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

Paul Syverson
U.S. Naval Research Laboratory
paul.syverson@nrl.navy.mil

Griffin Boyce
Berkman Center for Internet and Society at Harvard University
griffin@cryptolab.net

***Abstract*—Tor is a communications infrastructure widely used for unfettered and anonymous access to Internet websites. Tor is also used to access sites on the .onion virtual domain. The focus of .onion use and discussion has traditionally been on the offering of hidden services, services that separate their reachability from the identification of their IP addresses. We argue that Tor's .onion system can be used to provide an entirely separate benefit: basic website authentication. We also argue that not only can onionsites provide website authentication, but doing so can be easier, faster, cheaper, and more flexible and secure than existing alternatives. We illustrate this with two approaches: a general but manual technique based on PGP that can be used right now, and an automated approach that can be integrated with traditional TLS certificates.**

I. Introduction

Tor is a widely popular communications infrastructure for anonymous communication. Millions use its thousands of relays for unfettered traffic-secure access to the Internet. The vast majority of Tor traffic by bandwidth (over 96% at our last check) is on circuits connecting Tor clients to servers that are otherwise accessible [1]. Tor also provides protocols for connecting to services on the pseudo-top-level domain .onion, which are only accessible via Tor. In this paper we explore using Tor's .onion infrastructure so that individuals operating a website can create authentication, integrity and other guarantees more simply, easily, fully, cheaply, and flexibly than by other currently available means.

Tor's onionsites have been advocated since their introduction as a way to protect network location information for servers not just clients [2]. (Such advocacy actually predates their introduction inasmuch as the same was said for web servers contacted by reply onions [3], See also [4] for description of an implemented predecessor to Tor's hidden services.) Discussion in the popular press, as well as research to date, has focused almost exclusively on location hiding and associated properties provided by onionsites and the protocols to interact with them. Indeed, these are generally referred to collectively as *Tor Hidden Services* in the research literature and as the *Dark Web* in the popular press. (Although so many importantly distinct things are often subsumed and run together under 'Dark Web' as to rob the term of clear significance, other than as a caution flag for the hidden incoherence that surrounds most occasions of its use.)

Our intent is to challenge the narrowness of this view of onionsites. In particular we will discuss security protections they readily facilitate that are largely orthogonal to hiding server location. We hope by the end of this paper the reader will agree that they should be called *onion services* or in any case something that is more properly inclusive of the variety of security properties they offer.

II. Brief background on Tor and onion services

We sketch out minimal basics of Tor onion services. For more detailed descriptions see the Tor design paper [2] and related documentation at the Tor website [5]. For a high-level graphical description of onion services see [6]. For a more up to date, and much more technical, description of onion services protocols see the Tor Rendezvous Specification [7].

Tor clients randomly select three of the roughly 6500 relays [8] comprising the current Tor network, and create a cryptographic circuit through these to connect to Internet services. Since only the first relay in the circuit sees the IP address of the client and only the last (exit) relay sees the IP address of the destination, this technique separates identification from routing. To offer an onion service, a web (or other) server creates Tor circuits to multiple *Introduction Points* that await connection attempts from clients. A user wishing to connect to a particular onion service uses the onion address to look up these Introduction Points in a directory system. In a successful interaction, the client and onionsite then both create Tor circuits to a client-selected *Rendezvous Point*. The Rendezvous Point mates their circuits together, and they can then interact as ordinary client and server of a web connection over this rendezvous circuit.

Since the onionsite only communicates over Tor circuits it creates, this protocol hides its network location, the feature that gives it the name 'hidden service'. But, there are other important features to the .onion system, notably self-authentication. The onion address is actually the hash of the public key of the onionsite. For example, if one wishes to connect to the DuckDuckGo search engine's onion service, the address is 3g2upl4pq6kufc4m.onion. The Tor client recognizes this as an onion address and thus knows to use the above protocol rather than attempting to pass the address through a Tor circuit for DNS resolution at the exit. The public key corresponds to the key that signs the list of Introduction Points and other service descriptor information provided by the directory system. In this way, onion addresses are self-authenticating.

