Policing Privacy ▪ Dynamic Cloud Certification ▪ Security for High-Risk Users

IEEE SECURITY & PRIVACY

BUILDING DEPENDABILITY, RELIABILITY, AND TRUST

IEEE Symposium on Security and Privacy

March/April 2016
Vol. 14, No. 2

IEEE

IEEE computer society
CELEBRATING 70 YEARS

IEEE Reliability Society

Bake in .onion for Tear-Free and Stronger Website Authentication

Paul Syverson | US Naval Research Laboratory
Griffin Boyce | Berkman Center for Internet & Society at Harvard University

Although their inherent authentication properties are generally overlooked in the shadow of the network-address hiding they provide, Tor's .onion services might just deliver stronger website authentication than existing alternatives.

Tor is a widely popular infrastructure for anonymous communication (www.torproject.org). Millions of people use Tor's thousands of relays for unfettered, traffic-secure Internet access. Approximately 95 percent of Tor bandwidth traffic is on circuits connecting Tor clients to servers that are otherwise accessible on the Internet.[1] Tor also provides protocols for connecting to services on its reserved top-level domain .onion, which are only accessible via Tor.

Tor's .onion design continues the original onion-routing idea of protecting not only clients' but also servers' network location information.[2,3] Research to date has been so focused on the location-hiding aspects of onionsites and services that it simply calls them "hidden servers." The popular press sometimes uses "Dark Web" to refer to onionsites, but more often than not, usage of that term is misleading or incoherent. Because spies and criminals attack users from hiding spots throughout the infrastructure on today's Internet, rather than being dark, Tor's authenticated routing overlay typically provides users the only visibility of or control over where their traffic goes. Thus, we challenge the common narrow view of onionsites. In this article, we explore how individuals might use Tor's .onion infrastructure to create website authentication, integrity, and other guarantees more simply, easily, fully, and inexpensively than by currently available means.

Tor and Onion Services: A Brief Background

In this article, we sketch the basics of Tor onion services. For more details, we refer readers to Roger Dingledine and his colleagues' Tor design paper,[2] the Tor Project's high-level graphical description of onion services (www.torproject.org/docs/hidden-services.html.en), and related documentation on the Tor homepage (www.torproject.org). The "Tor Rendezvous Specification" also provides a more up-to-date and much more technical description of onion service protocols (https://gitweb.torproject.org/torspec.git/tree/rend-spec.txt).

Tor clients randomly select three of the roughly 7,400 Tor relays to create a cryptographic circuit to connect to Internet services (https://metrics.torproject.org/networksize.html). Because only the first relay in the circuit sees the client's IP address and only the last (exit) relay sees the destination's IP address, identification is separated from routing. To offer an onion service, a Web (or other) server creates Tor circuits to multiple introduction points that await clients' connection attempts. Clients wanting to connect to a particular onion service use the onion address to look up its introduction points in a directory. In a successful interaction, clients and onionsites both create Tor circuits to a client-selected *rendezvous point*. The rendezvous point mates their circuits, which then interact over the rendezvous circuit like ordinary Web clients and servers.

Because a properly configured onionsite communicates only over the Tor circuits it creates, this protocol hides its network location—thus the name "hidden service." But the .onion system has other important features, including self-authentication. The onion address is actually a hash of the onionsite's public key. For example, if users want to connect to the DuckDuckGo (https://duckduckgo.com) search engine's onion service, they use the address 3g2upl4pq6kufc4m.onion. The Tor client, recognizing this as an onion address, knows to use the above protocol rather than pass the address through a Tor circuit for DNS resolution at the exit. Avoiding a DNS resolution outside the Tor network protects against leakage of client interests by preventing observation of DNS lookups as well as against any of the well-known DNS hijinks, such as redirection by ISPs or rogue DNS servers and cache poisoning. The public key corresponds to the key that signs the directory system's list of introduction points and other service descriptor information. In this way, onion addresses are self-authenticating.

