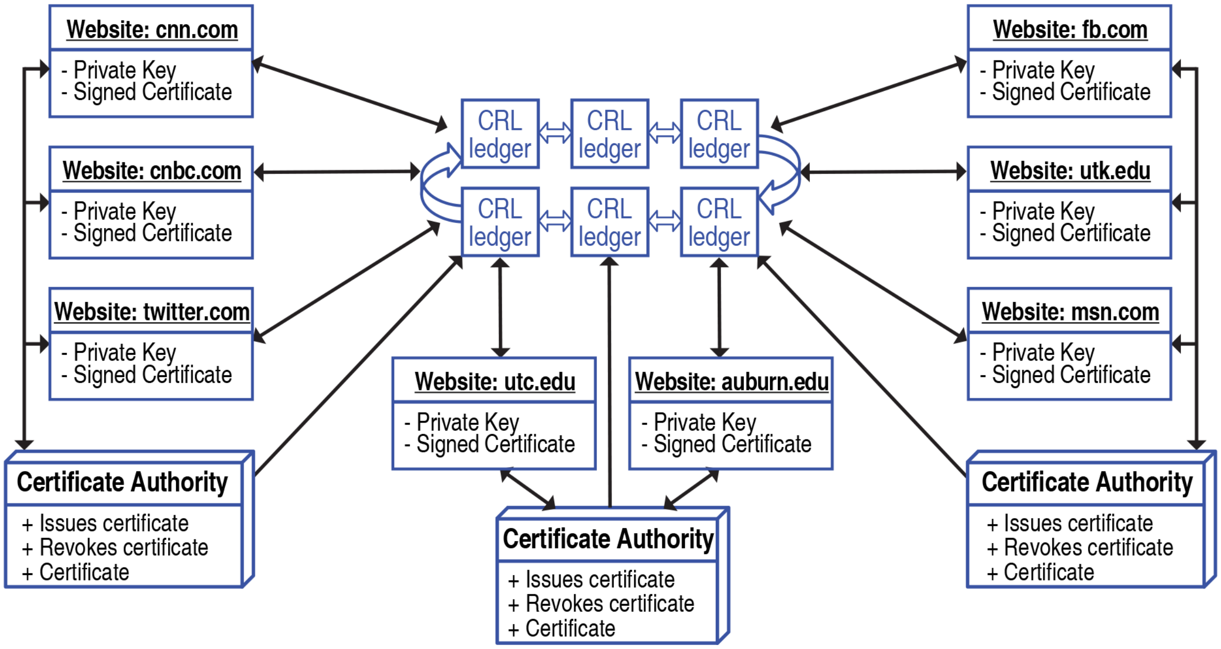

Figure 3. Multiple CAs adding revocation to Scrybe. The revocation service is not just CA-centric. Multiple CAs have equal opportunity to contribute revocation information to our open revocation service. Because of the blockchain, integrity and availability of revocation data, once recorded, is assured.

[5], and these servers may be queried for quick revocation response.

In an ideal scenario in which SCRaaPS is implemented and utilized by CAs across the world, there would be a uniform distributed ledger containing revoked certificates from all CAs and a single, robust distributed service that all browsers/web servers can rely upon for revocation status.

6 Conclusion and Future Work

We introduced SCRaaPS, Secure Certificate Revocation as a Peer Service, as an alternative, reliable method and system to support X.509 certificate revocation, accomplished using the Scrybe secure provenance system. Scrybe is based on distributed ledger technology (a blockchain with lightweight mining). X.509 certificates need to be revoked promptly and reliably once they are compromised. The method specified in the X.509 standard to distribute CRLs is old and untrustworthy. CRLs are susceptible to attacks such as Man-in-the-Middle and Denial of Service. The newer OCSP and OCSP-stapling approaches have drawbacks as well. Such issues were described above.

Scrybe's blockchain support enables a durable, reliable revocation service that can withstand Denial-of-Service attacks and ensure non-repudiation. Our architecture provides cross-CA-revocation information. We address the problem of intermediate-certificate revocation. We described the advantages of using a blockchain-based system, and in particular, Scrybe's advantages.

Future work includes defining a deployment strategy of a production-grade, self-sustaining implementation of SCRaaPS that scales out to enable practical utilization Internet-wide, plug-ins for popular web browsers to enable wide utilization, and possible secondary connection of the Scrybe distributed ledger technology to Ethereum. Ethereum could be harnessed to provide a payment or crypto token via smart contracts associated with revocation activities in order to create a self-sustaining public service and dissuade nuisance accesses of the system. A study of the economics of the revocation system will be relevant to ensure practicality of the system and understand required incentives for all the actors to participate positively. Potential benefits and drawbacks of making SCRaaPS a permissioned service should also be studied (aka SCRaaP^{2}S). The potential requirement to standardize the service, vs. providing a de facto service, should also be considered in future work.

