Post-quantum cryptography

Tanja Lange

PQCRYPTO
ICT-645622

07 October 2015

SPACE 2015

In the long term, all encryption needs to be post-quantum

- Mark Ketchen, IBM Research, 2012, on quantum computing: "Were actually doing things that are making us think like, 'hey this isn't 50 years off, this is maybe just 10 years off, or 15 years off.' It's within reach."

In the long term, all encryption needs to be post-quantum

- Mark Ketchen, IBM Research, 2012, on quantum computing: "Were actually doing things that are making us think like, 'hey this isn't 50 years off, this is maybe just 10 years off, or 15 years off.' It's within reach."
- Fast-forward to 2022, or 2027. Quantum computers exist.
- Shor's algorithm solves in polynomial time:
  - Integer factorization.
  - The discrete-logarithm problem in finite fields.
  - The discrete-logarithm problem on elliptic curves.
- This breaks all current public-key encryption on the Internet!

In the long term, all encryption needs to be post-quantum

- Mark Ketchen, IBM Research, 2012, on quantum computing: "Were actually doing things that are making us think like, 'hey this isn't 50 years off, this is maybe just 10 years off, or 15 years off.' It's within reach."
- Fast-forward to 2022, or 2027. Quantum computers exist.
- Shor's algorithm solves in polynomial time:
  - Integer factorization.
  - The discrete-logarithm problem in finite fields.
  - The discrete-logarithm problem on elliptic curves.
- This breaks all current public-key encryption on the Internet!
- Also, Grover's algorithm speeds up brute-force searches.
- Example: Only 2^{64} quantum operations to break AES-128.

PQCRYPTO
ICT-645622

In the long term, all encryption needs to be post-quantum

- Mark Ketchen, IBM Research, 2012, on quantum computing: “Were actually doing things that are making us think like, ‘hey this isn’t 50 years off, this is maybe just 10 years off, or 15 years off.’ It’s within reach.”
- Fast-forward to 2022, or 2027. Quantum computers exist.
- Shor’s algorithm solves in polynomial time:
  - Integer factorization.
  - The discrete-logarithm problem in finite fields.
  - The discrete-logarithm problem on elliptic curves.
- This breaks all current public-key encryption on the Internet!
- Also, Grover’s algorithm speeds up brute-force searches.
- Example: Only 2^{64} quantum operations to break AES-128.
- Need to switch the Internet to post-quantum encryption.

PQCRYPTO
ICT-645622

Confidence-inspiring crypto takes time to build

- Many stages of research from cryptographic design to deployment:
  - Explore space of cryptosystems.
  - Study algorithms for the attackers.
  - Focus on secure cryptosystems.

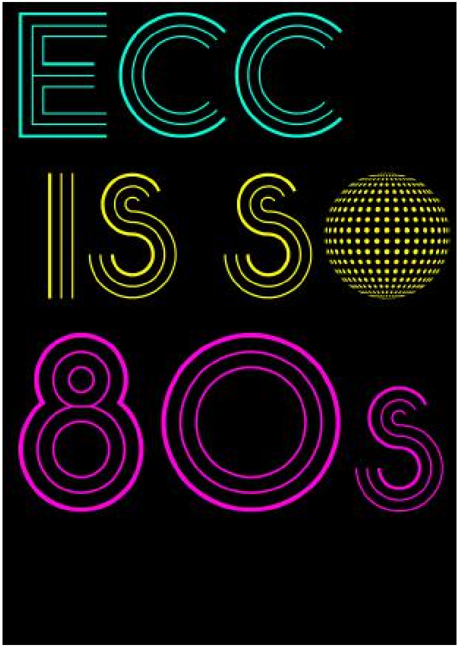
PQCRYPTO
ICT-645622

Confidence-inspiring crypto takes time to build

- Many stages of research from cryptographic design to deployment:
  - Explore space of cryptosystems.
  - Study algorithms for the attackers.
  - Focus on secure cryptosystems.
  - Study algorithms for the users.
  - Study implementations on real hardware.
  - Study side-channel attacks, fault attacks, etc.
  - Focus on secure, reliable implementations.
  - Focus on implementations meeting performance requirements.
  - Integrate securely into real-world applications.