For services such as DuckDuckGo, the onion service's value lies not in its location hiding but in the Tor connection's additional authentication and assurance of improved route security. Because the Tor circuits necessary to reach introduction and rendezvous points are there to protect the confidentiality of server network location, their complexity, latency, and network overhead aren't needed to provide improved authentication or route security. Nonetheless, there are performance advantages to providing an onion service to users wanting to connect to a site via Tor (for example, skirting the effects of exit relay bandwidth scarcity). And Tor proposals (the Tor equivalent of the Internet Engineering Task Force's [IETF's] RFCs) to standardize simplified onion services without location hiding are underway. Facebook's onion service already uses such simplifications.

Knowing to Which Self to Be True

DuckDuckGo's onion address is self-authenticating in that it binds the service descriptor information to 3g2upl4pq6kufc4m.onion. Presumably, users want assurance that they're reaching DuckDuckGo and receiving DuckDuckGo search results, not what might be returned by some other, possibly malicious, server. In addition to the integrity guarantee, users rely on authentication so that their queries are revealed only to DuckDuckGo. The onion address alone doesn't offer this. Using the traditional Web trust infrastructure, Facebook offers a DigiCert certificate for its onion addresses to ensure that users aren't misled by onionsites purporting to be official.

Although cryptographic binding is essential to the technical mechanisms of trust, users also rely on human-readable familiarity, for example, that the browser indicates graphically that they've made a certified encrypted connection as a result of typing "facebook.com" into the browser. To some extent, it's possible to make use of this familiarity in onionspace. By generating many keys whose hash had "facebook" as the initial string and then searching the full hashes for an adequately felicitous result, Facebook obtained the facebookcorewwwi.onion address. However, this method won't work widely, because it's difficult to generate custom addresses in this way.

The Onion Name System is an attempt at a system for globally unique but still human-meaningful onionsite names.[4] This has the advantage of not depending on existing naming schemes, such as the domain registration system. Nevertheless, we can leverage the effective usage and infrastructure that existing naming approaches have evolved through experience and design. We focus herein on approaches that link onion addresses to already meaningful ways of referring to sites. In particular, we focus on a case in which an individual controls a registered domain name, although it's also possible to bind to other meaningful Web locations such as a Facebook page or WordPress blog.

If you have a registered domain name, why not just obtain certificates from traditional authorities, as Facebook has done? For many server operators, getting even a basic server certificate is just too much of a hassle. The application process can be confusing. It usually costs money. It's tricky to install correctly. It's a pain to update.[5] These are not original observations. Indeed, that description is actually a quote from Josh Aas's first blog entry for Let's Encrypt, a new certificate authority dedicated, among other things, to making TLS certification free and automatic for most websites.

Setting up a certificate using the existing X.509 public-key infrastructure system can take hours or even days. When a collective or organization operates the website, SSL/TLS certificates have been known to take months because of ownership and authorization questions. This time cost is in addition to the certificate's monetary cost, if any. In contrast, setting up an onionsite takes a few minutes and costs nothing. Once Tor is installed, you simply add two lines to your torrc file to define where Tor will store the onion service's key information and port, if necessary. Then, simply start Tor for the key and address to be generated. To migrate the service elsewhere, add these files to the new machine, then configure and start Tor as before. The Tor Project provides additional tips and advanced options (www.torproject.org/docs/tor-hidden-service.html.en). Even if Pretty Good Privacy (PGP) encryption is used for the binding (as we describe later) and the process of learning to create a PGP key and signature is considered, the time investment is dramatically less than with X.509.

As of this writing, Let's Encrypt services are available only in beta release. Nonetheless, it's already quite popular and successful. Should it be willing to offer onion domain certificates, Let's Encrypt could be an easy way for onionsite operators to take advantage of the traditional certification infrastructure. This is already a focus of Let's Encrypt discussions, both internally and with its community (https://community.letsencrypt.org/t/if-when-will-le-support-onion-addresses/341/10).

"We explore how individuals might use Tor's .onion infrastructure to create website authentication more simply, easily, fully, and inexpensively."