Hold the extra onions, Make mine a single

For services such as DuckDuckGo, the value provided by onion services lie, not in their location hiding, but in this additional authentication and in the greater assurance of improved route security offered to their users by requiring a connection

via Tor. The complexity, latency, and network overhead of the several Tor circuits needed to reach Introduction Points and Rendezvous Points is not needed to provide those protections.

Currently deployed onion services are essentially as described above. They ultimately provide a *double-onion service* that joins two Tor circuits: one protecting the client and one protecting the server. When server location is not a concern, a *single-onion service* that uses just one Tor circuit from the client may adequate. And a single-onion service brings advantages including reduced connection and transmission latency, as well as reduced network overhead. For these reasons there is a current draft Tor Proposal (the Tor equivalent of IETF RFCs), for just such a single-onion service design [9]. Server location protection is not the only use of double-onion services. Many people administering systems behind restrictive firewalls that only permit outbound connections currently use onion services to administer their systems. There are, however, settings where using a single-onion service is advantageous.

III. Knowing to which self to be true

Of course the self-authentication described above for DuckDuckGo only binds the service descriptor information to the 3g2upl4pq6kufc4m.onion address. What a user would like to be assured of is that s/he is reaching DuckDuckGo. Presumably the user wants the search results DuckDuckGo offers and not what might be returned by some other, possibly malicious, server. In addition to the integrity guarantee, the user relies on authentication so that queries are revealed only to DuckDuckGo and not to others. The onion address by itself does not offer this. Making use of the traditional web trust infrastructure, Facebook offers a certificate for its onion addresses issued by DigiCert. This helps ensure that users are not misled by onionsites purporting to be official.

Though cryptographic binding is essential to the technical mechanisms of trust, users also rely on human-readable familiarity, for example, that the browser graphically indicates s/he has made a certified encrypted connection as a result of typing facebook.com into the browser. To some extent, it is at least possible to make use of this in .onion space. By generating many keys whose hash had 'facebook' as initial string and then looking among the full hashes for an adequately felicitous result, Facebook was able to obtain facebookcorewwwi.onion for its address. Whatever its value for Facebook, this is clearly not something that will work widely, as it is difficult to generate custom addresses in this way.

The Onion Name System (OnioNS) attempts to respond to these concerns by creating a system for globally-unique but still human-meaningful names for onionsites [10]. This has the advantage of not being dependent on any existing naming scheme, such as existing domain registration. On the other hand, through much experience and design, existing approaches to naming have evolved effective usage and infrastructure that we can leverage. We will focus herein on approaches that link onion addresses to already meaningful ways of referring to sites. We focus primarily on the case where one controls a registered domain name, but we discuss binding to other meaningful web locations even when this is not available.

If one has 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. These are not original observations. Indeed that description is actually a quote from the blog of Let's Encrypt, a new certificate authority dedicated, among other things, to making TLS certification free and automatic [11].

Using the existing X.509 system, setting up a certificate can take hours or even days. In cases where the website is operated by a collective or organization, SSL/TLS certificates have been known to take months, due to questions around ownership and authorization. This time cost is in addition to the monetary cost of the certificate, 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 and start Tor. The Tor Project also provides a brief page with additional tips and advanced options [12]. Even if one follows the option described below of PGP for the binding and the process of learning how to create a PGP key and signature is taken into account, the time investment is dramatically less than with the current X.509 public-key infrastructure.

As of this writing, Let's Encrypt services are still a few months away. Should they be willing to offer certificates for onion domains, using Let's Encrypt could be an easy way for onionsite operators to take advantage of the traditional certification infrastructure. We will return to this prospect below.

Traditional SSL certificates are not without problems, even ignoring the above cost and convenience questions. The nature of 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 cases of 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 some of the most popular websites [13].

EFF's SSL Observatory [14] monitors for such problems and documents their occurrence. Google's Certificate Transparency Effort [15] is similar but broader, adding (amongst other things) append-only signed public logs that make certificate shenanigans all the harder to bring off undetectably.