Confidence-inspiring crypto takes time to build

- Many stages of research from cryptographic design to deployment:
  - Explore space of cryptosystems.
  - Study algorithms for the attackers.
  - Focus on secure cryptosystems.
  - Study algorithms for the users.
  - Study implementations on real hardware.
  - Study side-channel attacks, fault attacks, etc.
  - Focus on secure, reliable implementations.
  - Focus on implementations meeting performance requirements.
  - Integrate securely into real-world applications.
- Example: ECC introduced **1985**; big advantages over RSA. Robust ECC is starting to take over the Internet in **2015**.
- Post-quantum research can't wait for quantum computers!


ECC
IS SO
80s

Even higher urgency for long-term confidentiality

- Today's encrypted communication is being stored by attackers and will be decrypted years later with quantum computers. Danger for human-rights workers, medical records, journalists, security research, legal proceedings, state secrets, ...

TOP SECRET//COMINT//REL TO USA, AUS, CAN, GBR, NZL
Where is X-KEYSCORE?
KEYSC

Post-Quantum Cryptography for Long-term Security

- Project funded by EU in Horizon 2020.
- Starting date 1 March 2015, runs for 3 years.
- 11 partners from academia and industry, TU/e is coordinator

TU/e technische universiteit eindhoven

BUNDES DRUCKEREI

DTU Danmarks Tekniske Universitet

Inria INVENTEURS DU MONDE NUMÉRIQUE

KU LEUVEN

NXP

50 Jahre RUB

Radboud Universiteit IN DEI NOMINE FELICITER

TECHNISCHE UNIVERSITÄT DARMSTADT

אוניברסיטת חיפה University of Haifa جامعة حيفا University of Haifa

中央研究院 ACADEMIA SINICA 1928

Impact of PQCRYPTO

- All currently used public-key systems on the Internet are broken by quantum computers.
- Today's encrypted communication can be (and is being!) stored by attackers and can be decrypted later with quantum computer.
- Post-quantum secure cryptosystems exist but are under-researched – we can recommend secure systems now, but they are big and slow

Impact of PQCRYPTO

- All currently used public-key systems on the Internet are broken by quantum computers.
- Today's encrypted communication can be (and is being!) stored by attackers and can be decrypted later with quantum computer.
- Post-quantum secure cryptosystems exist but are under-researched – we can recommend secure systems now, but they are big and slow hence the logo.

Impact of PQCRYPTO

- All currently used public-key systems on the Internet are broken by quantum computers.
- Today's encrypted communication can be (and is being!) stored by attackers and can be decrypted later with quantum computer.
- Post-quantum secure cryptosystems exist but are under-researched – we can recommend secure systems now, but they are big and slow hence the logo.
- PQCRYPTO will design a portfolio of high-security post-quantum public-key systems, and will improve the speed of these systems, adapting to the different performance challenges of mobile devices, the cloud, and the Internet.
- PQCRYPTO will provide efficient implementations of high-security post-quantum cryptography for a broad spectrum of real-world applications.

WP1: Post-quantum cryptography for small devices

Leaders: Tim Güneysu, co-leader: Peter Schwabe

- Find post-quantum secure cryptosystems suitable for small devices in power and memory requirements (e.g. smart cards with 8-bit or 16-bit or 32-bit architectures, with different amounts of RAM, with or without coprocessors).
- Develop efficient implementations of these systems.
- Investigate and improve their security against implementation attacks.
- Deliverables include reference implementations and optimized implementations for software for platforms ranging from small 8-bit microcontrollers to more powerful 32-bit ARM processors.
- Deliverables also include FPGA and ASIC designs and physical security analysis.

WP2: Post-quantum cryptography for the Internet

Leaders: Daniel J. Bernstein, co-leader: Bart Preneel

- Find post-quantum secure cryptosystems suitable for busy Internet servers handling many clients simultaneously.
- Develop secure and efficient implementations.
- Integrate these systems into Internet protocols.
- Deliverables include software library for all common Internet platforms, including large server CPUs, smaller desktop and laptop CPUs, netbook CPUs (Atom, Bobcat, etc.), and smartphone CPUs (ARM).
- Aim is to get high-security post-quantum crypto ready for the Internet.