Traditional SSL certificate problems go beyond questions of cost and convenience. The trust hierarchy is opaque to direct usage, and the sheer number of trusted authorities is large enough to be of concern. In particular, there have been numerous man-in-the-middle (MITM) attacks through certificate manipulation as well as hacking of certificate authorities or certificate validation software leading to use of fraudulent certificates for several popular websites.[6]

The Electronic Frontier Foundation's SSL Observatory (www.eff.org/observatory) monitors and documents the occurrence of such problems. Google's Certificate Transparency (www.certificate-transparency.org) effort is similar but broader, adding, among other things, append-only signed public logs that make undetectable certificate shenanigans harder to achieve.

The problems with certificates, though real, are largely moot for those wanting to create onion services. As of this writing, the Certification Authority (CA)/Browser Forum (https://cabforum.org) has approved only extended validation (EV) certificates for .onion addresses. This limits the certificates' use to those with the significant time, money, and desire required to complete the extensive identity validation process. EV certificates are primarily used by large businesses; individuals, organizations, and small businesses more commonly obtain domain validation (DV) certificates, which typically require a simple email confirmation based on information in the WHOIS database.

Furthermore, the .onion top-level domain itself was unofficial until recently. However, an IETF RFC reserving .onion as one of the handful of special-use domain names was approved as a proposed standard in October 2015.[7] With this RFC's official release, the approval of certificates for .onion addresses is now on firmer footing.[8]

Our Onions Ourselves

As noted, onionsites already provide self-authenticated binding of public keys to onion addresses—but not to something recognizably associated with that site. We seek an authentication solution for all websites, especially moderately popular or short-lived ones such as webpages for individuals, hometown sports teams, one-time local events, small businesses, and municipal election campaigns. Although these are smaller targets than the more popular, long-lived sites, they're subject to similar controversies and attacks. Even if they aren't the targets of attacks, they might be collateral victims.

Sometimes, users of these less popular or temporary sites don't have Internet accounts that permit setting up servers. Onionsites can generally work with this limitation because they make only outbound client connections. Similarly, onionsites can be used to administer systems behind restrictive firewalls that permit only outbound connections. Even if users do have Internet accounts that permit them to provide Web services, their providers might not offer HTTPS, or offer it only at an additional fee.

With Tor's user base in the millions and growing, website owners might also want to ensure that their sites are accessible to Tor users. Sites such as Facebook use onion services to give Tor users better performance, security, and user experiences than what they receive when connected over a simple Tor circuit to facebook.com.[8] On the other hand, those with small personal sites might discover that their hosting provider blocks access from Tor exits. When product designer Glenn Sorrentino realized that this was true of his site, glennsorrentino.com, he set up a version on a small personal system at at3o24mj2rfabkca.onion. Doing so offered other benefits as well, but his motivation was reachability for Tor users. Note that because the Tor network is designed to be reached even by users experiencing censorship, another way to solve this problem could be to run the site as an onion service from the same Web server but connecting to Tor via bridges and obfuscating pluggable transports (www.torproject.org/docs/bridges).

We focus primarily on using onionsites to improve authentication, setting properties of network location hiding aside as orthogonal to our goals. However, these properties can be complementary for some use cases. Authenticated hidden services are an appealing option for those who'd like to secure their onionsites for personal use. Unlike with traditional websites, which are discoverable online before authentication, users lacking authentication information for private onionsites won't be able to determine easily whether they exist, nor will they be able to probe them for vulnerabilities. Configuring onionsites for obfuscation of site existence, and thus site vulnerability, is ideal for operating a personal cloud service. With privacy and cost in mind, many people operate their own cloud infrastructures to store files and calendar entries by using open source systems such as Cozy (https://cozy.io) and OwnCloud (https://owncloud.org). Authenticated hidden services are also often used as personal RSS readers, because onionsites ensure some level of feed integrity—particularly important when fetching news feeds that don't utilize TLS.