Rather than simply monitor and flag certificate authority problems, the Perspectives Project [16] strives to provide end users with control over the trust they place in website certificates. Instead of trusting anointed CAs, semi-trusted network notaries probe network services and build a record of public keys those services have used over time, somewhat similar to the approach of certificate transparency. Users can choose which notaries they wish to trust, and clients encountering unfamiliar public keys will query notaries for a history of keys used by a service [17]. This is especially intended to enhance trust on first use (tofu) authentication, although it can also supplement traditional CA based PKI security.

The problems with certificates raised above and below, though real, are largely moot at the moment for onion services. As of this writing, the CA/Browser Forum has approved only EV (Extended Validation) Certificates for .onion addresses.

This limits their use to those with the significant amount of time, money, and desire to follow the extensive identity validation process required. EV Certificates are primarily used by large businesses [18]. More common for individuals, organizations, and small business that obtain certificates for their sites are Domain Validation (DV) Certs. These typically require a simple email confirmation based on information in the WHOIS database. In addition, the CA/Browser Forum approval for EV Certs for .onion addresses is valid only until November 1, 2015 [19].

The future prospects for .onion address certificates are not as hopeless as the above might make it seem. An IETF RFC for registering .onion addresses as special use domain names [20] has recently been approved by the Internet Engineering Steering Group. With the status of .onion addresses appropriately resolved, this should clear the way for the CA/Browser Forum to approve eligibility of .onion addresses for SSL Certificates by November 1.

IV. Our onions ourselves

As noted, onionsites already provide a self-authenticated binding of public key to onion address but do not bind that public key to something recognizably associated with that site. We seek a solution that will work for all kinds of sites, but in this section we are especially interested in providing authentication for only moderately popular and/or short-lived websites, e.g., personal web pages, hometown sports teams, sites for local one-time events, small businesses, municipal election campaigns, etc. Though not such large targets as more popular long-lived sites, they are still subject to controversy and have been subject to many of the same sorts of attacks as more well-known sites. They might also not be the target of attacks but simply collateral victims.

Some users of this kind may not even have Internet accounts that allow them to set up servers. Onionsites are compliant with such a limitation since they actually only make outbound client connections. As noted above, a related existing usage is for administering systems behind restrictive firewalls that only permit outbound connections. Even if the user has an Internet account that permits setting up a web page, HTTPS may not be available from that provider or only available for an additional fee.

We are primarily focused on improvements to authentication using onionsites and thus mostly leave properties of network location hiding aside as orthogonal to our goals. They can be complementary, however. Even for hidden service applications, it might still be desired to connect the onionsite to some pseudonymous reputation. Authenticated hidden services are also an appealing option for those who'd like to secure their onionsites for personal use. Unlike traditional websites that appear online prior to authentication, users lacking authentication information for such a site will not be easily able to determine that it even exists, nor will they be able to probe it for vulnerabilities. These qualities make an ideal environment for operating a personal cloud service. With privacy and cost in mind, many people are operating their own cloud infrastructure to store files and calendar entries using open-source systems such as Cozy and OwnCloud [21]. Another common use of authenticated hidden services is as a personal RSS reader, as onionsites ensure some level of feed integrity (particularly important when fetching news feeds that do not utilize TLS).

Of course one can always create a Facebook page or something similar that is protected by HTTPS and TLS certificates, and this is often done. But this makes the service dependent on the reputation, trust, policies, and protections of such a host, not to mention the dynamics thereof, rather than allowing the user to readily understand and control these aspects of his own service. Also, Facebook policy requires identification of the person providing the site, while we would prefer to leave this as simply a separate issue.

A very simple way to add binding of the onionsite public key to a known entity using widely available mechanisms is to provide a signature on the onion address. We envision a PGP/GPG signature, but it could be an X.509 signature (or other as we discuss below). The signed text can simply be included on the onionsite, making it self-authenticating in this sense as well. The trust in the authentication will then be whatever trust is associated with the public key that does the signing. Such techniques are already used for signing code. For example, the Tor Project offers signatures on all source and binaries it makes available for download.

