Technical level: Intermediate

# Cryptoagility Today

## Why wait until 2030 when you can act in 2024?

By 2030, organisations are expected to phase out cryptographic algorithms with low-security strengths, and by 2035, NIST recommends transitioning fully to quantum-resistant alternatives. Waiting until 2030 to start this process is risky. Proactive planning today can streamline this shift without disruption.

> *Cryptoagility refers to the ability to swiftly replace cryptographic algorithms without disrupting operations. It should be as simple as changing your password.*

In my view, we should not wait for 2030 to improve the cryptographic strength currently in use. In most cases, with decent planning, this should be relatively straightforward. Simplified code is included using Certbot as an example, to make your life easier.

# X.509 Public Key Certificates

Adjust the following commands based on your domain verification method.

## ECDSA

This should be straightforward for a majority of the servers across the world.

> sudo certbot certonly --elliptic-curve secp521r1 --standalone -d example.com
>
> *Note: You could use secp384r1 instead of secp521r1 if there is a client compatibility issue.*

## EdDSA

Certbot does not directly support EdDSA and therefore I do not use these certificates for my websites. You could try to use OpenSSL to generate a Certificate Signing Request CSR using an EdDSA key and then use it with Certbot, which will require manual handling.

> openssl genpkey -algorithm ed25519 -out ed25519-key.pem
>
> sudo certbot certonly --manual --preferred-challenges dns -d example.com --csr ed25519-key.pem

If I were you, I would use ECDSA certificates instead (see above).

## RSA

Just to let you know, I stopped using RSA about five years ago. The code below is for the die-hard fans of the RSA.

> sudo certbot certonly --rsa-key-size 4096 --standalone -d example.com

If I were you, I would use ECDSA certificates instead (see above).

# SSH Keys

At the time of writing this note, there is no production level program to generate the key pair using the “mlkem768x25519-sha256” or “sntrup761x25519-sha512” algorithms. Therefore, although recent versions of OpenSSH server allow hybrid algorithms, there is currently no straightforward way to use them.

I therefore use defence-in-depth based on a EdDSA key pair. Although puttygen allows Ed25519 (255 bits) and Ed448 (448 bits), the latter is incompatible with the control panels of too many cloud service providers.

For now, Ed25519 (with a passphrase) is good enough when combined with the following defensive layers:

- RPKI (route authorisation against man-in-the-middle),
- rp_filter (against IP spoofing),
- IP restrictions (for access control),

- fail2ban (against brute force), and
- other ssh hardening measures.

## What else?

Researchers should keep an eye on the OQS experiments and try to build liboqs / openssl-oqs in their laboratories. Also be on the lookout for what Red Hat will introduce via the Fedora upstream.

Are you practicing cryptoagility? DMs are always open.

**Santosh Pandit**

18 Nov 2024

LinkedIn: https://www.linkedin.com/in/santoshpandit/