To investigate the implications of global support for CA alternatives, we perform a comprehensive performance evaluation of the Convergence architecture. A key insight of our approach is that real-world SSL traffic can be mapped to certificate validation attempts in alternate trust models. We apply this insight to a university traffic capture of over 2 million SSL flows. Our contributions can thus be summarized as follows:

- **Synthesized traffic for CA Alternatives:** We map a university-wide trace of HTTPS traffic into a realistic workload for a Convergence Notary. Our approach can be generalized to other SSL trust enhancements such as DANE [18], AKI [19], Certificate Transparency [20], and Sovereign Keys [10].

- **Benchmark Convergence Architecture:** We apply this traffic in real time to a Convergence notary. We analyze the results for general performance, cache utilization, and network overhead, discovering that a single notary scales to upwards of 300 requests per second in the general case, and 1000 requests per second using fully cached results.

- **Costs to End Users:** We benchmark client costs on both continental and globally distributed notary distributions, and find that the average time to validate a certificate with a continental notary deployment is 260 ms, and as little as 108 ms on a return visit to the same domain.

The rest of this paper is organized as follows. In Section 2 we provide background on SSL, Certificate Authorities, and recently proposed CA alternatives. In Section 3, we perform a rigorous performance analysis of a Convergence notary deployment, and in Section 4 we consider the performance of convergence from the perspective of an end user. We summarize our findings in Section 5. In Section 6, we introduce libraries for the development of Convergence-aware applications, and in Section 7 we conclude.

## 2. BACKGROUND & RELATED WORK

Establishing destination identity on the Internet is a surprisingly difficult problem. In 1995, Netscape introduced the SSL protocol family (Secure Socket Layer) [17] to provide confidentiality, integrity, and identity to network communications. Identity assurance is a critical property in SSL, as its absence leaves communications vulnerable to impersonation and Man-in-the-Middle (MitM) attacks. As a solution to the identity problem, SSL introduced the Certificate Authority (CA) system, an industry that facilitates the verification of website certificates (public keys) through a Public Key Infrastructure (PKI). The CA business model is a domain identity service. Using a protected private key, CAs sign the certificates of customers that wish to prove their identity on the Internet. CA public keys, which are distributed in browsers and operating systems, can then be used to verify the authenticity of the server certificate presented when an SSL connection is established.

Independent of the known difficulties of SSL use and development [2, 4, 5, 13, 15, 32], the CA trust model introduces many additional problems. First, Vendor CA selection is largely an opaque process. Browser and operating system

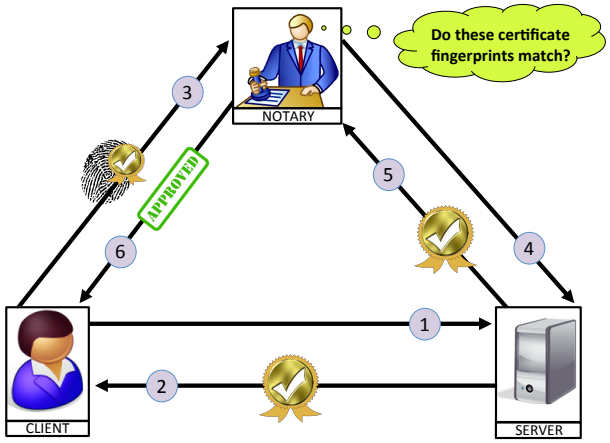


Figure 1: Multipath Probing (e.g., Convergence) leverages network path diversity to verify SSL certificates.

vendors do not justify the CAs they include in their products, and even well meaning agents (e.g., Mozilla) accept the vast majority of CAs to ensure compatibility with Internet services [11], while end users are limited to their ability to override vendor trust decisions. Second, CAs are not sufficiently incentivized to protect their customers. Even in cases where a CA is subject to a catastrophic security breach, it may still avoid serious repercussions [6, 22]. Third, each CA has unlimited scope, meaning that any CA can certify any certificate. As a result, conscientious businesses that certify with reputable CAs are just as at-risk for attack [33]. Additionally, CAs have a history of losing and licensing their private keys. DigiNotar, Comodo, and VeriSign have all had their private keys stolen or hijacked [22]. There is even evidence of certificate authorities that provide wiretap services or license their private keys [16, 29, 30, 31], creating a conflict of interest regarding customers' security needs.

