namecoin

Self-Authenticating TLS Certificates for Tor Onion Services

Jeremy Rand
Lead Application Engineer, The Namecoin Project
https://www.namecoin.org/

PGP: 1D04 FB9D 50BF 2A8E 9F3E 58AD DC7E 7F8A E30E 73E6

Presented at GPN22

Content Note!

- This talk contains mentions of geopolitical Internet censorship pertaining to the war in Ukraine.
  - Both text on slides, and verbal elaboration.
  - Only a couple minutes of the talk, but please be aware of this.

What are Tor onion services?

- Host TCP services anonymously.
  - http://2gzyxa5ihm7nsggfxnu52rck2vv4rvmdlkiu3zzui5du4xyclen53wid.onion/
- Encrypted and authenticated by the Tor daemon.
  - Using an Ed25519 key (base32) in the domain name.
  - No trusted 3rd parties, it’s just crypto.

What is TLS?

- Transport Layer Security.
  - Encryption and authentication for TCP services.
  - It’s the “S” in “HTTPS”.
- Based on the 90’s era Netscape protocol SSL.
  - Occasionally people still incorrectly call it SSL.

Combining onion services and TLS?

- Onion services are anonymous.
- But TLS has better end-to-end security.
  - Onion service encryption is terminated at the Tor daemon.
  - TLS is terminated at the application.

Why does the endpoint matter?

- Tor daemon and application may be in different trust domains.
  - Whonix – Tor Browser runs in one VM; Tor daemon runs in another VM.
  - Web server could also be on Whonix, not just the client.
  - Client and server won’t know whether each other has such a threat model.

Why does the endpoint matter? (2)

- Tor daemon and application may have an insecure network connection to each other.
  - Running Tor daemon and an HTTPS server on different cloud infrastructure IP’s.
  - Using Tor Browser on a laptop with a Tor daemon on a WiFi router.
  - Again, client and server won’t know whether each other has such a threat model.

Compatibility

- Many applications expect TLS to be there.
- Web browsers show scary warnings without TLS.
  - And they restrict features too (e.g. webcam access).
- Giving them TLS is easier than patching them.
  - Tor Browser’s relevant patches are a maintenance nightmare.

The challenge

- TLS authenticates servers with certificates.
  - So we need a way to issue certificates for onion services.
- We *could* use TLS without authentication (self-signed certs)…
  - Vulnerable to man-in-the-middle (MITM) attacks.
  - Also would require patching apps.

Using public certificate authorities for onion services?

- Standard websites would normally get certificates from a public certificate authority via ACME.
  - The protocol that Let’s Encrypt uses.
- Q Misell has done amazing work getting ACME to work with onion services.

Public CA’s are Problematic

- Centralized, trusted 3rd parties.
- Routinely engage in censorship.
  - Sci-Hub has had their certs revoked (copyright lawsuit).
  - So have all websites located in Donetsk (OFAC sanctions).
  - So has a Russian media outlet aimed at American audiences (OFAC sanctions again).
  - Let’s Encrypt says they revoke (censor) about one website certificate per month due to OFAC compliance reasons.

Public CA’s for Onions can be a Liability

- To censor a cert, you don’t have to get a court order.
  - You don’t even need to get a formal government request.
- Let’s Encrypt is proactively censoring sites due to OFAC issues, without OFAC ever asking them to.
  - If even Let’s Encrypt (backed by EFF) is so scared about OFAC violations that they’re censoring without being asked to do so…
  - It would be nice to investigate complementary options that might be more censorship-resistant (like onion services themselves).

Other motivation: Certificate Transparency

- Some onions want it.
- Some really don’t want it.
  - Onion operators who don’t want CT need another option besides public CA’s.

Inspiration from DNSSEC

- In the DNS world, there are TLSA records.
- You look up a TLSA record for a domain name…
  - You get back a public key.
- This public key can be used as a domain-specific CA to authenticate TLS certs for that domain.

We don’t want to use DNS of course

- We’re also not using a blockchain, don’t worry.
- But hmm… put in a domain name, get back a public key?
  - How about… parse the .onion domain, extract the Ed25519 public key?
- You can make TLS CA’s that use Ed25519 keys.
  - Proof of concept: onion-x509 by Alexander Færøy.