If the signer wishes to post the signed onion address to a public site such as her Facebook page, she can do this also. (An advantage of doing so will be discussed below.) Indeed, a useful public site for doing this would be an unauthenticated version of the same exact service as the one being offered at the onionsite. The unauthenticated version and the onionsite version should both contain a signed pointer to both versions. It is then easy for anyone who desires to check their association. For example, we have made an authenticated version of http://cryptic.be/about.html available at http://swfpqf5iga4stake.onion/about.html. A natural place to also post the association would be to Keybase, which is in beta and a somewhat similarly motivated "people directory" [22]. Keybase lets one look up via usernames on github, reddit, twitter, bitcoin etc. identifiers signed with the same PGP key. Keybase is, incidentally, another commercial site with an onion address (http://fncuwbiisyh6ak3i.onion/) for its registered-domain address (https://keybase.io/), although at last check they have not recursively listed this itself within Keybase.

Why even bother with the non-onionsite version? There are several reasons. First, this allows for a binding of the public domain name to the onionsite. As mentioned, onion addresses are inherently not humanly meaningful, which can lead to confusion among end-users. To obtain the entirety of a specific domain name of choice is also technologically infeasible, as onion addresses are randomly-generated alphanumeric strings (currently of 16 digits). Some other approach is needed. The signed-onion technique allows someone to choose and retain a desired domain name for the site, while still being able to offer an authenticated and integrity protected version easily. This also illustrates one of the benefits of using GPG or similar signatures. If the authentication simply showed that the same party that provided the not-secured site provided the onionsite, an attacker could set up an altered version, employ usual techniques to hijack the not-secure site, and offer a self-authenticated onionsite that matched the hijacked site. To do this undetectably against the GPG-signed onionsite would

require subversion of the trust in the GPG key. A concern with using GPG signatures is that users may not be familiar with them, have appropriate trust in the key, or bother to check the signature and the trust in the key. We will address these below.

In addition, many intended users of the site may not have Tor installed. Though installation is a simple point-and-click download, many may be disinclined against even this small effort. The onionsite would still be available via Tor2web, a website that proxies connections from non-Tor clients to onionsites [23]. To connect to an onionsite, one enters a URL such as the following for reaching DuckDuckGo's onionsite via Tor2web: https://3g2upl4pq6kufc4m.tor2web.org/. The Tor2web site explicitly states that "Tor2web only protects publishers, not readers." This is because the client connects to Tor2web over a direct TLS connection rather than via Tor, as would be the case of someone connecting to 3g2upl4pq6kufc4m.onion via the Tor Browser. For our purposes, authentication of the onionsite in this case is limited to the trust in authentication of this TLS connection (and trust in Tor2web itself) regardless of the trust in the GPG signature. Thus, no significant usability limitation arises from using other browsers, but security is significantly downgraded for the reasons already mentioned and by various MitM possibilities raised below. In section VI, we will describe how to remove this authentication trust dependency on Tor2web.

Finally, traditional search and indexing engines such as Google do not generally reflect links to onionsites. The search engine ahmia.fi [24] is limited to onionsites and known primarily to those familiar with them. Ahmia creator Juha Nurmi has, however, agreed to incorporate linking of onion and clearnet addresses into Ahmia, together with the GPG signatures binding that linking. He has also suggested to us that Ahmia could automatically test the signatures and check the clearnet and onion sites. Thus, a user who trusts Ahmia (and her connection to Ahmia) on this can verify that a pair of websites is operated by the same party, even if personally inexpert with manual PGP verification. Crawling and indexing of onionsites is also in its infancy and can thus not be expected to be as appropriately representative as the much more mature indexing of the surface web by Google and similar sites.