WP3: Post-quantum cryptography for the cloud

Leaders: Nicolas Sendrier, co-leader: Lars Knudsen

- Provide 50 years of protection for files that users store in the cloud, even if the cloud service providers are not trustworthy.
- Allow sharing and editing of cloud data under user-specified security policies.
- Support advanced cloud applications such as privacy-preserving keyword search.
- Work includes public-key and symmetric-key cryptography.
- Prioritize high security and speed over key size.

What does PQCRYPTO mean for you?

- Expert recommendations for post-quantum secure cryptosystems.
- Recommended systems will get faster/smaller as result of PQCRYPTO research.
- More benchmarking to compare cryptosystems.
- Cryptographic libraries will be made freely available for several computer architectures.
- Workshop and "summer" school on post-quantum cryptography (Spring or summer 2017)
- Find more information online at http://pqcrypto.eu.org/.
- Follow us on twitter https://twitter.com/pqc_eu.

PQCRYPTO
ICT-645622

State of the art in post-quantum encryption

- Code-based encryption: e.g., 1978 McEliece.
  - Attacker tries to correct errors for a "random-looking" code.
  - Which codes should users take? Start from Reed–Solomon; add scaling? permutation? puncturing? subcodes? subfields? wildness? list decoding? increased genus? Or start from LDPC? MDPC? QC-MDPC? QD-MDPC? Rank metric? . . .
  - Some papers studying algorithms for attackers:
    1962 Prange; 1981 Omura; 1988 Lee–Brickell; 1988 Leon; 1989 Krouk; 1989 Stern; 1989 Dumer; 1990 Coffey–Goodman; 1990 van Tilburg; 1991 Dumer; 1991 Coffey–Goodman–Farrell; 1993 Chabanne–Courteau; 1993 Chabaud; 1994 van Tilburg; 1994 Canteaut–Chabanne; 1998 Canteaut–Chabaud; 1998 Canteaut–Sendrier; 2008 Bernstein–Lange–Peters; 2009 Bernstein–Lange–Peters–van Tilborg; 2009 Bernstein (post-quantum); 2009 Finiasz–Sendrier; 2010 Bernstein–Lange–Peters; 2011 May–Meurer–Thomae; 2011 Becker–Coron–Joux; 2012 Becker–Joux–May–Meurer; 2013 Bernstein–Jeffery–Lange–Meurer (post-quantum); 2015 May–Ozerov.
  - We have confidence in scaled-up McEliece, but keys are huge.
  - QC-MDPC: much smaller keys, but is it secure?
  - Side-channel protection? Higher-level protocols? . . .
- Lattice-based encryption: even more complex.
- Multivariate-quadratic encryption.

Linear Codes (following slides from Tung Chou)

A binary linear code C of length n and dimension k is a k-dimensional subspace of $\mathbb{F}_2^n$.

C is usually specified as

- the row space of a generating matrix $G \in \mathbb{F}_2^{k\times n}$

$$C = \{\mathbf{m}G | \mathbf{m} \in \mathbb{F}_2^k\}$$

- the kernel space of a parity-check matrix $H \in \mathbb{F}_2^{(n-k)\times n}$

$$C = \{\mathbf{c} | H\mathbf{c}^\intercal = 0,\ \ \mathbf{c} \in \mathbb{F}_2^n\}$$

Example:

$$G = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

$\mathbf{c} = (111)G = (10011)$ is a codeword.

Weight, distance, decoding problem

- The Hamming weight of a word is the number of nonzero coordinates.
- The Hamming distance between two words in $\mathbb{F}_2^n$ is the number of coordinates where they differ.

Decoding problem: find the closest codeword $\mathbf{c} \in C$ to a given $\mathbf{r} \in \mathbb{F}_2^n$, assuming that there is a unique closest codeword. Let $\mathbf{r} = \mathbf{c} + \mathbf{e}$. Note that finding $\mathbf{e}$ is an equivalent problem.