It would also be feasible for SCRaaPS to enable an entity to revoke its own certificate by broadcasting a transaction using its private key for that certificate. This possibility could further reduce the delay in revocation by removing the extra step of notifying the CA. The concept "Right to

Revoke" may be refined to avoid unnecessary or accidental revocation. Similarity to the goals of Certcoin would be studied in this regard.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant Nos. 1547164, 1547245, and 1821926. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] M. Almeshekah. Proposal for better revocation model of ssl certificates. https://wiki.mozilla.org/images/e/e3/SSLcertRevocation.pdf, 2013. [Online, Accessed:2/21/2018].

[2] B. Awerbuch and C. Scheideler. Robust random number generation for peer-to-peer systems. *Theor. Comput. Sci.*, 410:453–466, 2006.

[3] M. Benchoufi, R. Porcher, and P. Ravaud. Blockchain protocols in clinical trials: Transparency and traceability of consent. *F1000Research*, 6, 2017.

[4] M. Benchoufi, R. Porcher, and P. Ravaud. traceability of consent [version 3; referees: 1 approved, 2. 2017.

[5] Bitcoin.org. Bitcoin core version 0.11.0 release notes. https://github.com/bitcoin/bitcoin/blob/v0.11.0/doc/release-notes.mdblock-file-pruning, 2015. [Online, Accessed:3/4/2018].

[6] H. Bock. The problem with ocsp stapling and must staple and why certificate revocation is still broken. https://blog.hboeck.de/archives/886-The-Problem-with-OCSP-Stapling-and-Must-Staple-and-why-Certificate-Revocation-is-still-broken.html, 2017. [Online, Accessed:2/14/2018].

[7] R. Brooks and A. Skjellum. Using the blockchain to secure provenance meta-data (a CCoE webinar presentation), June 2017. Technical presentation. NCCoE seminar.

[8] Chromium.org. Crlsets. https://dev.chromium.org/Home/chromium-security/crlsets. [Online, Accessed:2/26/2018].

[9] A. Chumbley and C. Williams. Cuckoo filter. https://brilliant.org/wiki/cuckoo-filter/. [Online, Accessed:8/15/2018].

[10] M. F. community. Ca:improving revocation. https://wiki.mozilla.org/CA:ImprovingRevocation. [Online, Accessed:2/15/2018].

[11] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. RFC 5280, RFC Editor, May 2008. http://www.rfc-editor.org/rfc/rfc5280.txt.

[12] D. Eastlake. Transport layer security (tls) extensions: Extension definitions. RFC 6066, RFC Editor, January 2011. http://www.rfc-editor.org/rfc/rfc6066.txt.

[13] B. Fan, D. G. Andersen, M. Kaminsky, and M. D. Mitzenmacher. Cuckoo filter: Practically better than bloom. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, pages 75–88. ACM, 2014.

[14] M. Goodwin. Revoking intermediate certificates: Introducing onecrl. https://blog.mozilla.org/security/2015/03/03/revoking-intermediate-certificates-introducing-onecrl/, 2015. [Online, Accessed:1/29/2018].

[15] P. Hallam-Baker. X.509v3 transport layer security (tls) feature extension. RFC 7633, RFC Editor, October 2015.

[16] ics.sans.edu. Certificates revoked per day. https://isc.sans.edu/crls.html. [Online, Accessed:3/12/2018].

[17] ImperialViolet. Revocation checking and chrome's crl. https://www.imperialviolet.org/2012/02/05/crlsets.html. [Online, Accessed:2/4/2018].

[18] Mozilla.org. Common ca database. [Online, Accessed:3/6/2018].

[19] P. Mutton. Heartbleed: Revoke! The time is nigh! "https://news.netcraft.com/archives/2014/04/15/revoke-the-time-is-nigh.html". [Online, Accessed:3/9/2018].

[20] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[21] R. Pagh and F. F. Rodler. Cuckoo hashing. *Journal of Algorithms*, 51(2):122–144, 2004.

[22] Y. Pettersen. The transport layer security (tls) multiple certificate status request extension. RFC 6961, RFC Editor, June 2013. http://www.rfc-editor.org/rfc/rfc6961.txt.

[23] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 internet public key infrastructure online certificate status protocol - ocsp. RFC 6960, RFC Editor, June 2013. "http://www.rfc-editor.org/rfc/rfc6960.txt".

[24] Technopedia. Data lineage. https://www.techopedia.com/definition/28040/data-lineage. [Online, Accessed:2/27/2018].

[25] Wikipedia.org. Bloom filter. https://en.wikipedia.org/wiki/Bloom_filter. [Online, Accessed:3/8/2018].

[26] C. Worley, L. Yu, R. R. Brooks, J. Oakley, O. Hambolu, A. Skjellum, A. Altarawneh, J. S. Obeid, L. Lenert, K. Wang, and U. Mukhopadhyay. Scrybe: A 2nd-generation Blockchain technology with Lightweight Mining for secure provenance and related applications. 2018. under review for IEEE Blockchain 2018, Halifax.