V. Usability, Convenience, and Security

As most onionsite visitors use the Tor Browser, deployment and debugging of hidden services can be faster than their clearnet counterparts because there is only one browser to test, with only minor variation across users. Website operators can assume that users do not have AdBlock or other browser extensions that may impact how content is displayed. However, plugins that may mitigate Tor Browser's privacy protections, such as Java and Flash, have been disabled by default. Many privacy-conscious users do 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.

What we have described so far implies a relatively manual authentication of PGP/GPG signatures. It would be natural and straightforward to create a plugin that verifies the signature and provides different indications to the user depending on the trust in it. There are already related tools, e.g., Monkeysphere, a Firefox plugin that uses the PGP trust infrastructure for validation only when the browser does not default accept the TLS certificate validation [25]. A simpler plugin could also just check the planned Ahmia validation mentioned above.

Our PGP approach naturally complements Perspectives and similar endeavors. Perspectives offers an improvement against certificate based MitM attacks. But if a site is newly available, the onionsite can still be trusted to be bound to the owner of the PGP signature. A new self-signed certificate, on the other hand, will have little or no Perspectives history, and users are reduced to a tofu decision. Also, Perspectives notaries largely function as detectors of certificate misbehavior over time. This is useful but cannot detect static misuse of certificates. For example, consider a typo-squatting site that uses a self-signed certificate to pass through connections to the site on which it is squatting, but does not misbehave or alter its key. Perspectives will not reflect anything wrong with such a site, whereas our approach will presumably not give the site a high degree of trust unless the squatter has that trust exogenously.

Our PGP approach also can be used (at least in manual form) right now by website operators. It would benefit from usability developments and simplification, and it can complement other approaches. It does not, however, rely fundamentally on the deployment and continued commitment to new infrastructure that is specific to it. It can instead rely on whatever authentication infrastructure might be popular and likely to be maintained for independent reasons, rather than needing to grow and maintain interest in its approach.

Convergence takes a similar view and *is* automated and deployed (but currently is designed for existing TLS signatures). Extending Perspectives, Convergence notaries can use additional strategies beyond network perspective to create trust [26]. For example, in addition to perspective notaries, one can choose to place some trust in traditional CAs, or DNSSEC authorities, or anyone you like. Also, instead of trusting an authority or class of authorities essentially forever as is the case for existing approaches, Convergence more straightforwardly allows one to add or remove notaries. Convergence is available in Firefox plugins. It would be interesting to investigate a Convergence-like addition to the Tor Browser that works with onion addresses and PGP/GPG signatures.

In the PGP web of trust, signature authority is built up in a decentralized manner from direct personal connections and introductions. This more naturally fits with many of the kinds of websites that we have suggested could most benefit from our approach, for which local or personal trust relationships are important [27]. The X.509 trust model currently in general use to support TLS certificates is by contrast primarily a hierarchical centralized chain of trust delegated down from some ultimate national or global corporate trust anchor.

PGP and its successors also remain much less familiar than TLS. Although popular familiarity is not so much with TLS as with interfaces that tell the user little more than whether or not TLS is in operation at all, which is roughly as it should be for usable security. As noted, similar interfaces for PGP have been designed but have not yet received the extensive development of TLS interfaces, unsurprising given the fundamental role of TLS in global ecommerce. For those who do not otherwise rely on the social or local protections of PGP's web of trust

TLS certificates are likely to remain the primary ground of linking public, human-readable domain names to the signatures authenticating websites.

NetTrust attempts to leverage a PGP-style social web of trust structure for websites and to provide usable feedback in a browser toolbar [28]. It is, however, intended as a recommender system. Trust is thus based on behavioral reputation not simply judgment about authentication, which is our focus.

VI. Let's Authenticate

Unlike conventional web URLs, onion addresses are inextricably connected to the site authentication key. This means that if one has publicized the onion address, e.g., through blogs, Twitter, or Facebook, people following those address links will not be vulnerable to hijack or MitM by the subverted CA the way they would be by a link to a regular URL. This significantly raises the bar on the hijacker fairly automatically and easily. Further, non-CA-based MitM techniques such as forcing the site to fall back to a non-SSL version (e.g., SSLStrip) or to use a weaker cipher to communicate (e.g, BEAST and FREAK) are also not possible as the address and key are inextricably linked and generated using a strong cipher. And, if onionsite private keys were to sign not just the Introduction Points and other elements currently stored in the .onion directory system, but also the TLS certificate, onion keys would bind the TLS certificate to the site as well as vice versa.