Proposed CA enhancements and alternatives were extensively surveyed by Clark and van Oorschot [7], who identify *families* of proposals based on their underlying fundamental principles of operation. These alternatives vary widely in terms of both their advantages and limitations, reflecting differences of opinion on the fundamental problems with the CA trust model. An important subset of these security enhancements possess the property that they require no server side changes in order to be adopted; primitives such as multipath probing [3, 22, 34], client-based key pinning [27, 10, 23, 31], and certificate revocation lists [26, 28, 9, 20] are eligible for immediate deployment by individual users, providing tangible security enhancements to today's Internet threats. This aspect of multi-path probing motivated our selection of Convergence for this study.

### 2.1 Convergence

Convergence, based largely on Perspectives, is a method of using multi-path probing to establish a domain identity [22, 34]. Convergence clients verify a site's certificate by comparing it to the certificates obtained by trusted *notaries* that have accessed the target site via different network paths. By comparing the certificates, Convergence can probabilistically detect the presence of a MitM attack, with increasing confidence as the notary set's network path diversity increases. This procedure *replaces* traditional signature ver-

ification in multi-path probing systems; hence, even when a site's certificate was issued by a CA, Convergence need not trust that CA's public key in order to validate the certificate. Convergence offers the property of *trust agility* for the end user. Trust agility is comprised of two properties: first, the ability to re-evaluate trust decisions at anytime without repercussion; second, an ability for each individual to select their own trust anchors [22].

While Convergence Notaries can actually be extended to make use of a variety of verification methods [24], in this work we use the term "Convergence" to succinctly refer to the `NetworkPerspectivesVerifier` of the Convergence System. A simplified example of this verifier is pictured in Figure 1: The `Client` attempts to initiate an SSL session with the `Server` (1), and receives a certificate as part of the SSL handshake (2). Rather than verifying the CA signature on the certificate, the `Client` sends a request to the `Notary` containing the domain, port, and certificate hash of the `Server` (3). The `Notary` contacts the `Server` via a different network path (4) and receives another copy of the certificate (5). If the certificate hashes match, the `Notary` endorses the certificate to the `Client` (6). If the certificates do not match, the `Client` is alerted to the possibility of a MitM attack. Convergence draws its strength from network path diversity, so in practice a client queries many notaries before deciding whether to trust a certificate.

Convergence's primary improvements over Perspectives are performance and privacy. Convergence eliminates a *notary lag* problem by maintaining certificate fingerprint caches at both the notary and client. When a fingerprint is already present in the notary's cache, the target domain need only be contacted in the event of a mismatch between the cache entry and the client-presented fingerprint. When a fingerprint is present in the client's cache and the cache entry matches the site-presented certificate, the client need not contact a notary at all. Since mismatches provide a natural cue for refreshing stale cache entries, the lifespan of a cached fingerprint can be set on the order of months [24]. Convergence also enhances user privacy through use of a request-bouncing scheme that prevents a malicious notary from discovering the browsing habits of clients.

# 3. BENCHMARKING NOTARY SERVICES

We now investigate the costs of running a Convergence notary service. The primary question we wished to answer was: *Can the Convergence architecture efficiently validate the majority of SSL connections in a large network?* To answer this question, we made use of a Convergence workload that was synthesized from actual HTTPS traffic from a major university. We then ran the workload against a notary service in real time, benchmarking the notary for general performance, cache utilization, and network overhead.

## 3.1 Finding Convergence Usage Data

An ideal investigation of any alternate CA proposal would include an empirical study of its usage in the wild, but this is presently impossible for Convergence due to a lack of users and infrastructure. However, through consultation with a major university's network services, we overcame this obstacle – we obtained anonymized netflow data of HTTPS traffic from the university's network gateway, which we then used to synthesize a realistic Convergence workload at university scale. The traffic trace was from a link that was connected to three outbound networks: a third party Internet service provider, an in-state inter-university network, and Internet 2. Aggregated on this link was all of the outbound IPV4 HTTPS traffic for the entire university, including the campus wireless network, department buildings, and residence halls. The capture occurred on Monday, April 21st, 2013, starting at noon local time. Source IP addresses in our dataset were deterministically anonymized before we received the data. Destination IP addresses were left unchanged. Our use of this data was in accordance with university policy, and their network services department approved this paper before submission.