- $\mathbf{e}$ is called the error vector.
- If $\mathbf{c}$ is t errors away from $\mathbf{r}$, i.e., the Hamming weight of $\mathbf{e}$ is t, this is called a t-error correcting problem.
- There are lots of code families with fast decoding algorithms, e.g., Reed–Solomon codes, Goppa codes/alternant codes, etc.
- However, the general decoding problem is hard: Information-set decoding takes exponential time.

Binary Goppa code

A binary Goppa code is often defined by

- a list $L = (a_1, \dots, a_n)$ of n distinct elements in $\mathbb{F}_q$, called the support.
- a square-free polynomial $g(x) \in \mathbb{F}_q[x]$ of degree t such that $g(a) \neq 0$ for all $a \in L$. $g(x)$ is called the Goppa polynomial.

Then the corresponding binary Goppa code, denoted as $\Gamma(L, g)$, is the set of words $c = (c_1, \dots, c_n) \in \mathbb{F}_2^n$ that satisfy

$$\frac{c_1}{x - a_1} + \frac{c_2}{x - a_2} + \cdots + \frac{c_n}{x - a_n} \equiv 0 \pmod{g(x)}$$

- can correct t errors
- used in code-based cryptography

The Niederreiter cryptosystem

Developed in 1986 by Harald Niederreiter as a variant of the McEliece cryptosystem.

- Public Key: a parity-check matrix $K \in \mathbb{F}_q^{(n-k)\times n}$ for the binary Goppa code
- Encryption: The plaintext $\mathbf{m}$ is an n-bit vector of weight t. The ciphertext $\mathbf{c}$ is an $(n-k)$-bit vector:
$$\mathbf{c}^\intercal = K\mathbf{m}^\intercal.$$
- Decryption: Find a n-bit vector $\mathbf{r}$ such that
$$\mathbf{c}^\intercal = K\mathbf{r}^\intercal,$$
then use any available decoder to decode $\mathbf{r}$. Can just let $\mathbf{r}$ be the ciphertext followed by k zeros, so **decryption is basically decoding**.
- The passive attacker is facing a t-error correcting problem for the public key, which seems to be random.

McBits

Daniel J. Bernstein, Tung Chou, Peter Schwabe, CHES 2013.

- Encryption is super fast anyways (just a vector-matrix multiplication).
- Main step in decryption is decoding of Goppa code. The McBits software achieves this in constant time.
- Decoding speed at 2^{128} pre-quantum security: $(n; t) = (4096; 41)$ uses 60493 Ivy Bridge cycles.
- Decoding speed at 2^{263} pre-quantum security: $(n; t) = (6960; 119)$ uses 306102 Ivy Bridge cycles.
- Grover speedup is less than halving the security level, so the latter parameters offer at least 2^{128} post-quantum security.

SPHINCS: practical stateless hash-based signatures

Daniel J. Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Peter Schwabe, Zooko Wilcox-O'Hearn

Hash-based signatures

- 1979 Lamport one-time signature scheme.
- Fix a k-bit one-way function $G : \{0,1\}^k \to \{0,1\}^k$ and hash function $H : \{0,1\}^* \to \{0,1\}^k$.
- Signer's secret key X: $2k$ strings $x_1[0], x_1[1], \ldots, x_k[0], x_k[1]$, each k bits. Total: $2k^2$ bits.
- Signer's public key Y: $2k$ strings $y_1[0], y_1[1], \ldots, y_k[0], y_k[1]$, each k bits, computed as $y_i[b] = G(x_i[b])$. Total: $2k^2$ bits.
- Signature $S(X, r, m)$ of a message m: $r, x_1[h_1], \ldots, x_k[h_k]$ where $H(r, m) = (h_1, \ldots, h_k)$.
- Must never use secret key more than once.
- Usually choose $G = H$ (restricted to k bits).
- 1979 Merkle extends to more signatures.