Assuming Let's Encrypt is successful, we can also envisage incorporation of TLS with onionsites for even the "everyman" users described above. While certificate transparency and the like will help increase trust in authenticating such sites via their certificates, the self-authentication of onion addresses can also add to this trust, and again in a way more directly under website owner control. It is not just the little guy however. We have already noted some of the increasing number of prominent companies that offer onion services. Also, the General Services Administration negotiates federal-friendly Terms of Service amendments for the rest of the U.S. government [29]. As of this writing, it is negotiating a federal-friendly amendment for the current Let's Encrypt ToS agreement.

In addition to the PGP-based approach given above, we now sketch a description of incorporating onion addresses into certificates for registered domains. We also describe how to incorporate these into a relatively transparent and simple system for website access with improved security.

Creating the Domain Validation Certificate

We assume the cert to be obtained will have the onion address listed as a SAN (subjectAltName) in the certificate issued for the registered domain name. At least the same DV level of checking should occur as is currently done when issuing certificates registered domain names. The email check should include the onion name that is being bound as well as the registered domain name. (For simplicity, we assume a single onion address and a single registered domain name, although doing this for a small number of registered domains and a similarly small number of onion addresses might be made to work as well.)

If someone were to obtain a certificate for multiple registered domain names by showing control of only one, they could thereby fraudulently authenticate others covered by the certificate. The self-authentication of onion addresses limits this damage. This check alone would not prevent someone from obtaining certificates for onion addresses not under her control. But, since she would not possess the private key for the onion address, people thereby tricked into going to that address thinking it was authenticated by the cert or thereby associated with the registered domain name would simply experience a failure to successfully complete a connection. Nonetheless, many subtle attacks on authentication are possible when parties are confused about who they are connecting to and in what role, especially if authentication protocol runs are interleaved [30]. It is therefore advisable to have a similar check that someone with control over the onion address authorizes binding of the registered domain name to the onion address.

That raises a difficulty, which will also need to be overcome if DV certs are to be issued for onion addresses at all, whether or not they are associated with a registered domain name. It is not as reasonable to assume that an email infrastructure corresponding to the onion address exists. Indeed, we are expecting a process with an option to include an onion address when obtaining a certificate via Let's Encrypt. For most users, this will need to be accompanied by an explanation of what that means and a pointer to instructions for setting up an onionsite corresponding to their registered domain. So it is a given that they will not yet have an appropriately associated email address. And even for users who do have an existing onionsite, there is no reason to assume an appropriately associated email address. There should not be a WHOIS entry for the onion address, and the draft RFC mentioned above (on .onion as a special-use domain name) explicitly prohibits registering onion addresses. Instead a validation query protocol will be needed that simply connects to the onionsite and asks if it is acceptable to certify association of the onionsite with the registered domain. This can also serve as a check that an onionsite the user just set up is properly configured and thus the certification process can continue. Only if all DV checks complete successfully should the CA be willing to issue the Cert.

Connecting to an onionsite by the client

Assuming an onionsite has been configured and certificate issued for it, how should a client connect to the onionsite? If connection to the onion address has been requested, e.g., by the user clicking on a link to that address, then the connection should proceed as normal and the browser should display appropriately for a DV certified destination. But a client may request a connection to the registered-domain address associated in the certificate and be automatically redirected to the onionsite as a security enhancement. This could be done by additions to the HTTPS Everywhere ruleset.