Users can, and often do, create Facebook or similar pages that are protected by HTTPS and TLS certificates. But then the service must depend on the host's reputation, trust, policies, and protections—not to mention dynamics—rather than let users understand and control these aspects of their own services.

A simple way to bind the onionsite public key to a known entity that uses widely available mechanisms is to provide a signature on the onion address, such as a PGP/GNU Privacy Guard (GPG) signature. The signed text can be included on the onionsite, making it self-authenticating in this sense as well. The trust level in the authentication is then equivalent to the trust in the public key doing the signing. Such techniques are already used for signing code. For example, the Tor Project offers signatures on all sources and binaries it makes available for download.

Signers can also post the signed onion address to a public site, such as their Facebook page. Indeed, a useful public site for doing this would be an unauthenticated version of the same service as the one the onionsite offers. The unauthenticated version and the onionsite version should contain signed pointers to each other so that anyone can check their association. For example, by posting his PGP signature at both http://glennsorrentino.com/onion-binding.php and http://at3o24mj2rfabkca.onion/onion-binding.php, Sorrentino binds the addresses of his site's unauthenticated and authenticated versions.

Another potential place to post the association is Keybase (https://keybase.io), a "people directory" in beta release. Keybase lets you look up by username GitHub, Reddit, Twitter, and Bitcoin identifiers signed with the same PGP key. Incidentally, Keybase has an onion address (http://fncuwbiisyh6ak3i.onion) for its registered domain address.

Given onionsites' authentication benefits, why bother with a non–onionsite version? Providing a site at the registered domain makes it available to users not coming over Tor. Typically, an onionsite can still be accessed via Tor2web (https://tor2web.org), a website that proxies connections from non–Tor clients to onionsites. Such proxying services might provide broader availability; however, at best, they offer overtly acknowledged MITM onionsite connections. Because we're focused on not merely maintaining but improving authentication, we'll say no more about such proxies and limit our discussion to secure onionsite access for current and future Tor users. Site operators wanting to provide wider, if less secure, access should do so by connecting to the registered domain name, which is hopefully at least protected by HTTPS.

Finally, Google and other traditional search and indexing engines don't generally reflect links to onionsites, unless onionsites associated with registered domains are included in the sites' metadata, as in our glennsorrentino.com example. The Ahmia search engine (https://ahmia.fi) is limited to onionsites and thus likely to be known only to those already familiar with them. However, its creator, Juha Nurmi, has agreed to link onion and registered domain addresses in Ahmia, together with the GPG signatures that bind the linking. He's also suggested to us that Ahmia could automatically test the signatures and check the registered-domain and onion sites. Thus, even if they aren't comfortable performing PGP verification, users who trust Ahmia (and their connection to Ahmia) can verify that the same party operates a pair of websites. Onionsite crawling and indexing are in their infancy and thus aren't as representative of their target space as Google's and similar sites' much more mature indexing of the surface Web.

Usability, Convenience, and Security

Because most onionsite visitors use Tor Browser, deployment and debugging of onion services are faster than for their registered domain counterparts—there's only one browser to test, with only minor user variation. Website operators can assume that users don't have AdBlock or other browser extensions that affect how content is displayed. Plug-ins such as Java and Flash that might mitigate Tor Browser's privacy protections are disabled by default. Many privacy-conscious users enable the NoScript extension to block JavaScript as well. Despite this, rich content such as video, audio, and interactive storytelling are still available for designers willing to use HTML5 and CSS3. And because Tor Browser generally restricts what it will process more than other browsers do, operators wanting to offer access to their Tor Browser–tested site at a registered domain shouldn't have to make any changes.

What we've described so far implies relatively manual PGP/GPG signature authentication. It would be straightforward to create a plug-in that verifies the signature and the trust in it, then gives users different indications depending on the results. Related tools have already been developed; for example, Monkeysphere (http://web.monkeysphere.info) is a Firefox plug-in that uses the PGP trust infrastructure for validation only when the browser doesn't accept the TLS certificate validation by default. A simpler plug-in could also check the Ahmia validation suggested earlier.

