Post-quantum cryptography – dealing with the fallout of physics success

Daniel J. Bernstein[1,2] and Tanja Lange[1]

[1]Technische Universiteit Eindhoven

[2]University of Illinois at Chicago

Abstract

Cryptography is essential for the security of Internet communication, cars, and implanted medical devices. However, many commonly used cryptosystems will be completely broken once big quantum computers exist.

Post-quantum cryptography is cryptography under the assumption that the attacker has a large quantum computer; post-quantum cryptosystems strive to remain secure even in this scenario. This relatively young research area has seen some successes in identifying mathematical operations for which quantum algorithms offer little speedup, and then building cryptographic systems around those. The central challenge in post-quantum cryptography is to meet demands for cryptographic usability and flexibility without sacrificing trust.

1 Introduction

Attackers are recording, and sometimes forging, vast volumes of human communication. Some of this communication is protected by cryptographic systems such as RSA and ECC, but if quantum computing scales as expected then it will break both RSA and ECC. We are in a race against time to deploy *post-quantum cryptography* before quantum computers arrive.

This article reviews what cryptography does, and the damage done by quantum computing. It then surveys several major candidates for post-quantum cryptography, and concludes with a look to the future.

Acknowledgments. Thanks to Andreas Hülsing and Bo-Yin Yang for their comments.

Author list in alphabetical order; see https://www.ams.org/profession/leaders/culture/CultureStatement04.pdf. This work was supported by the European Commission under Contract ICT-645622 PQCRYPTO; by the Netherlands Organisation for Scientific Research (NWO) under grant 639.073.005; and by the U.S. National Science Foundation under grant 1314919. “Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation” (or other funding agencies). Permanent ID of this document: `ad292de5977a3bee1dff5ee0b0f04115b2b47b3e`. Date: 2017.04.09.

2 Introduction to cryptography

When a user visits a website starting with `https`, the user's computer uses "Transport Layer Security" (TLS) to connect securely to the web server. TLS combines a sequence of cryptographic operations to ensure that no third party can understand what is being sent (confidentiality), that no third party can modify the messages without being detected (integrity), and that no third party can impersonate one of the communicating parties (authenticity).

Cryptographers typically call the communicating parties Alice and Bob, even though in real life the parties are typically computing devices, such as desktops, laptops, cell phones, or smart cards. A party is identified by knowing certain secret values that are unique to it or to a connection. Cryptographic functions can be categorized into symmetric functions and public-key functions, depending on who has access to those secret values, the *keys*.

In *symmetric cryptography* a key is known to both communicating parties. An encryption algorithm requires use of an encryption key k_{enc} and the matching decryption algorithm uses the same key. Alice uses k_{enc} to encrypt a message m into a ciphertext c for Bob; Bob then uses k_{enc} to decrypt c obtaining m. An encryption function is called a *block cipher* or *stream cipher* depending on whether it requires m to have a fixed length (the block length) or can handle a stream of bits of arbitrary length. A block cipher can be turned into a stream cipher by defining how messages longer than one block are processed and how to fill up short messages to the block length by using a form of padding.

Symmetric keys can also be used for authentication: if only Alice and Bob know k_{auth} then Alice can apply k_{auth} to a challenge string provided by Bob to prove that she has access to the key and Bob can verify the computation. Alice can also apply k_{auth} to a ciphertext; this provides not only authenticity but also integrity for the message. These cryptographic functions are called *message-authentication codes* or just *MACs*.

Hash functions map strings of arbitrary length to strings of some fixed length n. In this article we consider only hash functions designed to make the following operations computationally hard: (1) given a value z in the image of h find a preimage, i.e. a string m with $h(m) = z$; (2) given a string m and $h(m)$ find a second preimage, i.e. a string $m' \neq m$ with $h(m) = h(m')$; and (3) find a collision, i.e. strings $m \neq m'$ with $h(m) = h(m')$. Hash functions provide compact fingerprints of messages; a small change to the message produces a completely different fingerprint. MACs are often built from hash functions.

In *public-key cryptography* each party has two keys: a public key and a private key. The private key is only known to the party while the public key can be made public. Given Alice's public encryption key anybody can encrypt a message to her while only she is in possession of the matching private key which she uses to decrypt.