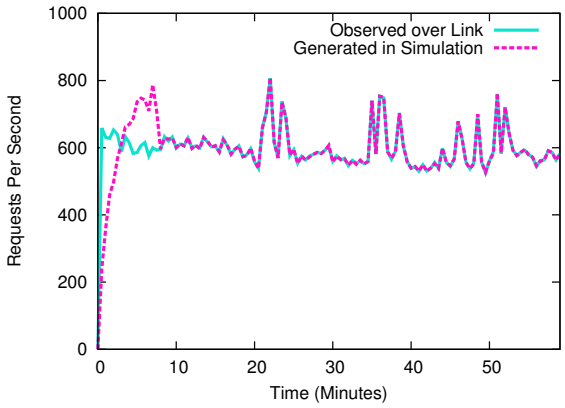


Figure 2: Convergence requests per second during synthesized university workload.

Our method for mapping SSL traffic into a Convergence workload was as follows. Using `nfdump`, we filtered for TCP traffic with destination port 443, giving us per-flow statistics for duration, bytes per second, and packets per second. In total, there were 3,922,967 flows in the trace. We then performed reverse DNS lookups on the destination IP addresses. Flows with destination IP addresses that were not resolvable were at this point discarded. This left a total of 2,139,632 flows, 54% of the original dataset. We performed `whois` lookups on the unresolvable IP addresses and found that the vast majority belonged to content distribution networks; 33% of the addresses were registered to Apple.com, but presented certificates with canonical names of the form `*.apple.com.akadns.net`, indicating that Akamai was proxying the SSL connection. We attribute the failed reverse DNS lookups to the request-routing techniques used by CDN-managed domains. We note also that Liang et al.'s recent study details the systemic conflicts between the end-to-end nature of HTTPS and the man-in-the-middle nature of CDN [21].

The full workload is visualized in Figure 2. The generated traffic lagged behind observed traffic in requests per second during the first 3 minutes, after which the simulator was able to accurately model the workload for the remainder of the trial. The traffic reflected a great diversity of SSL usage, including content distribution network domains such as Akamai (24%), cloud services like Amazon AWS (9%), and personal use domains such as Facebook (14%), Twitter (4%), and Spotify (0.2%).

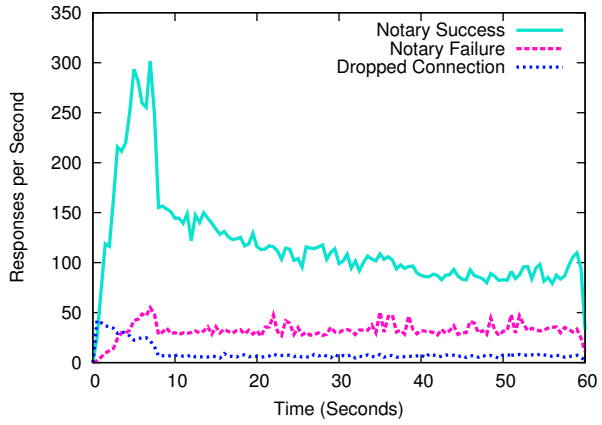


Figure 3: Notary performance against workload.