“With Tor's user base in the millions and growing, website owners might also want to ensure that their sites are accessible to Tor users.”

Website operators can now use our PGP approach (at least manually). Although our approach could benefit from usability developments and simplification, it can complement other approaches, as it doesn't rely fundamentally on the deployment and continued commitment to new infrastructure. Instead, it can rely on whatever authentication infrastructure is popular and likely to be maintained for independent reasons.

The PGP web of trust builds up signature authority in a decentralized manner from direct personal connections and introductions. This fits more naturally with, for example, community, local business, personal, and collaborative work sites, for which local or personal trust relationships are important.[9] By contrast, the X.509 trust model is a hierarchical centralized trust chain delegated down from a national or global corporate trust anchor.

PGP remains much less familiar than TLS. Popular familiarity is, however, not so much with TLS as with interfaces such as the lock icon in the browser search bar. This indicates little more than whether TLS and certificates from default-accepted authorities are in operation. However, most users lack even this basic understanding: to them it means “secure.” It's up to us to design systems so that such simple judgments are correct and users will naturally do the right thing. As noted, similar PGP interfaces have been designed but haven't been developed extensively the way TLS interfaces have—unsurprising given TLS's fundamental role in global e-commerce. For those who don't otherwise rely on the PGP web of trust's social or local protections, TLS certificates will likely remain the primary ground for linking public, human-readable domain names to signatures that authenticate websites.

Let's Authenticate

Again, unlike conventional Web URLs, onion addresses are connected inextricably to the site authentication key. Thus, if you've publicized the onion address on, for example, blogs, Twitter, or Facebook, people following those address links won't be vulnerable to hijacks or MITM attacks by a subverted CA. This significantly raises the bar on the hijacker. Furthermore, non-CA-based MITM techniques, such as forcing the site to fall back to a non-SSL version (for example, by using SSLStrip) or to use a weaker cipher to communicate (for example, via BEAST or FREAK), won't be possible because, unlike for conventional Web addresses, the onion address and key are linked inextricably and generated cryptographically.

Given the success of Let's Encrypt, we envision eventual incorporation of TLS with onionsites for the “everyman” users we described. Whereas certificate transparency and the like will help increase trust in authenticating such sites via their certificates, onion addresses' self-authentication adds to this trust in two ways. They strengthen the certificate-based authentication that certificate transparency addresses, and the use of onion routing implies authentication of the route, not just the destination. And both of these are under more direct owner control. But, it's not just for the little guys. The US General Services Administration—which negotiates federal-friendly terms of service (ToS) for the US government[10]—has negotiated an amendment to the Let's Encrypt Subscriber Agreement for US government users. And Let's Encrypt already has significant US government adoption (https://crt.sh/?Identity=%25.gov&iCAID=7395).

Creating the Domain Validation Certificate

We assume that the certificate to be obtained will have the onion address listed as a SAN (subjectAltName) in the certificate issued for the registered domain name. Currently, CA/Browser Forum policy allows only registered domain names and wildcards thereof, such as *.duckduckgo.com. The only exception is for EV certificates, which are prohibitive for many site owners and, hence, problematic. Nonetheless, in response to numerous requests, DigiCert now provides instructions for ordering .onion certificates.[11] We'll explore some concerns and reasons why the approach set out in this section supports changing current restrictions. But, first, we describe how this approach would work if onion addresses were allowed as names in DV certificates.

You could simply create a self-signed certificate binding the onion and registered-domain names. But then a popup would warn users because the browser won't trust you to be a signing authority. Such warnings are important because most people use Tor precisely for safer connections to registered domain addresses. We're pursuing a strengthening of—not an alternative to—the current authority-based Web authentication infrastructure, to which user experience is central. Thus, we want to avoid both accepting self-signed certificates without warning and adding to circumstances in which popup warnings occur superfluously.