Public-key *signatures* authenticate messages with public-key cryptography. A signature system has Alice perform a computation using her private signing key and the message m in a way that everybody can verify it using her public signing key but that

Name	**function**	**pre-quantum security level**	**post-quantum security level**
Symmetric cryptography			
AES-128 [1]	block cipher	128	64 (Grover)
AES-256 [1]	block cipher	256	128 (Grover)
Salsa20 [2]	stream cipher	256	128 (Grover)
GMAC [3]	MAC	128	128 (no impact)
Poly1305 [4]	MAC	128	128 (no impact)
SHA-256 [5]	hash function	256	128 (Grover)
SHA-3 [6]	hash function	256	128 (Grover)
Public-key cryptography			
RSA-3072 [7]	encryption	128	broken (Shor)
RSA-3072 [7]	signature	128	broken (Shor)
DH-3072 [8]	key exchange	128	broken (Shor)
DSA-3072 [9, 10]	signature	128	broken (Shor)
256-bit ECDH [11, 12, 13]	key exchange	128	broken (Shor)
256-bit ECDSA [14, 15]	signature	128	broken (Shor)

Table 1: Examples of widely deployed cryptographic systems and their security levels against the best pre-quantum and post-quantum attacks known. Security level b means that the best attacks use approximately 2^b operations. For hash functions, "security" in this table refers to preimage security.

nobody else could have produced it. Like a MAC, a signature system ensures authenticity and integrity of the message. The key differences are that a signature can be verified by anybody using the public key, while a MAC is between the parties sharing k_{auth}, and that only one single party could have produced a valid signature while a MAC could have been computed by Alice or Bob.

Public-key signatures are used in certificates to guarantee authenticity of websites, in passports to certify the carrier as a citizen and to ensure integrity of the stored information, and in operating system upgrades to ensure that the files originated from a trusted source and have not been modified. These files are typically very large; the signature system signs hashes of the files.

In TLS, public-key encryption is used in a first phase for Alice and Bob to establish a shared symmetric key. Signatures guarantee the authenticity of the public keys. The bulk of the communication is then secured by symmetric cryptography using a block or stream cipher along with a MAC.

3 The apocalypse: Shor's algorithm

In the popular RSA public-key system [7], the public key is a product $N = pq$ of two secret prime numbers p, q. The security of RSA relies critically on the difficulty of finding the factors p, q of N. However, in 1994, Shor [16] introduced a fast quantum algorithm to find the prime factorization of any positive integer N.

There has been some research into analyzing and optimizing the exact costs of Shor's algorithm: in particular, the number of qubits required, and the number of qubit operations required. For example, a variant of Shor's algorithm by Beauregard [17] uses $O(n^3 \log n)$ operations on $2n + 3$ qubits if $N = pq$ fits into n bits. One can reduce the number of operations to $n^{2+o(1)}$, at some expense in the number of qubits. One can also run many of those operations in parallel.

Internally, Shor's algorithm evaluates a periodic function on a superposition of all inputs within a wide range; applies a quantum Fourier transform to obtain an approximate superposition of periods of the function; and measures this superposition to find a random period. The periodic function is $e \mapsto a^e \bmod N$, where a is a random integer coprime to N. If N is not a power of a prime (an easy case to recognize) then a random period reveals a factor of N with significant probability.

Shor introduced a similar algorithm to quickly find periods of the function $e, f \mapsto g^e h^f \bmod p$, revealing k such that $h = g^k \bmod p$. Replacing multiplication mod p with addition of points on an "elliptic curve" mod p breaks "elliptic-curve cryptography" [11, 12] (ECC), a popular alternative to RSA.

These algorithms, when applied to widely deployed public-key sizes for RSA and ECC, require billions of operations on thousands of logical qubits. Fault-tolerant attacks seem likely to require trillions of operations on millions of physical qubits. Perhaps quantum computing will encounter a fundamental obstacle that prevents it from ever scaling successfully to such sizes. However, no such obstacles have been identified. Prudent risk management requires defending against the possibility that these attacks will be successful.

4 Grover's algorithm

Many more cryptographic systems are affected by an algorithm that Grover [18] introduced in 1996. This algorithm is also the foundation for most, although not all, of the positive applications that have been identified for quantum computing.

Grover originally described his algorithm as searching an unordered database of size N using $\sqrt{N}$ quantum queries. This begs the question of why the database creator did not simply put the database into order, allowing it to be searched using $O(\log N)$ queries. A closer look at the details of Grover's algorithm also raises difficult questions regarding the physical cost of quantum database queries.