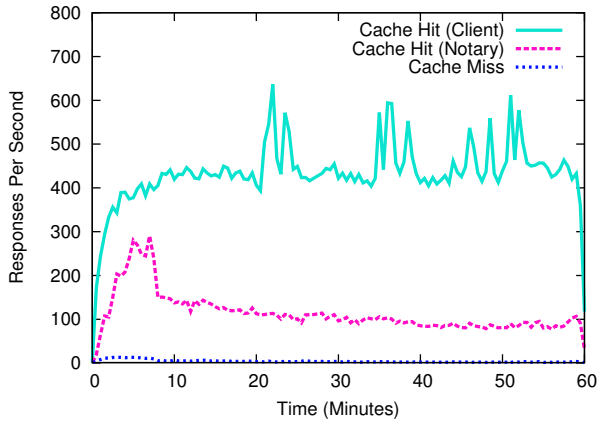


Figure 4: Caching frequency at Client and Notary.

| Notary Response/Result | Count | Percent |
|---|---|---|
| Notary: 202 Accepted | 423,438 | 74% |
| Notary: 503 Service Unavailable | 116,000 | 20% |
| Client-Notary Connection Error | 33,837 | 6% |
| Client Timeout | 803 | <1% |

Table 1: Results for every Notary request in our workload, grouped by response code.

| Event | Count | Perc. |
|---|---|---|
| Client: Cached Domain | 1580087 | 77% |
| Notary: Cached Domain | 436085 | 21% |
| Cache Miss | 38704 | 1% |

Table 2: Cache utilization breakdowns for University workload. This table considers only successful requests, as failed requests cannot be cached.

## 3.2 Testbed

We ran a series of benchmarking tests against a Convergence 0.4 notary. The notary was deployed on a Dell PowerEdge R610 server with two 4-core Intel Xeon E5606 processors and 12 GB RAM. The operating system used the Red Had Linux 2.6.32 kernel and was connected to a university network using twisted-pair copper transmitting in 1000 BaseT. The notary exclusively made use of the `NetworkPerspectiveVerifier` method. Several small changes were made to the convergence-0.4 source before launching the server. Convergence–0.4 is implemented in Python's Twisted library as a multithreaded web server, but is not multicore as implemented. We inserted `fork` commands during initialization such that there were two Convergence processes per core, all of which were listening on the same ports; we found that this did not introduce instability or corrupt the shared cache database. Convergence-0.4 is also configured to have a 10 second timeout on its outbound connections; however, many of the requests in our dataset were to bad domains that were no longer running an HTTPS service, creating excessive delay. To account for this, we reduced the timeout to 2 seconds. We did not find that reducing the timeout decreased the number of successful requests.

Elsewhere in the same subnet, we deployed several VMs to represent several thousand Convergence users, issuing hundreds of requests per second to the notary. VMs were provisioned sufficiently to ensure that any delay was the result of a bottleneck at the notary; in practice, we found that 3 VMs were sufficient for this task. For each request, the VM recorded the status code returned by the notary and the length of time between the request and response timestamps. The VMs simulated the client-side caching mechanism by maintaining a separate cache for each source IP address (*src_ip*); when the VM found a (*src_ip*, *dst_domain*) cache hit, it moved on without issuing a new notary request for domain *dst_domain*.

## 3.3 General Performance

Distributing synthesized Convergence requests across our client VMs, we simulated a full hour of university SSL traffic. As noted above, we also simulated a client-side caching mechanism that reduced the number of requests issued to the notary from 2,139,632 to 574,078; we discuss cache performance in greater depth in Section 3.4. Over the course of the entire trial, the average number of notary requests per second was 160.

The breakdown of notary responses from the university tests are found in Table 1. **202** codes indicate that the Notary successfully contacted the target domain and validated the certificate fingerprint. **503** codes indicate that the Notary-Target connection failed because the domain no longer ran an HTTPS service on port 443; many of these codes were from cloud services' VMs (e.g. Akamai, Amazon AWS) whose domains still resolved, but presumably had been transferred to new customers that were not using port 443. Overall notary performance is summarized in Figure 3. A number of connections were dropped by the notary due to sudden bursts of requests, the majority of which occurred in the first 7 minutes of the workload. A few connections were also cancelled due to a client-side timeout.

Over the course of the trial, the notary reached upwards of 300 correctly handled requests per second. To arrive at a tighter upper bound for the performance of our deployment, we ran a series of Apache bench trials [1] in which the notary targeted a web server in the same LAN. We found that our notary could field up to 1051 requests per second with its caching enabled, and up to 199 requests per second with caching disabled.