Onion addresses should receive at least the same DV level of checking as occurs now for registered domain names. The latest ballot-approved CA/Browser Forum's baseline requirements list several ways to demonstrate domain control.[12] The most familiar is probably responding to an email sent to administrator@[registered domain] or a similar address. The baseline requirements also let certificate applicants demonstrate their ability to make requested changes, such as adding a nonce to a page whose name terminates in the requested domain name. So, a validation query protocol can be used that freshly connects to the onionsite and asks whether it's acceptable to certify association of the onionsite with the registered domain. This can also verify that the onionsite is configured properly. The CA should issue the certificate only if all DV checks are completed successfully.

An email or other check of the registered domain must also include the onion name. If applicants could obtain a certificate for multiple registered domain names by showing control of only one, they could fraudulently authenticate other sites covered by the certificate. Onion addresses' self-authentication limits this risk. This check alone wouldn't prevent people from obtaining certificates for onion addresses not under their control. But, because they wouldn't possess the onion address's private key, people tricked into going to that address simply wouldn't connect successfully. Nonetheless, many subtle authentication attacks are possible when users are confused about who they're connecting to and in what role, especially if authentication protocol runs are interleaved.[13] Therefore, we recommend that the certificate-issuing protocol include a check that whoever controls the onion address authorizes its binding to the registered domain name.

Connecting to Onionsites

Assuming an onionsite has been configured and certified, how should users connect to it? If users request a connection to the onion address by, for example, clicking a link, then the connection should proceed as normal. But if users request a connection to the associated registered domain address, they could be redirected automatically to the onionsite as a security enhancement. Additions to the HTTPS Everywhere (www.eff.org/https-everywhere) ruleset could accomplish this.

HTTPS Everywhere—a browser extension incorporated by default in Tor Browser and available for Firefox, Chrome, and Opera—rewrites requests to connect to sites via unencrypted HTTP to HTTPS requests. This does more than add an "S" to the request. Sometimes a site's encrypted and unencrypted versions are in different domain locations. Conversely, adding an "S" to an HTTP request might connect users to a page that the domain owner intended for purposes other than a heightened-security version of the HTTP site. Like HTTP Strict Transport Security (HSTS), HTTPS Everywhere helps guard against SSLStrip and similar attacks. HTTPS Everywhere also includes the SSL Observatory. Note that the ruleset could also be expanded to allow redirection to onionsites using the GPG binding approach we described earlier.

Another advantage of using HTTPS Everywhere to direct registered domain requests to onionsites is that DNS lookup of an IP address won't be associated with the domain name. This means that such connections won't be affected by attacks on DNS resolution or by observations of DNS lookups exiting the Tor network.

An Onion by Any Other Name Would Cert as Sweet

So, why not just permit onion addresses to be used as names in certificates? CA/Browser Forum discussions have raised two broad classes of objections.

First, currently deployed onion addresses and protocols rely on SHA-1 and RSA-1024, both of which have reached the end of their effective-security lifetimes. But Tor client and relay software has transitioned in stable releases to SHA-256 and Ed25519, which are adequate for the foreseeable future. And Tor is expected to transition onion services to these cryptographic primitives within the year. Therefore, any valid objections based on this concern will be short-lived. More important, when combined, onion protections can only add to TLS and certificate protections. Breaking the private RSA-1024 key associated with an onion address that has an appropriately stronger TLS key and certificate doesn't, by itself, allow an attacker to subvert a certified TLS session with the onionsite. Conversely, MITM, cipher degradation, or other certificate or TLS instance attacks aren't possible with onion addresses unless the attacker also breaks the self-authentication.

Second, for various reasons, some individuals support a CA's ability to link real-world identities to issued certificates, as occurs when validating registered domain names. This is why only EV certificates have been approved for onion addresses. But the described design proposes that a DV certificate for an onion address be issued only when it's fully bound to a registered domain name and validated by the same process as for the registered domain name. Whatever benefits such linking provides is supported as strongly for the onion address as for the registered domain name alone.