Web browsers don’t do TLSA

- So at Namecoin we had to solve this problem a while back.
- Browser add-on?
  - WebExtensions are no good.
  - Mozilla doesn’t want malware WebExtensions to interfere with TLS.

But there’s another kind of browser add-on

- PKCS#11 modules.
- Well-known: it’s how smartcards and HSM’s are added to browsers.
- Less well-known: browsers like Firefox use PKCS#11 as a database query API for TLS certificates.

Example PKCS#11 modules

- Mozilla’s built-in root CA list
  - `libnssckbi.so` – it’s where Let’s Encrypt, Comodo, etc. are.
  - This is a PKCS#11 module!
- SQLite-based trust database in Firefox.
  - “Softoken” – it’s where CA’s are stored that you added in the Firefox UI.
  - This is also a PKCS#11 module!

Can we make our own PKCS#11 module for this?

- Namecoin already had a PKCS#11 module (ncp11) for TLSA lookups.
  - Works great: Namecoin websites’ TLS certs validate without issues.
  - I patched it to do .onion too.
  - For .onion domains, instead of doing a TLSA DNS lookup, it just extracts the Ed25519 pubkey from the domain name.

Ed25519 Support Issues

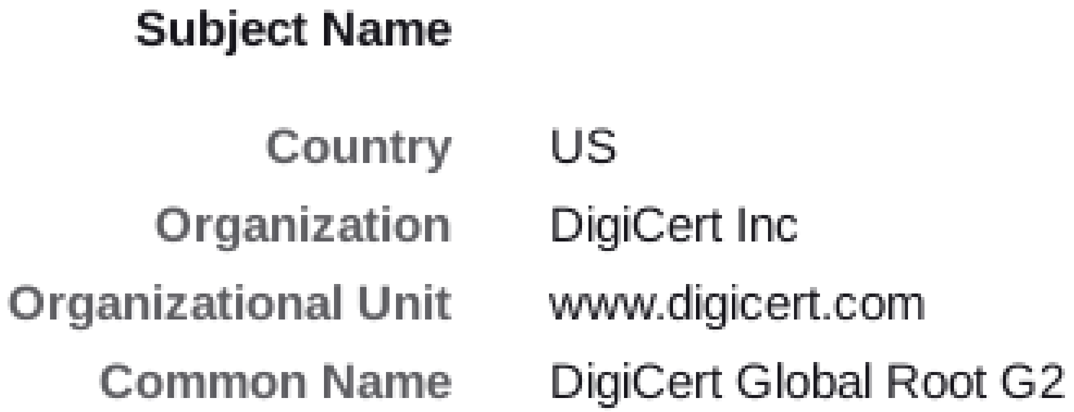
- TLS specs allow Ed25519 keys.
  - But Firefox doesn’t allow them.
  - It only allows NIST ECDSA keys.
- I found a workaround though.
  - It’s stupid but it works.

How do PKCS#11 queries work?

- Firefox sends PKCS#11 module the Subject Name of a CA certificate.
- PKCS#11 module responds with the full certificate of that CA and whether it’s trusted.

What’s a Subject Name?

- A Subject Name looks like this:



Subject Name	
Country	US
Organization	DigiCert Inc
Organizational Unit	www.digicert.com
Common Name	DigiCert Global Root G2