8-time Merkle hash tree

Eight Lamport one-time keys $Y_1, Y_2, \ldots, Y_8$ with corresponding $X_1, X_2, \ldots, X_8$, where $X_i = (x_{i,1}[0], x_{i,1}[1], \ldots, x_{i,k}[0], x_{i,k}[1])$ and $Y_i = (y_{i,1}[0], y_{i,1}[1], \ldots, y_{i,k}[0], y_{i,k}[1])$.

$$\begin{tikzcd}
X_1 \arrow[d] & X_2 \arrow[d] & X_3 \arrow[d] & X_4 \arrow[d] & X_5 \arrow[d] & X_6 \arrow[d] & X_7 \arrow[d] & X_8 \arrow[d] \\
Y_1 \arrow[dr] & Y_2 \arrow[d] & Y_3 \arrow[dr] & Y_4 \arrow[d] & Y_5 \arrow[dr] & Y_6 \arrow[d] & Y_7 \arrow[dr] & Y_8 \arrow[d] \\
 & Y_9 = H(Y_1, Y_2) \arrow[dr] & & Y_{10} = H(Y_3, Y_4) \arrow[dl] & & Y_{11} = H(Y_5, Y_6) \arrow[dr] & & Y_{12} = H(Y_7, Y_8) \arrow[dl] \\
 & & Y_{13} = H(Y_9, Y_{10}) \arrow[drr] & & & & Y_{14} = H(Y_{11}, Y_{12}) \arrow[dll] & \\
 & & & & Y_{15} = H(Y_{13}, Y_{14}) & & &
\end{tikzcd}$$

The Merkle public key is Y_{15}.

Signature in 8-time Merkle hash tree

First message has signature is $(S(X_1, r, m), Y_1, Y_2, Y_{10}, Y_{14})$.

$$\begin{tikzcd}
\boxed{X_1} \arrow[d] & X_2 \arrow[d] & X_3 \arrow[d] & X_4 \arrow[d] & X_5 \arrow[d] & X_6 \arrow[d] & X_7 \arrow[d] & X_8 \arrow[d] \\
\boxed{Y_1} \arrow[dr] & \boxed{Y_2} \arrow[d] & Y_3 \arrow[dr] & Y_4 \arrow[d] & Y_5 \arrow[dr] & Y_6 \arrow[d] & Y_7 \arrow[dr] & Y_8 \arrow[d] \\
& Y_9 = H(Y_1, Y_2) \arrow[drr] & & \boxed{Y_{10} = H(Y_3, Y_4)} \arrow[d] & & Y_{11} = H(Y_5, Y_6) \arrow[drr] & & Y_{12} = H(Y_7, Y_8) \arrow[d] \\
& & & Y_{13} = H(Y_9, Y_{10}) \arrow[dr] & & & & \boxed{Y_{14} = H(Y_{11}, Y_{12})} \arrow[dlll] \\
& & & & \boxed{Y_{15} = H(Y_{13}, Y_{14})} & & &
\end{tikzcd}$$

Signature in 8-time Merkle hash tree

First message has signature is $(S(X_1, r, m), Y_1, Y_2, Y_{10}, Y_{14})$.

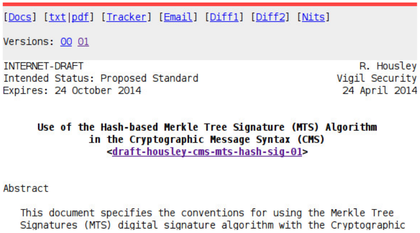
$$\begin{tikzcd}
\boxed{X_1} \arrow[d] & X_2 \arrow[d] & X_3 \arrow[d] & X_4 \arrow[d] & X_5 \arrow[d] & X_6 \arrow[d] & X_7 \arrow[d] & X_8 \arrow[d] \\
\boxed{Y_1} \arrow[d] & \boxed{Y_2} \arrow[dl] & Y_3 \arrow[d] & Y_4 \arrow[dl] & Y_5 \arrow[d] & Y_6 \arrow[dl] & Y_7 \arrow[d] & Y_8 \arrow[dl] \\
Y_9 = H(Y_1, Y_2) \arrow[dr] & & \boxed{Y_{10} = H(Y_3, Y_4)} \arrow[dl] & & Y_{11} = H(Y_5, Y_6) \arrow[dr] & & Y_{12} = H(Y_7, Y_8) \arrow[dl] & \\
& Y_{13} = H(Y_9, Y_{10}) \arrow[drr] & & & & \boxed{Y_{14} = H(Y_{11}, Y_{12})} \arrow[dll] & & \\
& & & \boxed{Y_{15} = H(Y_{13}, Y_{14})} & & & &
\end{tikzcd}$$

Verify by checking signature $S(X_1, r, m)$ on m against Y_1. Link Y_1 against public key Y_{15} by computing $Y'_9 = H(Y_1, Y_2)$, $Y'_{13} = H(Y'_9, Y_{10})$, and comparing $H(Y'_{13}, Y_{14})$ with Y_{15}.

