PQCRYPTO
ICT-645622

Horizon 2020

# PQCRYPTO
# Post-Quantum Cryptography for Long-Term Security

Project number: Horizon 2020 ICT-645622

# D2.1
# Internet: Preliminary portfolio

Due date of deliverable: 29. February 2016
Actual submission date: 18. March 2016

WP contributing to the deliverable: WP2

Start date of project: 1. March 2015 Duration: 3 years

Coordinator:
Technische Universiteit Eindhoven
Email: `coordinator@pqcrypto.eu.org`
`www.pqcrypto.eu.org`

Revision 0.8

| **Project co-funded by the European Commission within Horizon 2020** | | |
|---|---|---|
| **Dissemination Level** | | |
| **PU** | Public | X |
| **PP** | Restricted to other programme participants (including the Commission services) | |
| **RE** | Restricted to a group specified by the consortium (including the Commission services) | |
| **CO** | Confidential, only for members of the consortium (including the Commission services) | |

# Internet: Preliminary portfolio

Daniel J. Bernstein (editor)

18. March 2016
Revision 0.8

The work described in this report has in part been supported by the Commission of the European Communities through the Horizon 2020 program under project number 645622 PQCRYPTO. The information in this document is provided as is, and no warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

**Abstract**

This document describes what currently appear to be the most promising post-quantum systems for Internet applications. This document will be superseded by D2.3.

**Keywords:** SPHINCS, NTRU, Ring-LWE, QC-MDPC, McEliece, Niederreiter

## 1 Introduction

This document describes what currently appear to be the most promising post-quantum systems for Internet applications. This document will be superseded by D2.3.

This document focuses on two case studies. The first case study is software updates; the security goal here is integrity. The second case study is web browsing; the security goals here are confidentiality, integrity, and availability.

This document uses names of public-key cryptosystems such as NTRU. Complete specifications of these systems include secret-key components, and any 128-bit key is subject to a $2^{64}$ attack by Grover's algorithm, so these components should be designed with 256-bit keys.

## 2 Application: integrity for software updates

When your computer downloads a new version of its operating system (OS), it checks a signature on the download from the OS author. This is a critical use of cryptography: otherwise criminals could easily insert malware into your computer by sending you a fake version of the OS.

For example, updates to the security-oriented OpenBSD operating system are signed using a state-of-the-art elliptic-curve signature system, Ed25519. This signature system was selected by the IRTF Crypto Forum Research Group in 2015, paving the way for standardization in a broad range of IETF protocols, and is already deployed in many other applications.

Ed25519 was designed in 2011 by Bernstein (now PQCRYPTO WP2 leader), Duif (at the time a student at TU/e), Lange (now PQCRYPTO coordinator), Schwabe (now PQCRYPTO WP1 co-leader), and Yang (Academia Sinica, now part of PQCRYPTO), but this does not mean that it resists quantum computers. On the contrary: Ed25519, like the rest of elliptic-curve cryptography, will be broken by Shor's algorithm.

This raises the question of how to protect software updates against an attacker armed with a quantum computer. It is critical to fully deploy a high-security post-quantum solution *before* any quantum computers become available to attackers.

Beware that there is a large gap between finishing a high-security *implementation* and achieving full *deployment*, even when implementations are transparent upgrades available through standard upgrade channels. The problem is that Internet devices vary tremendously in their upgrade schedule. Some devices are *never* updated, as illustrated by the "Misfortune Cookie" attack: 12 million devices around the Internet in 2014 were exploitable through a vulnerability that had been fixed in 2005.

### 2.1 Combining pre-quantum and post-quantum systems

The obvious answer is to take each deployed pre-quantum signature system $P$ and replace it with a post-quantum signature $Q$.

The WP2 recommendation is slightly different: namely, to replace $P$ with $P + Q$. Here $P + Q$ is the signature system defined as follows: the secret key for $P + Q$ is a pair of independent secret keys, one for $P$ and one for $Q$; a $P + Q$ signature is a corresponding pair of signatures, one for $P$ and one for $Q$; the public key for $P + Q$ is the corresponding pair of public keys; verification for $P + Q$ means verification of *both* the $P$ signature and the $Q$ signature, so that forging a $P + Q$ signature requires forging both a $P$ signature and a $Q$ signature.

This recommendation is of course equivalent to a restriction of the original recommendation: namely, replacing $P$ with a post-quantum signature system $Q'$, specifically the system $Q' = P + Q$. What is important about this restriction is that the auditor can immediately see that $P + Q$ is as secure as $P$.

In contexts where the smallest possible key size is desired, one can replace the public key for $P+Q$ with its hash, and include the missing information inside each signature. The space required for the missing information is *at most* the space for the original $P + Q$ key, and can often be compressed further.

### 2.2 A concrete recommendation: Ed25519+SPHINCS-256

SPHINCS-256 is a stateless hash-based signature system designed to provide $2^{128}$ post-quantum security even beyond $2^{50}$ messages (more than 30 years signing $2^{20}$ messages per second) signed by a single key.

SPHINCS-256 signatures are 41KB, costing approximately 50 million cycles to generate and 1 million cycles to verify on a modern Intel CPU. These costs are negligible compared to the cost of an OS update. For example, in the Debian operating system, which is designed for frequent updates, the average size of a single package is 1.2MB.

The combined signature system Ed25519+SPHINCS-256 is visibly at least as secure as Ed25519, and visibly at least as secure as SPHINCS-256. Compared to just SPHINCS-256, the extra cost of Ed25519+SPHINCS-256 is unnoticeable in CPU time and bandwidth. There *is* a noticeable extra cost in the complexity of the cryptographic software, but the overall system includes Ed25519 code anyway.