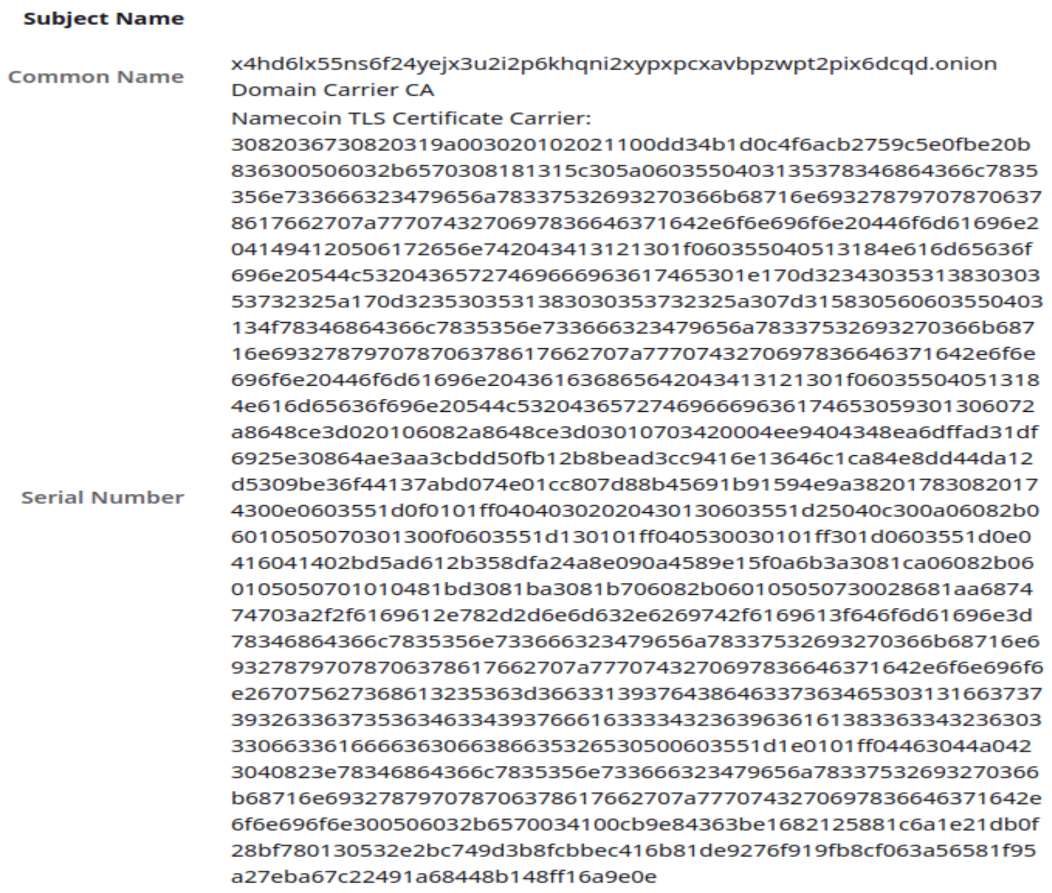
There's no public key in a Subject Name

- Our PKCS#11 module can return a CA certificate containing any public key we want.
- Our module just has to know which public key to return.
- If we wanted our PKCS#11 module to return a CA with an Ed25519 public key, it'd be easy: just embed the .onion domain in the Subject Name (which we'd do anyway so that we know what domain name to mark it as trusted for), and extract the Ed25519 public key from the .onion domain.
- But we want our module to return a CA with an ECDSA public key.
  - How do we know which ECDSA public key the onion service owner owns?

Subject Name Tricks

- Subject names can have a serial number.
  - Not the same as the certificate serial number.
- Subject Serial Numbers are arbitrary text with no length limit.
  - The onion service owner can embed an ECDSA public key, and an Ed25519 signature of that ECDSA public key, in the Subject Serial Number.

Subject Serial Number with Embedded Signature



Subject Name

Common Name	x4hd6lx55ns6f24yejx3u2i2p6khqni2xypxpcxavbpzwpt2pix6dcqd.onion
	Domain Carrier CA
Serial Number	Namecoin TLS Certificate Carrier: 3082036730820319a003020102021100dd34b1d0c4f6acb2759c5e0fbe20b

```
Namecoin TLS Certificate Carrier:
3082036730820319a003020102021100dd34b1d0c4f6acb2759c5e0fbe20b
836300506032b6570308181315c305a0603550403135378346864366c7835
356e733666323479656a78337532693270366b68716e69327879707870637
8617662707a7770743270697836646371642e6f6e696f6e20446f6d61696e2
041494120506172656e742043413121301f060355040513184e616d65636f
696e20544c53204365727469666963617465301e170d32343035313830303
53732325a170d3235303531383030353732325a307d315830560603550403
134f78346864366c7835356e733666323479656a78337532693270366b687
16e693278797078706378617662707a7770743270697836646371642e6f6e
696f6e20446f6d61696e204361636865642043413121301f06035504051318
4e616d65636f696e20544c532043657274696669636174653059301306072
a8648ce3d020106082a8648ce3d03010703420004ee9404348ea6dffad31df
6925e30864ae3aa3cbdd50fb12b8bead3cc9416e13646c1ca84e8dd44da12
d5309be36f44137abd074e01cc807d88b45691b91594e9a38201783082017
4300e0603551d0f0101ff04040302020430130603551d25040c300a06082b0
6010505070301300f0603551d130101ff040530030101ff301d0603551d0e0
416041402bd5ad612b358dfa24a8e090a4589e15f0a6b3a3081ca06082b06
0105050701010481bd3081ba3081b706082b060105050730028681aa6874
74703a2f2f6169612e782d2d6e6d632e6269742f6169613f646f6d61696e3d
78346864366c7835356e733666323479656a78337532693270366b68716e6
93278797078706378617662707a7770743270697836646371642e6f6e696f6
e267075627368613235363d36633139376438646337363465303131663737
3932633637353634633439376661633334323639636161383363343236303
3306633616666363066386635326530500603551d1e0101ff04463044a042
3040823e78346864366c7835356e733666323479656a78337532693270366
b68716e693278797078706378617662707a7770743270697836646371642e
6f6e696f6e300506032b6570034100cb9e84363be1682125881c6a1e21db0f
28bf780130532e2bc749d3b8fcbbec416b81de9276f919fb8cf063a56581f95
a27eba67c22491a68448b148ff16a9e0e
```