HTTPS Everywhere is a browser extension incorporated by default in Tor Browser and currently available for Firefox, Chrome, and Opera [31]. It rewrites requests to visit sites via unencrypted HTTP to HTTPS requests. This does more than simply add an "S" to the request. Sometimes the encrypted version of a site and the unencrypted version are at different locations in the domain. Conversely, sometimes adding an "S"

to an HTTP request will succeed, but will connect to a page intended by the domain owner for an entirely different purpose than offering a secure version of the site at the unencrypted address. Like HSTS, HTTPS Everywhere also helps guard against SSLStrip and similar attacks. HTTPS Everywhere also includes the SSL Observatory mentioned above.

Another advantage of relying on HTTPS Everywhere is that there will be no DNS lookup of IP address associated with the domain name. This means that no attacks on DNS resolution or even observations of DNS lookups exiting the Tor network can affect such connections.

As mentioned above, by using Tor2web one need not rely on Tor Browser to access onionsites. A configuration choice at time of onionsite certification could add a detect-browser-and-use-tor2web-if-needed option to the default HTTPS Everywhere ruleset. Then users of other browsers that support HTTPS Everywhere could obtain the added authentication provided by an onionsite version of a registered domain. Whether or not to use rules that redirect to onionsites could, nonetheless, also be a simple general settings choice for the HTTPS Everywhere client as well.

Note that authentication even for a client using Tor2web is much stronger because of the combined use of self-authenticating onion addresses and binding to known domain names by a Let's Encrypt Certificate. A redirection to a different onion address by Tor2web that went unnoticed would permit a successful connection, but this should fail an authentication check of the certificate of the requested site.

An Onion of Trust

We had suggested above that people could bind their public sites to their onion addresses with a GPG key. We finish with a similar alternative that makes different trade-offs.

One motivation for using PGP keys is that there is an existing web of trust that can be leveraged, and existing mechanisms to do the verification. This would allow not just securing of human-meaningful addresses and connections but also certification by more human-meaningful trust relations than the usual CA. This is also something that requires no new software to be created in order to do right now. Indeed, there are registered-domain (non-onion) sites with PGP signed certs for such purpose right now, e.g., psg.com. Further, this allows things like binding a wordpress blog to an onion address as mentioned above, so people need not have a registered domain to have a human-meaningful address for a site for which they would like to offer a route-secure alternative.

As noted there is no simple automated significant infrastructure to support this, however. The Monkeysphere plugin does provide support but has not seen wide adoption. A primary reason is that PGP remains something of a geek tool. That is possibly changing slowly, but in any case a plugin like Monkeysphere assumes people have PGP keys and a trust network rather than facilitating creation of one. Keybase, which is just getting started, may help change this as well since it uses existing social network identities and ties these to each other via PGP keys.

If .onion keys could be themselves linked in a PGP-like web of trust, then this could be more directly relevant and meaningful to people operating on the web, especially those unfamiliar with PGP-like notions. This would of course require things like the check and indication of sufficient trust being either built into the browser or as a plugin. And it would similarly need an easy interface and easy to understand criteria for people deciding when to sign another's key, as well as mechanisms for that to all happen securely. This holds more hope for a successful widespread web of trust to complement the X.509 type trust hierarchy than the PGP approach of section IV would support. How to decide whether a site is trusted (depending what each of these trust mechanisms indicate) would also need to be worked out and might again be configurable with standard defaults.

VII. Conclusion

In this paper we have described how Tor's onion services can be used, not for the usual stated purpose of hiding server network location, but for website authentication. We have also argued that onion services offer users, a simple, effective, cheap and flexible means of authentication with security advantages not provided by existing approaches. We have described two approaches to this: One, based on PGP, is feasible for immediate albeit manual use and works generally to provide authentication for any human meaningful site identifier. The other will work for onionsites associated with registered domains and has the potential for simple setup and use in conjunction with the Let's Encrypt infrastructure. We also hope our expanded view of the possibilities created by Tor's onion services will encourage others to explore this fascinating system for other interesting properties and applications.

Acknowledgments. We thank the anonymous reviewers of an earlier version 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, and Seth Schoen.