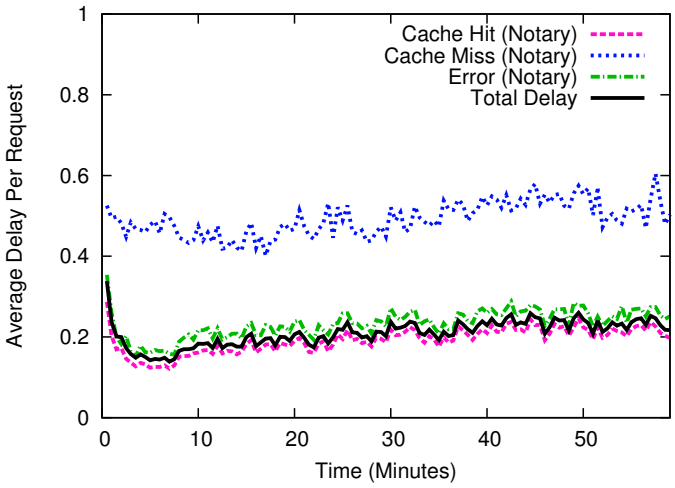


Figure 5: Average delay per minute for requests under different caching conditions.

| Cache Utilization | Average Time |
|---|---|
| 65% | 16 sec |
| 70% | 25 sec |
| 75% | 27 sec |
| 80% | 35 sec |
| 85% | 122 sec |
| 90% | 271 sec |

Table 3: Time required to reach various notary cache utilization levels, starting from the beginning of the trial.

### 3.4 Cache Utilization

The proportion of client cache hits, notary cache hits, and cache misses from the university workload can be seen in Figure 4 and Table 2. 79% of the requests triggered our simulated client-side cache mechanism, and hence did not require communication with the notary. For the requests that did reach the notary, almost all of them used a cached result. Only 0.7% of requests in the workload actually required the notary to contact the target domain. Figure 5 shows the performance improvement of the notary caching mechanism – for a cache miss, the notary took approximately 0.5 seconds to respond, whereas only 0.2 seconds were required to respond when a cached fingerprint was found.

We also recorded the speed with which the notary reached various levels of cache utilization. For each second of the trial, we calculated the number of cache hits per second as a ratio of the total number of successful requests. We considered the cache to have reached a certain utilization only once it maintained a given hit rate for the remainder of the trial; that is, the cache hits per second rate did not drop below that level later in the trial. As shown in Table 3, the notary reached 90% cache utilization in under 5 minutes, meaning that notaries rarely need to contact the target domain in order to validate certificate fingerprints.

### 3.5 Network Overhead

As Convergence replaces a cryptographic signature check with additional network activity, another important consideration is the potential impact on the network. We can express this impact as a measure of the overhead imposed on each SSL session:

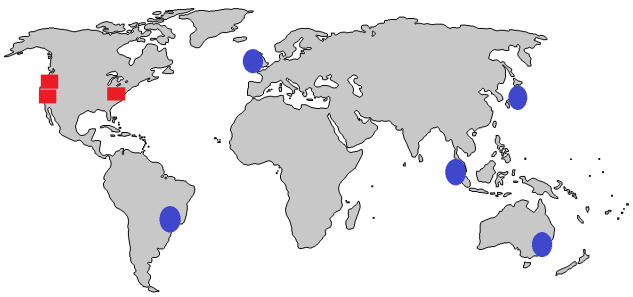

Figure 6: Notary services run on EC2 instances. Red squares mark a continental notary (size 3). Our global set used both the red dots and blue circles (size 8).

$$Overhead = \frac{Notary\ Bytes}{SSL\ Bytes} \quad (1)$$

In order to discover *Notary Bytes*, we performed packet captures at the notary using *tcpdump* while running the university workload. The capture filtered for packets that were both inbound and outbound for port 443. We then summed together the length of each captured frame at a per-second granularity. *SSL Bytes* was approximated from the original netflow data, which provided us with a start time, duration, and bits per second for each captured flow. For each second, we discovered the flows that were currently active, then summed their bits per second. If a flow ended during a given second, only a fraction of the bits per second was summed. This resulted in a bandwidth overhead of 0.06% for Convergence: the total amount of SSL traffic during the trial was 2.77 TB, while the corresponding Convergence traffic was calculated to be 1.73 GB.