Pros and cons

Pros:

- Post quantum
- Only need secure hash function
- Small public key
- Security well understood
- Fast
- Proposed for standards http://tools.ietf.org/html/draft-housley-cms-mts-hash-sig-01

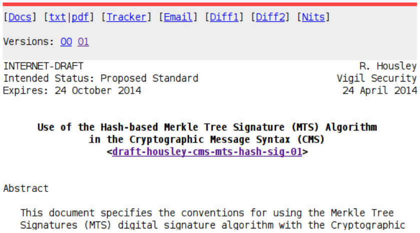


```
[Docs] [txt|pdf] [Tracker] [Email] [Diff1] [Diff2] [Nits]

Versions: 00 01

INTERNET-DRAFT                                              R. Housley
Intended Status: Proposed Standard                      Vigil Security
Expires: 24 October 2014                                 24 April 2014

       Use of the Hash-based Merkle Tree Signature (MTS) Algorithm
              in the Cryptographic Message Syntax (CMS)
                   <draft-housley-cms-mts-hash-sig-01>

Abstract

   This document specifies the conventions for using the Merkle Tree
   Signatures (MTS) digital signature algorithm with the Cryptographic
```

Pros and cons

Pros:

- Post quantum
- Only need secure hash function
- Small public key
- Security well understood
- Fast
- Proposed for standards http://tools.ietf.org/html/draft-housley-cms-mts-hash-sig-01



[Docs] [txt|pdf] [Tracker] [Email] [Diff1] [Diff2] [Nits]

Versions: 00 01

```
INTERNET-DRAFT                                              R. Housley
Intended Status: Proposed Standard                      Vigil Security
Expires: 24 October 2014                                 24 April 2014
```

Use of the Hash-based Merkle Tree Signature (MTS) Algorithm in the Cryptographic Message Syntax (CMS)
<draft-housley-cms-mts-hash-sig-01>

Abstract

This document specifies the conventions for using the Merkle Tree Signatures (MTS) digital signature algorithm with the Cryptographic

Cons:

- Biggish signature and secret key
- Stateful

  Adam Langley "for most environments it's a huge foot-cannon."

ELMNATE
THE STATE

Huge trees (1987 Goldreich), keys on demand (Levin)

Signer chooses random $r \in \{2^{255}, 2^{255}+1, \ldots, 2^{256}-1\}$,
uses one-time public key T_r to sign message;
uses one-time public key T_i to sign (T_{2i}, T_{2i+1}) for $i < 2^{255}$.
Generates ith secret key as $H_k(i)$ where k is master secret.

$$\begin{tikzcd}
 & & & T_1 \arrow[dll] \arrow[drr] & & & \\
 & T_2 \arrow[dl] \arrow[dr] & & & & T_3 \arrow[dl] \arrow[dr] & \\
\iddots \arrow[dr] & & \ddots & & \iddots & & \ddots \arrow[dl] \\
 & T_{2^{254}} \arrow[dl] \arrow[d] & & \vdots \arrow[d] & & T_{2^{255}-1} \arrow[d] \arrow[dr] & \\
T_{2^{255}} & T_{2^{255}+1} & \cdots & T_r \arrow[d] & \cdots & T_{2^{256}-2} & T_{2^{256}-1} \\
 & & & m & & &
\end{tikzcd}$$

It works, but signatures are painfully long

0.6 MB for hash-based Goldreich signature using short-public-key Winternitz-16 one-time signatures.

Would dominate traffic in typical applications, and add user-visible latency on typical network connections.