References

[1] G. Kadianakis and K. Loesing, "Extrapolating network totals from hidden-service statistics," The Tor Project, Tor Tech Report 2015-01-001, January 2015.

[2] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," in *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[3] D. M. Goldschlag, M. G. Reed, and P. F. Syverson, "Onion Routing for Anonymous and Private Internet Connections," *Communications of the ACM*, vol. 42, no. 2, pp. 39–41, February 1999.

[4] I. Goldberg and D. Wagner, "TAZ Servers and the Rewebber Network: Enabling Anonymous Publishing on the World Wide Web," *First Monday*, vol. 3, no. 4, April 1998.

[5] "The Tor Project," https://www.torproject.org/.

[6] "Tor: Hidden Services Protocol," https://www.torproject.org/docs/hidden-services.html.en.

[7] "Tor Rendezvous Specification," https://gitweb.torproject.org/torspec.git/tree/rend-spec.txt.

[8] "Tor network size," https://metrics.torproject.org/networksize.html.

[9] J. Brooks, P. Syverson, and R. Dingledine, "Single onion services, (Tor proposal 252," 2015. [Online]. Available: https://gitweb.torproject.org/torspec.git/tree/proposals/252-single-onion.txt

[10] J. Vickers, "OnioNS – the onion name system," https://github.com/Jesse-V/OnioNS-server.

[11] "Let's Encrypt: Delivering SSL/TLS Everywhere," https://letsencrypt.org/2014/11/18/announcing-lets-encrypt.html, November 2014.

[12] "Configuring Hidden Services for Tor," https://www.torproject.org/docs/tor-hidden-service.html.en.

[13] L.-S. Huang, A. Rice, E. Ellingsen, and C. Jackson, "Analyzing Forged SSL Certificates in the Wild," in *IEEE Symposium on Security and Privacy (SP), 2014*, May 2014, pp. 83–97.
[14] "The EFF SSL Observatory," https://www.eff.org/observatory.
[15] "Certificate Transparency," http://www.certificate-transparency.org/.
[16] "Perspectives Project," https://www.perspectives-project.org/.
[17] D. Wendlandt, D. G. Andersen, and A. Perrig, "Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing," in *USENIX Annual Technical Conference*, 2008, pp. 321–334.
[18] "Extended validation certificate," https://en.wikipedia.org/wiki/Extended_Validation_Certificate.
[19] "Ballot 144 – validation rules for .onion names," https://cabforum.org/2015/02/18/ballot-144-validation-rules-dot-onion-names/, Feb. 2015.
[20] J. Appelbaum and A. Muffett, "The .onion special-use domain name," https://datatracker.ietf.org/doc/draft-ietf-dnsop-onion-tld/, 2015.
[21] "Cozy: a Personal Cloud You can Host, Hack and Delete," http://cozy.io/.
[22] "Keybase," https://keybase.io.
[23] "Tor2web: browse the anonymous internet," https://www.tor2web.org/.
[24] "Tor Hidden Service (.onion) search: Ahmia.fi," https://ahmia.fi/search/.
[25] "Monkeysphere," http://web.monkeysphere.info/.
[26] "Convergence," http://convergence.io/, 2011.
[27] P. Zimmerman, "Why OpenPGP's PKI is better than an X.509 PKI," http://www.openpgp.org/technical/whybetter.shtml, February 2001.
[28] P. Hariharan, F. Asgharpour, and L. J. Camp, "NetTrust - recommendation system for embedding trust in a virtual realm," in *Proceedings of the ACM Conference on Recommender Systems*. ACM Press, 2007.
[29] U.S. General Services Administration, "List of negotiated terms of service agreements," https://www.digitalgov.gov/resources/negotiated-terms-of-service-agreements/.
[30] P. Syverson and I. Cervesato, "The logic of authentication protocols," in *Foundations of Security Analysis and Design: Tutorial Lectures*, R. Focardi and R. Gorrieri, Eds. Springer-Verlag, LNCS 2171, 2001, pp. 63–136.
[31] HTTPS Everywhere, https://www.eff.org/https-everywhere.