A decade ago, websites available via encrypted and authenticated connections were relatively rare. Providing users with such options seemed the province of the paranoid rather than standard good practice. Whether or not our specific design recommendations are adopted, we hope that in our general approach, readers recognize prospective changes, which onionsites facilitate, that are as important to the future of secure and robust access to and use of the Internet as certificates and TLS were at the turn of the century. We also hope our expanded view of Tor's onion services will encourage others to explore this fascinating system for novel properties and applications. ■

Acknowledgments

We thank the anonymous reviewers for their feedback and suggestions. We have also benefited from conversations with many people, including Richard Barnes, Roger Dingledine, Peter Eckersley, Eric Mill, Alec Muffett, Mike Perry, Seth Schoen, and Ryan Sleevi.

References

1. G. Kadianakis and K. Loesing, *Extrapolating Network Totals from Hidden-Service Statistics*, Tor tech. report 2015-01-001, Tor Project, 31 Jan. 2015; https://research.torproject.org/techreports/extrapolating-hidserv-stats-2015-01-31.pdf.
2. R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," *Proc. 13th USENIX Security Symp.* (SSYM 04), Aug. 2004, p. 21.
3. D.M. Goldschlag, M.G. Reed, and P.F. Syverson, "Onion Routing for Anonymous and Private Internet Connections," *Comm. ACM*, vol. 42, no. 2, 1999, pp. 39–41.
4. J. Vickers, "OnioNS-server: The Onion Name System—Networking Protocols," GitHub, 28 Sept. 2015; https://github.com/Jesse-V/OnioNS-server.
5. J. Aas, "Let's Encrypt: Delivering SSL/TLS Everywhere," Let's Encrypt, 18 Nov. 2014; https://letsencrypt.org/2014/11/18/announcing-lets-encrypt.html.
6. L.-S. Huang et al., "Analyzing Forged SSL Certificates in the Wild," *Proc. IEEE Symp. Security and Privacy* (SP 14), May 2014, pp. 83–97.
7. J. Appelbaum and A. Muffett, "The '.onion' Special-Use Domain Name," Internet Engineering Task Force, Oct. 2015; https://tools.ietf.org/html/rfc7686.
8. A. Muffett, "RFC 7686 and All That ...," Facebook, 23 Oct. 2015; www.facebook.com/notes/alec-muffett/rfc-7686-and-all-that/10153809113970962.
9. P. Zimmerman, "Why OpenPGP's PKI Is Better than an X.509 PKI," OpenPGP Alliance, 27 Feb. 2001; www.openpgp.org/technical/whybetter.shtml.
10. "List of Negotiated Terms of Service Agreements," US General Services Administration, 2015; www.digitalgov.gov/resources/negotiated-terms-of-service-agreements.
11. "Ordering a .Onion Certificate from DigiCert," DigiCert, 15 Dec. 2015; https://blog.digicert.com/ordering-a-onion-certificate-from-digicert.
12. "CA/Browser Forum Baseline Requirements Certificate Policy for the Issuance and Management of Publicly-Trusted Certificates, Version 1.3.0," CA/Browser Forum, 16 Apr. 2015; https://cabforum.org/wp-content/uploads/CAB-Forum-BR-1.3.0.pdf.
13. P. Syverson and I. Cervesato, "The Logic of Authentication Protocols," *Proc. Int'l School on Foundations of Security Analysis and Design on Foundations of Security Analysis and Design* (FOSAD 00), LNCS 2171, 2001, pp. 63–136.

Paul Syverson is a mathematician at the US Naval Research Laboratory, Center for High Assurance Computer Systems. His research interests include computer and communications security and privacy with an emphasis on theory, design, and analysis of traffic-secure systems, especially onion routing. Syverson received an MA and PhD in philosophy and an MA in mathematics from Indiana University. He's an Electronic Frontier Foundation Pioneer, Foreign Policy Global Thinker, and ACM Fellow. Contact him at paul.syverson@nrl.navy.mil.

Griffin Boyce is a fellow at the Berkman Center for Internet & Society at Harvard University and a senior censorship researcher for the Open Internet Tools Project. He works on various anticensorship projects, including Satori and Cupcake Bridge. Contact him at griffin@cryptolab.net.