It is better to describe Grover's algorithm as searching for roots of a function f: i.e., searching for solutions x to the equation $f(x) = 0$. If one out of every N inputs is a root of f then Grover's algorithm finds a root using only about $\sqrt{N}$ quantum evaluations of f on superpositions of inputs. If f can be evaluated quickly *by a small circuit* then quantum evaluations of f do not use many qubit operations. This circuit condition often holds for the functions f that appear in cryptography.

Assume, for example, that a user is known to have encrypted 128-bit plaintexts "7" and "8" under a secret 128-bit AES key k, producing a 256-bit ciphertext $c = (\mathrm{AES}_k(7), \mathrm{AES}_k(8))$ visible to the attacker. Define $f(x) = (\mathrm{AES}_x(7), \mathrm{AES}_x(8)) - c$. This function f can be evaluated quickly (about 20000 bit operations) by a small circuit, and Grover's algorithm finds a root of f using only about 2^{64} quantum evaluations of f (overall [19] about 2^{86} "T-gates" and a similar number of "Clifford gates" applied to about 3000 qubits). Presumably this root is k: unless AES is deeply flawed, there will be at most a few pairs of distinct 128-bit keys x, k with collisions $(\mathrm{AES}_x(7), \mathrm{AES}_x(8)) = (\mathrm{AES}_k(7), \mathrm{AES}_k(8))$, and the user will not have selected one of those keys by chance.

Grover's speedup from N to $\sqrt{N}$ is not as dramatic as Shor's speedup. Furthermore, each of Grover's $\sqrt{N}$ quantum evaluations must wait for the previous evaluation to finish. If $\sqrt{N}$ exceeds the time T available then Grover's algorithm cannot use fewer than N/T evaluations spread across N/T^2 parallel quantum processors. This is a factor T better than pre-quantum techniques, but this improvement could be wiped out by quantum overhead: perhaps scalable quantum computers will be built and will run Shor's algorithm successfully, but Grover's algorithm will be useless.

On the other hand, if qubit operations are small enough and fast enough, then Grover's algorithm will threaten many cryptographic systems that aim for 2^{128} security, such as 128-bit AES keys. We recommend simply switching to 256-bit AES keys: the extra costs are rarely noticeable.

5 Post-quantum cryptography

Table 1 summarizes the effects of Shor's and Grover's algorithms upon typical cryptosystems. The table gives the impression that the advent of quantum computers destroys public-key cryptography, leaving only symmetric cryptography (with larger key sizes). Fortunately, RSA and ECC are not the only public-key systems.

We now review details of several proposals that have solidly resisted every suggested attack. In particular, nobody has been able to figure out any useful way to apply Shor's algorithm and its generalizations to these proposals.

Choosing secure key sizes for these proposals does require attention to Grover's algorithm, along with generalizations of Grover's algorithm such as quantum walks. Simply doubling the target security level is adequate but generally imposes much more noticeable costs upon public-key systems than upon AES; these costs motivate research aimed at understanding the exact impact of Grover's algorithm, so as to be able to use smaller key sizes.

This is not a comprehensive list of proposals and attack ideas. We do not describe isogeny-based cryptography [20, 21, 22], for example, and we do not discuss Kuperberg's algorithm [23]. Our list is biased towards proposals that have survived decades of study.

6 Code-based encryption

High-reliability computer equipment uses an “error-correcting code” to store 64 bits of logical data in 72 bits of physical memory. There is a 72×64 “generator matrix” G specifying each of the 72 physical bits as a sum, modulo 2, of some of the 64 logical bits. The code is $G\mathbf{F}_2^{64}$, a 64-dimensional subspace of the vector space $\mathbf{F}_2^{72}$, where $\mathbf{F}_2 = \{0, 1\}$. The code is designed so that any single error in the 72 bits (any change of a bit to its opposite) can be reliably corrected, and any double error (changing any two bits) can be reliably detected.

Error-correcting codes can be scaled up to correct more errors in longer blocks of data. They are used in a wide range of applications, including hard drives, satellite communication, and fault-tolerant quantum computation.

In 1978, early in the history of public-key cryptography, McEliece [24] proposed using a generator matrix as a public key, and encrypting a codeword (an element of the code) by adding a specified number of errors to it. McEliece’s code is secretly generated as a random “Goppa code” that can efficiently correct the specified number of errors, but this structure is not obvious from the generator matrix.