### 2.3 The importance of auditing

Does deployment of $P + Q$ mean that we don't trust $Q$? On the contrary!

Our goal in designing a high-security post-quantum system $Q$ is for that system *by itself* to be safe for the foreseeable future. Even in the pre-quantum world, hash-based signatures such as SPHINCS-256 are confidence-inspiring *for the experts*, at least as confidence-inspiring as elliptic-curve signatures such as Ed25519. However, understanding this fact takes extra work *for the auditor*, while the auditor can immediately see that the security of Ed25519+SPHINCS-256 is at least as high as the security of SPHINCS-256. This simplification of the auditing process has considerable value for deployment: it removes an unnecessary source of fear.

The long-term situation is different. Users will see quantum computers breaking $P$ more and more easily, and in the meantime auditors will gain confidence in $Q$. The remaining value of including $P$ will continue to decrease, and eventually will be outweighed by the simplicity benefit of switching from $P + Q$ to $Q$.

### 2.4 Alternatives

Stateful hash-based signature systems, notably XMSS, are confidence-inspiring for the same reasons as SPHINCS-256, and provide considerably better performance than SPHINCS-256 for the same quantitative security level. However, in the context of signing software updates, performance pressure appears to be minimal. SPHINCS-256 has the advantage of being a drop-in replacement for existing pre-quantum signature systems. SPHINCS-256 software can be shared between users who can robustly maintain state and users who cannot, while XMSS code is suitable only for users of the first type.

The lack of performance pressure also means that SPHINCS-256 could be combined with other post-quantum signature systems, but this time there is a serious cost in complexity, apparently outweighing the potential security benefits.

## 3 Application: confidentiality, integrity, and availability for web browsing

This section considers a broader range of security goals in the context of web browsing: protecting confidentiality, integrity, and availability against espionage, corruption, and sabotage. Availability means that the correct data is provided, while integrity merely means that any incorrect data is detected. For example, an attacker overloading a network with a flood of random packets is compromising availability but not compromising integrity. The presence of confidentiality in the list of security goals means that full deployment of a high-security post-quantum solution must take place *many years before* any quantum computers become available to attackers, where the number of years is the number of years that the user needs data to remain confidential.

These goals raise considerably more difficult performance challenges and integration challenges than the goal of protecting the integrity of software updates. This difficulty is reflected by how poorly the Internet is achieving these goals today against *pre-quantum* attackers.

For example, there are no real availability protections in HTTPS, the main security mechanism for web browsing on the Internet today. An attacker forging a single TCP RST packet will destroy an entire HTTPS connection, and an attacker forging a single DNS packet will prevent the connection from being made in the first place. The HTTPS (TLS) software sees that something has gone wrong, but it has no way to recover. A browser using HTTPS can make a whole new connection, but this is slow and fragile. In short, a single forged packet causes huge damage. This attack is much less expensive, and optionally much more selective, than flooding a network.

WP2's original goal was to add post-quantum protection to aspects of the Internet that merely have pre-quantum protection today, but WP2 has identified a "stretch goal" of adding post-quantum protection to aspects of the Internet that have *no* protection today. Previous work by the editor of this document identified a strategy for using elliptic-curve cryptography to systematically authenticate and encrypt each Internet packet, and WP2 has begun analyzing options for achieving analogous results in the more challenging environment of post-quantum cryptography. Many of these options are "signature-free", using public-key encryption as a mechanism to exchange keys and thus to authenticate and encrypt subsequent packets, without any public-key signatures.

The most important desideratum identified by this analysis so far is that a ciphertext, including payload and some small routing data, needs to fit into the Internet's packet size, 1280 bytes. (The packet size actually varies from one physical network to another. The oldest Internet standards required $\geq$576; usually 1492 is safe, often 1500, sometimes more.) This allows a server to decrypt and act upon an encrypted packet from a new client without devoting any storage to that client.

Of course, it is also important for encryption and decryption to be fast enough, but work on several cryptosystems within PQCRYPTO is achieving speeds that meet the needs of typical applications. Another important cost constraint arises for users concerned with the following type of attack: steal a computer, and use keys stored inside that computer to retroactively decrypt stored ciphertext. Periodically erasing keys (providing "forward secrecy") limits the damage but increases the costs of generating and distributing keys.

## 3.1 Candidates

The original WP2 work description already identified the NTRU lattice-based encryption system as an attractive candidate for Internet deployment. NTRU and its variants ("NTRU Prime", the "New Hope" Ring-LWE system, etc.) are a major focus of activity in WP2. NTRU is patented, but the patent will expire in August 2017.

WP2 has not reached consensus on a single variant of NTRU as being clearly best. Work is therefore proceeding on several different variants. Another hub of activity is QC-MDPC, which can be viewed as a "code-based NTRU" and seems likely to perform better than NTRU in hardware. Note that some of the most important cryptographic primitives have been embedded into mass-market CPUs as hardware coprocessors; selecting hardware-friendly primitives for Internet use is one strategy to reduce long-term deployment costs.

A surprising side effect of the ciphertext-size desideratum described above is to draw new attention to the classic McEliece code-based encryption system. The dual Niederreiter form of this system has much smaller ciphertext overhead than NTRU, around 200 bytes at a high security level, allowing many more payload bytes to be included in an Internet packet. This system is also older than NTRU and has a more stable security track record. The main disadvantage of this system is its large key size, on the scale of a megabyte, but initial performance analyses suggest that this does not preclude deployment on most web sites, even with fairly frequent key erasure.