How does the PKCS#11 module handle this?

- PKCS#11 module sees a request from Firefox for a Subject Name containing a .onion domain, ECDSA public key, and Ed25519 signature.
- Verifies the signature with the .onion domain’s Ed25519 key.
- Constructs a certificate with the ECDSA public key.
- Returns that certificate to Firefox, marks as trusted for the given .onion domain.

From Firefox’s perspective…

- All Firefox sees is a CA with an ECDSA public key, and a really weird-looking Subject Serial Number.
- It sees that our PKCS#11 module marked it as a trusted CA for the onion service in question.
- Firefox is happy.

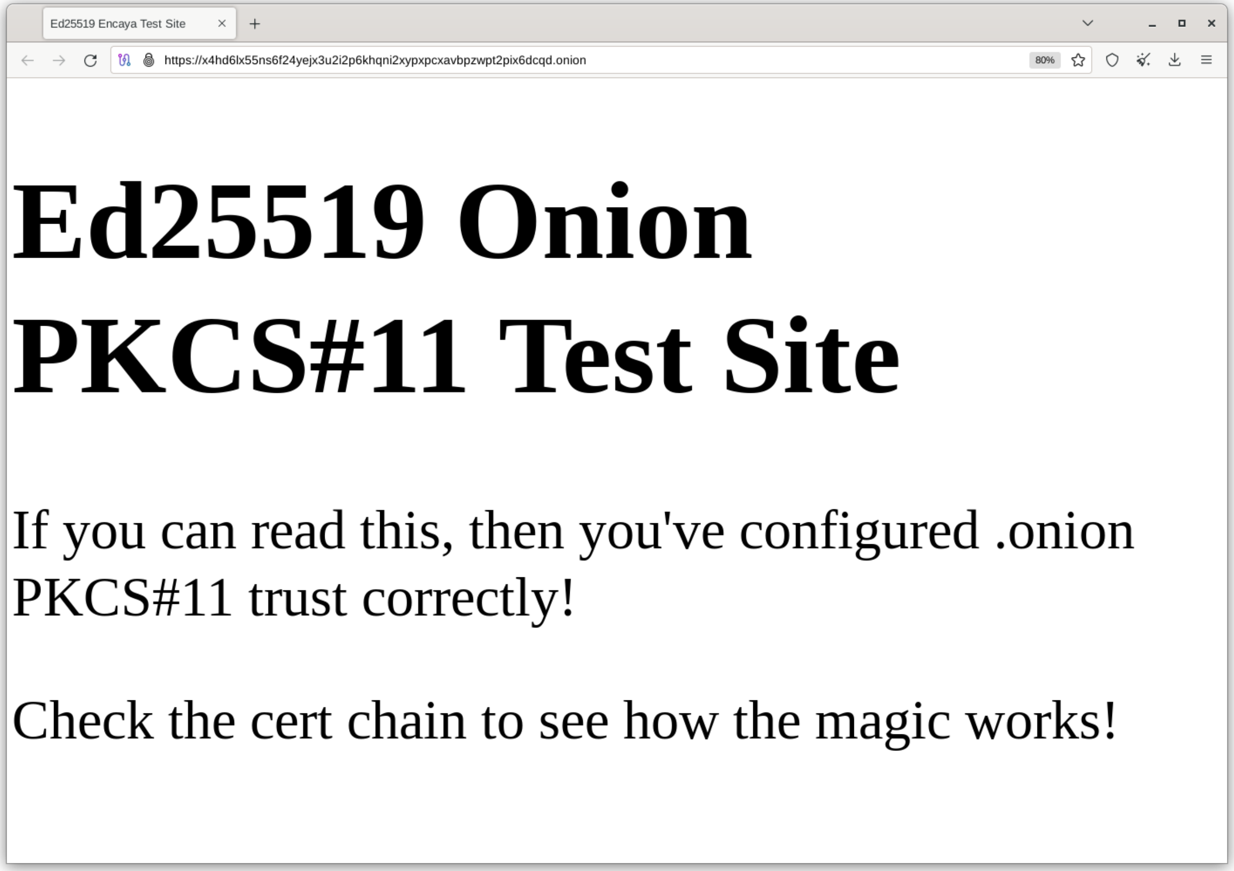
Deployment (Server-Side)

-
  ```
  sudo generate_nmc_cert -use-ca -use-aia -use-carrier -host x4hd6lx55ns6f24yejx3u2i2p6khqni2xypxpcxavbpzwpt2pix6dcqd.onion -grandparent-tor-key /var/lib/tor/hidden_service/hs_ed25519_secret_key
  ```
  - Put the resulting `chain.pem` and `key.pem` into Caddy. No network requests, no ACME client needed.