A simple but slow attack strategy against McEliece’s system is “information-set decoding” (ISD). An “information set” is a collection of codeword positions that determines the rest of the codeword. ISD guesses an information set, hoping that the ciphertext is error-free in those positions; uses linear algebra to (hopefully) determine the entire codeword; and checks that the ciphertext has the specified number of errors, in which case the codeword must be correct.

What makes ISD slow is that, for large matrices, the ciphertext is extremely unlikely to be error-free on any particular information set. More precisely, the number of guesses is $(c + o(1))^w$ where w is the number of errors added, $c > 1$ is a constant that depends on the selected ratio between the number of matrix rows and columns, and $o(1)$ means something that converges to 0 as $w \to \infty$.

Dozens of attack papers against McEliece’s system have found many improvements to ISD, but all of the pre-quantum attacks still take time $(c + o(1))^w$ for the same c, a remarkably stable track record. McEliece’s original key sizes were designed for 2^{64} security, and our successful attack against those key sizes thirty years later [25] took more than 2^{60} CPU cycles. All known algorithms to find the secret Goppa-code structure take even more time. The only post-quantum change in c has been a straightforward application [26] of Grover’s algorithm, replacing c with $\sqrt{c}$.

Rather than sending a message as a codeword, one should encrypt a random codeword, using a hash of the codeword as a secret key to authenticate and encrypt a message; this protects McEliece’s system against “chosen-ciphertext attacks” in which an active attacker sees the results of decrypting modified ciphertexts. Another improvement to McEliece’s system, due to Niederreiter [27], is to compress public keys to “systematic form”. When k bits are encoded as n bits, “systematic form” means that the first k physical bits are exactly the k logical bits, so the first $k \times k$ submatrix of the generator matrix is the identity matrix, which need not be transmitted. Yet another improvement,

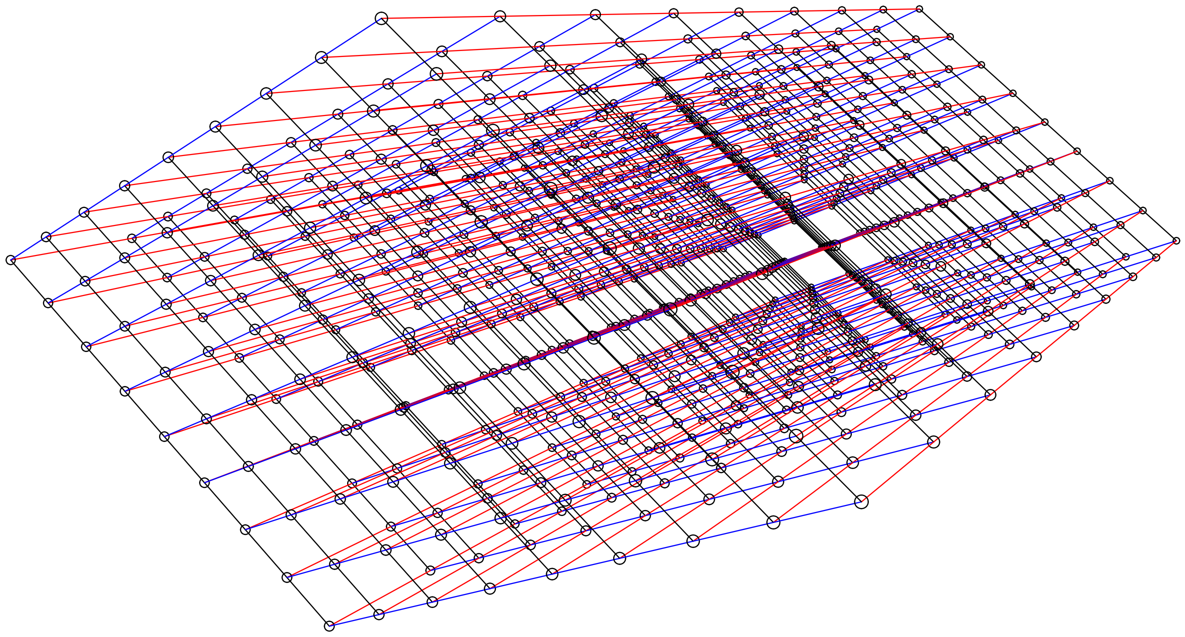
Figure 1: Perspective view of a $9 \times 9 \times 9$ subset of a non-orthogonal 3-dimensional lattice. Lattice-based cryptography hides a point in a high-dimensional lattice mod q by making small changes to all coordinates. Code-based cryptography hides a point in a very-high-dimensional lattice mod 2 by changing some coordinates.

also due to Neiderreiter, is to send “syndromes” rather than erroneous codewords; this reduces the ciphertext size to about 200 bytes at a high security level.