## 4. ESTABLISHING END USER COSTS

We now consider the performance of Convergence from the client perspective. In multi-path probe systems, domain identity assurances stem from the use of multiple, geographically disparate notaries.[3] As the number of network vantage points increases, the likelihood of an undetected MitM attack becomes vanishingly small. This claim is based on a threat motel in which persistent large-scale MitM attacks are not practical [34]. To arrive at a performance cost estimate for the end user, we created two notary sets (Figure 6); the first was a continental set of 3 notaries in the United States, and the second was a global set of 8 notaries. Notaries were run from Ubuntu server micro instances at Amazon EC2 sites in the United States, Ireland, Singapore, Japan, Australia, and Brazil. To simplify the benchmarking process, we did not use the available Firefox Plug-In, but instead issued Convergence API requests to the notaries via `wget`, caching results locally in a `sqlite3` database. Our client was an Ubuntu desktop that repeatedly attempted to validate a domain that was at a fixed location in North America.

[3]Clients reconcile conflicting results from different notaries by specifying a consensus policy.

| Event | Notaries | Real Time |
|---|---|---|
| Local Cache Hit | N/A | 108 ms [107, 110] |
| Notary Cache Hit | Continental | 743 ms [742, 744] |
| Notary Cache Hit | Global | 1,508 ms [1503, 1513] |
| Notary Cache Miss | Continental | 1,006 ms [996, 1017] |
| Notary Cache Miss | Global | 1,790 ms [1762, 1818] |

Table 4: Benchmarks for Convergence client usage. 95% confidence intervals are included in brackets.

The results, displayed in Table 4, show the real time in milliseconds for the request to return. The minimum cost of Convergence usage was 108 ms, which occurred when a domain's certificate fingerprint had previously been cached on the client machine. We attribute this relatively high baseline cost to the fact that our benchmarking client re-loaded the cache database for each execution. The remaining four rows of the table show the cost of Convergence when a request was issued to the notaries. At worst, when both local and notary caches miss, the cost of a request using a global set was approximately 1.8 seconds. These benchmarks establish upper and lower bounds for the cost of Convergence during normal operation.

An additional cost to consider is the storage size of the client cache. Each cache entry requires 20 bytes of space for a cache fingerprint, plus a small character string to represent the associated domain and port. For example, google.com's entire cache entry would require just 34 bytes. This means that a Convergence cache can store 100,000 records in approximately 3.2 MB. Cache records remain usable forever unless the cached domain changes its certificate, at which point the client contacts the notaries to refresh the record.

# 5. REMARKS

Based on the findings of Sections 3 - 4, we conclude that Convergence is an efficient means of certificate validation, in spite of additional network communication over traditional signature verification. Our deployment was able to handle over 1000 requests per second. Based on the fact that a single server was able to validate the SSL certificates of an entire university, we feel that Convergence has many strengths as a candidate for widespread deployment. A summary of the security properties of Convergence and other multi-path probing systems can be found in [7].

Convergence's high throughput is due in large part to the effectiveness of its client and notary caches. The notary caching mechanism quickly became saturated during a more realistic SSL workload, and the local caching mechanism proved highly effective, dramatically reducing the cost of system usage to the end user. 75% of all validated domains used locally cached fingerprints, requiring no request to a Convergence notary. Of the remaining requests, 92% made use of the notary's cache.

Based on the frequency of cache utilization from Section 3.4, and the benchmarks from Section 4, we can arrive at an expected validation delay for the end user. We estimate that the average time per certificate validation during our trial was 260 ms. When visiting a locally cached domain, validation required just 108 ms.