It works, but signatures are painfully long

0.6 MB for hash-based Goldreich signature using
short-public-key Winternitz-16 one-time signatures.

Would dominate traffic in typical applications,
and add user-visible latency on typical network connections.

Example:
Debian operating system is designed for frequent upgrades.
At least one new signature for each upgrade.
Typical upgrade: one package or just a few packages.
1.2 MB average package size.
0.08 MB median package size.

It works, but signatures are painfully long

0.6 MB for hash-based Goldreich signature using
short-public-key Winternitz-16 one-time signatures.

Would dominate traffic in typical applications,
and add user-visible latency on typical network connections.

Example:
Debian operating system is designed for frequent upgrades.
At least one new signature for each upgrade.
Typical upgrade: one package or just a few packages.
1.2 MB average package size.
0.08 MB median package size.

Example:
HTTPS typically sends multiple signatures per page.
1.8 MB average web page in Alexa Top 1000000.

New: SPHINCS-256

Reasonable sizes.
0.041 MB signature.
0.001 MB public key.
0.001 MB private key.

Reasonable speeds.
Benchmarks of our public-domain software on Haswell:
51.1 million cycles to sign. (RSA-3072: 14.2 million.)
1.5 million cycles to verify. (RSA-3072: 0.1 million.)
3.2 million cycles for keygen. (RSA-3072: 950 million.)

New: SPHINCS-256

Reasonable sizes.
0.041 MB signature.
0.001 MB public key.
0.001 MB private key.

Reasonable speeds.
Benchmarks of our public-domain software on Haswell:
51.1 million cycles to sign. (RSA-3072: 14.2 million.)
1.5 million cycles to verify. (RSA-3072: 0.1 million.)
3.2 million cycles for keygen. (RSA-3072: 950 million.)

Designed for 2^{128} post-quantum security,
even for a user signing more than 2^{50} messages:
2^{20} messages/second continuously for more than 30 years.
Yes, we did the analysis of quantum attacks.

Ingredients of SPHINCS (and SPHINCS-256)

Drastically reduce tree height (to 60).

Replace one-time leaves with few-time leaves.

Optimize few-time signature size *plus* key size.
New few-time HORST, improving upon HORS.

Use hyper-trees (12 layers), as in GMSS.

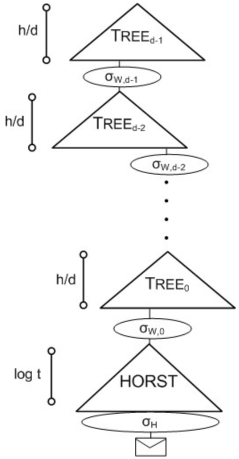
Use masks, as in XMSS and XMSS^{MT},
for standard-model security proofs.

Optimize short-input (256-bit) hashing speed.
Use sponge hash (with ChaCha12 permutation).

Use fast stream cipher (again ChaCha12).

Vectorize hash software and cipher software.

See paper for details: sphincs.cr.yp.to



Initial recommendations of long-term secure post-quantum systems

Daniel Augot, Lejla Batina, Daniel J. Bernstein, Joppe Bos, Johannes Buchmann, Wouter Castryck, Orr Dunkelman, Tim Güneysu, Shay Gueron, Andreas Hülsing, Tanja Lange, Mohamed Saied Emam Mohamed, Christian Rechberger, Peter Schwabe, Nicolas Sendrier, Frederik Vercauteren, Bo-Yin Yang

Initial recommendations

- **Symmetric encryption** Thoroughly analyzed, 256-bit keys:
  - AES-256
  - Salsa20 with a 256-bit key

  Evaluating: Serpent-256, . . .
- **Symmetric authentication** Information-theoretic MACs:
  - GCM using a 96-bit nonce and a 128-bit authenticator
  - Poly1305
- **Public-key encryption** McEliece with binary Goppa codes:
  - length $n = 6960$, dimension $k = 5413$, $t = 119$ errors

  Evaluating: QC-MDPC, Stehlé-Steinfeld NTRU, . . .
- **Public-key signatures** Hash-based (minimal assumptions):
  - XMSS with any of the parameters specified in CFRG draft
  - SPHINCS-256

  Evaluating: HFEv-, . . .