Deployment (Client-Side)

- Installing a PKCS#11 module in Firefox is easy.
  - Just go to “Security Devices” settings…
  - Click “Load” and choose the path of the PKCS#11 module…
    - It’s a `.so`, `.dylib` or `.dll` file depending on OS.
  - You’re done. Ed25519 onion certificates will work now.

Oh and it works in Tor Browser too.


Ed25519 Encaya Test Site
https://x4hd6lx55ns6f24yejx3u2i2p6khqni2xypxpcxavbpzwpt2pix6dcqd.onion
80%

Ed25519 Onion PKCS#11 Test Site

If you can read this, then you've configured .onion PKCS#11 trust correctly!

Check the cert chain to see how the magic works!

Compatibility

- Any application using NSS or GnuTLS knows how to use PKCS#11 for this.
- An analogous API called AIA is used by Chromium, and the Windows and macOS system cert validators.
  - Namecoin already ported our TLSA support to work with AIA.
  - .onion support is expected to be straightforward.
- Altogether that’s virtually all TLS implementations.
  - Exception is OpenSSL.

Certificate Transparency

- Some browsers mandate CT.
  - Policy set by a flag in PKCS#11 module.
  - We can exclude our module from browser CT requirements.

Will these API’s go away upstream?

- Using PKCS#11 for certificate database lookups has strong industry support.
- Red Hat is aggressively promoting it.
  - Red Hat is also adding support to OpenSSL.
- This is how Fedora and RHEL make all applications use the OS-level root CA list.
- Breaking our use case would probably break Red Hat’s use cases.
  - Seems OK for us.

Crypto Downside

- Even though Ed25519 is the trust anchor…
  - The cert chain still has ECDSA in it.
  - So if an attacker can break ECDSA, the better security of Ed25519 won’t protect you.
- Kind of unfortunate given that onion services use Ed25519 instead of ECDSA for good reason.

Thanks to funders

- NLnet Foundation’s Internet Hardening Fund / Netherlands Ministry of Economic Affairs and Climate Policy: funded me to do the Namecoin PKCS#11 module.
- Cyphrs: funded me to do the onion-specific and Ed25519-specific parts of the PKCS#11 module.

Next steps?

- This code and talk were written 2 weeks ago.
  - 1 week ago I discussed this work with Q Misell (ACME dev).
- Result? I get to rewrite my code soon!
  - New approach will put standard TLSA records in the onion service descriptor (downloaded by Tor clients when they connect to the onion service).
  - Much more similar to how DNS and Namecoin do it.
  - Huge thank you to Q for the feedback!

Contact me at...

- https://www.namecoin.org/
- OpenPGP (after mid-June):
  1D04 FB9D 50BF 2A8E 9F3E 58AD DC7E 7F8A E30E 73E6
- jeremyrand@danwin1210.de (after mid-June)
- byronlelah@airmail.cc (travel until mid-June – no PGP)
- DECT: NCJR / 6257 (during GPN)
- Or just find me here at GPN! (The Namecoin logo on my shirt should help you find me.)