# 6. DEVELOPMENT LIBRARIES

We recognize the long-term importance of developer support to the widespread deployment of Convergence. Convergence usage is currently specified as a REST API whose only implementation is as a Firefox plug-in. This is problematic because the Convergence Client API exposes the application developer to low-level artifacts, creating inconvenience and the opportunity for configuration errors. To address this, we are releasing 7 new Convergence clients for development in C, Java, PHP, Python, Perl, Ruby, and Lua. Through the C implementation, Convergence can also be embedded in C++, Objective-C, and other C runtimes. Each library contains a `converge(host, port, sha1sum)` that can be used in place of a `SSL_get_verify_result()` call, masking the complexity of Convergence's local cache and API. All configuration for the Convergence notaries are stored in flat files in `/usr/local/etc/converge`, making adding and managing notaries a simple process.

# 7. CONCLUSION

Widespread adoption of SSL trust enhancements like Convergence have in part been hampered by a lack of understanding of their performance characteristics under realistic conditions. We have performed a thorough case study of the Convergence architecture, evaluating its performance in simulation as the sole certificate validation service of a major university. Under realistic conditions, we found that Convergence was able to handle up to 300 requests per second. Under those same conditions end users would experience a barely perceptible average performance delay of 260 ms, and only a 108 ms delay on return trips to a domain. Our methodology is generalizable to the majority of recently proposed Certificate Authority alternatives and enhancements, allowing for a more objective and informed discussion on the future of authenticity on the Internet.

## Acknowledgments

We would like to thank Paul van Oorschot, Jeremy Clark, Patrick Traynor, and Boyana Norris for their valuable comments and insight, and Abdul Alkhelaifi for his assistance with the development libraries. This work is supported in part by the US National Science Foundation under grant numbers CNS-1118046 and CNS-1254198. Braden Hollembaek was funded in part through an NSF REU supplement.

# 8. REFERENCES

[1] ab - Apache HTTP server benchmarking tool. Available: `http://httpd.apache.org/docs/2.2/programs/ab.html`.

[2] D. Akhawe, B. Amann, M. Vallentin, and R. Sommer. Here's My Cert, So Trust Me, Maybe? Understanding TLS Errors on the Web. In *Proceedings of the 22nd International World Wide Web Conference (WWW 2013)*, Rio de Janeiro, Brazil, May 2013.

[3] M. Alicherry and A. D. Keromytis. Doublecheck: Multi-path verification against man-in-the-middle attacks. In *Computers and Communications, 2009. ISCC 2009. IEEE Symposium on*, pages 557–563. IEEE, 2009.

[4] B. Amann, R. Sommer, M. Vallentin, and S. Hall. No Attack Necessary: The Surprising Dynamics of SSL Trust Relationships. In *ACSAC '13: Proceedings of the 29th Annual Computer Security Applications Conference*, Dec. 2013.

[5] C. Brubaker, S. Jana, B. Ray, S. Khurshid, and V. Shmatikov. Using Frankencerts for Automated Adversarial Testing of Certificate Validation in SSL/TLS Implementations.

[6] R. Carly. Internet Security provider Comodo's CEO Named "Entrepreneur of the Year" by Info Security Products Guide. Available: `http://www.comodo.com/news/press_releases/2011/02/comodo-CEO-entrepreneur-of-the-Year-infosecurity-global-excellence-award.html`, February 2011.

[7] J. Clark and P. C. van Oorschot. SoK: SSL and HTTPS: Revisiting Past Challenges and Evaluating Certificate Trust Model Enhancements. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, San Francisco, CA, May 2013.

[8] P. Ducklin. The TURKTRUST SSL Certificate Fiasco – What Really Happened, and What Happens Next? Available: `http://nakedsecurity.sophos.com/2013/01/08/`Available: `the-turktrust-ssl-certificate-fiasco-what-happened-and-what-happens-next/`, January 2013.

[9] D. Eastlake et al. Transport Layer Security (TLS) Extensions: Extension Definitions. 2011.

[10] P. Eckersley. Sovereign Key Cryptography for Internet Domains, 2011.

[11] J. Edge. Mozilla and CNNIC. Available: `http://lwn.net/Articles/372386/`, February 2010.

[12] C. Ellison and B. Schneier. Ten Risks of PKI: What You're Not Being Told About Public Key Infrastructure. *Comput Secur J*, 16(1):1–7, 2000.

[13] S. Fahl, M. Harbach, H. Perl, M. Koetter, and M. Smith. Rethinking SSL Development in an Appified World. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 49–60, New York, NY, USA, 2013. ACM.

[14] D. Fisher. Microsoft Revokes Trust in Five Diginotar Root Certs. Wired. Available: `http://threatpost.com/microsoft-revokes-trust-five-diginotar-root-certs-mozilla-drops-trust-staat-der-nederland-cert`, September 2011.

[15] M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh, and V. Shmatikov. The most dangerous code in the world: validating SSL certificates in non-browser software. In *Proceedings of the 2012 ACM conference on Computer and communications security*, CCS '12, pages 38–49, Raleigh, NC, USA, 2012. ACM.

[16] I. Grigg. VeriSign's Conflict of Interest Creates New Threat. *Financial Cryptography*, 1, September 2004.

[17] K. Hickman and T. Elgamal. The SSL protocol. *Netscape Communications Corp*, 501, 1995.

[18] P. Hoffman and J. Schlyter. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. Technical report, RFC 6698, August, 2012.

[19] T. H.-J. Kim, L.-S. Huang, A. Perring, C. Jackson, and V. Gligor. Accountable Key Infrastructure (AKI): A Proposal for a Public-Key Validation Infrastructure. In *WWW '13: Proceedings of the 22nd International World Wide Web Conference*, Rio de Janeiro, Brazil, May 2013. International World Wide Web Conferences Steering Committee.

[20] B. Laurie, A. Langley, and E. Kasper. Certificate Transparency. *Available: ietf. org-Certificate Transparency (06.01. 2013)*, 2013.

[21] J. Liang, J. Jiang, H. Duan, K. Li, T. Wan, and J. Wu. When HTTPS Meets CDN: A Case of Authentication in Delegated Service. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, San Jose, CA, May 2014.

[22] M. Marlinspike. SSL and the Future of Authenticity. *Black Hat USA*, 2011.

[23] M. Marlinspike. Trust Assertions for Certificate Keys. 2013.

[24] Marlinspike, M. Convergence – Running a Notary. Available: `https://github.com/moxie0/Convergence/wiki/Running-a-Notary`.

[25] E. Mills. Comodo: Web Attack Broader Than Initially Thought. CNET. Available: `http://news.cnet.com/8301-27080_3-20048831-245.html?part=rss&tag=feed&subj=InSecurityComplex`, March 2011.

[26] M. Myers. Revocatoin: Options and challenges. In *Financial Cryptography*, pages 165–171. Springer, 1998.

[27] Psyced.org. Certificate Patrol. Available: `http://patrol.psyced.org/`.

[28] R. L. Rivest. Can We Eliminate Certificate Revocation Lists? In *Financial Cryptography*, pages 178–183. Springer, 1998.

[29] R. Sandvik. Security Vulnerability Found in Cyberoam DPI Devices (CVE-2012-3372). Available: `https://blog.torproject.org/blog/security-vulnerability-found-cyberoam-dpi-devices-cve-2012-3372`, July 2012.

[30] R. Singel. Law Enforcement Appliance Subverts SSL. Available: `http://www.wired.com/threatlevel/2010/03/packet-forensics`, March 2010.

[31] C. Soghoian and S. Stamm. Certified Lies: Detecting and Defeating Government Interception Attacks Against SSL. In *Financial Cryptography and Data Security*, pages 250–259. Springer, 2012.

[32] D. Sounthiraraj, J. Sahs, G. Greenwood, Z. Lin, and L. Khan. SMV-HUNTER: Large Scale, Automated Detection of SSL/TLS Man-in-the-Middle Vulnerabilities in Android Apps. In *Proceedings of the 19th Network and Distributed System Security Symposium.*, 2014.

[33] N. Vratonjic, J. Freudiger, V. Bindschaedler, and J.-P. Hubaux. The Inconvenient Truth About Web Certificates. In B. Schneier, editor, *Economics of Information Security and Privacy III*, pages 79–117. Springer New York, 2013.

[34] D. Wendlandt, D. G. Andersen, and A. Perrig. Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing. In *USENIX 2008 Annual Technical Conference*, ATC'08, pages 321–334, Boston, MA, 2008.