The main practical problem with these systems is the key size, roughly a megabyte (in systematic form) at a high security level. Many newer code-based systems put more structure into public keys to allow more compression, but some of those proposals have been broken. The only systems that have received enough study for us to recommend are the original McEliece/Niederreiter systems.

7 Lattice-based encryption

In the 1990s Hoffstein, Pipher, and Silverman [28] introduced an encryption system “NTRU” that has much smaller keys than McEliece’s system and that remains unbroken today. This system works as follows.

The public key is a p-coefficient polynomial $h = h_0 + h_1 x + \cdots + h_{p-1} x^{p-1}$, with each coefficient in the set $\{0, 1, \ldots, q-1\}$. A typical choice is $p = 743$ and $q = 2048 = 2^{11}$; then the public key has $743 \cdot 11 = 8173$ bits.

A ciphertext is another polynomial c in the same range. The sender chooses two secret polynomials d, e with small (say $-1, 0, 1$) coefficients, and computes $c = hd +$

$e \bmod x^p - 1 \bmod q$. The notation "mod $x^p - 1$" means that x^p is replaced by 1, x^{p+1} is replaced by x, etc., and the notation "mod q" means that each coefficient is replaced by its remainder upon division by q.

Define L as the set of pairs (u, v) of p-coefficient polynomials with integer coefficients such that $hu - v \bmod x^p - 1 \bmod q = 0$. Then L is a lattice in $2p$-dimensional space, and it contains a point close to $(0, c)$, namely $(d, c - e)$. The attacker's problem of finding the secrets d, e given the ciphertext c and public key h is thus an example of finding a lattice point close to a given point. This problem is analogous to the decoding problem for codes (see Figure 1), except that here "close" is interpreted as every coefficient being small, while codes simply count the number of nonzero coefficients.

NTRU, like McEliece's system, secretly generates the public key in a way that makes decoding efficient. Specifically, the receiver starts with a short vector of the form $(g, 3f)$, and uses a Euclidean algorithm to find h such that the lattice contains this vector, i.e., such that $hg - 3f \bmod x^p - 1 \bmod q = 0$. Then $cg \bmod x^p - 1 \bmod q$ is the same as $3df + eg \bmod x^p - 1 \bmod q$. An analysis of coefficient sizes shows that $3df + eg$ almost certainly has all coefficients strictly between $-q/2$ and $q/2$, and then it is an easy exercise to find (d, e) given f and g.

There are many potential attack avenues against NTRU and other lattice-based cryptosystems. For example, very recently the "cyclotomic" structure of $x^p - 1$ has been used to break [29, 30, 31] some lattice-based cryptosystems by an extension of Shor's algorithm. There are no known attacks exploiting cyclotomics *in NTRU*, but this attack avenue is new and has not been adequately explored. We recommend [32, 33] replacing $x^p - 1$ with $x^p - x - 1$, avoiding this structure. As another example, recent attacks [34, 35, 36] that work for arbitrary lattices, without exploiting any polynomial structure, have smaller exponents than the best such attacks known just a few years ago. Much more research is required to gain confidence in the security of lattice-based cryptography.

8 Lattice-based signatures

The first attempts [37, 38, 39] to turn hard lattice problems into signature systems were marred by attacks and surviving systems suffered from large signature sizes.

The most promising signature systems are based on Lyubashevsky's signature system [40] from 2012. Despite its evident youth we decided to include it because the resulting signatures are relatively short and fast to compute.

The system is most easily presented using integer matrices. Implementations typically use polynomial rings and fast Fourier transforms for compact representations and efficiency. Lyubashevsky's system uses several system parameters, namely integers k, m, n determining the sizes of matrices, κ limiting the Hamming weight of certain vectors, and q a modulus. Let A be an $n \times m$ integer matrix modulo q, i.e. $A \in \mathbf{Z}_q^{n \times m}$; this matrix may be shared by all users of the system but can also be chosen individually. The private key is a matrix $S \in \mathbf{Z}^{m \times k}$ with small entries, where small means significantly smaller than q, and often is restricted to $\{-1, 0, 1\}$. The public key is the $n \times k$ matrix $T = AS$, where the entries are computed modulo q. If A is not shared then it is also part of the public key.

The system uses a hash function $H : \mathbf{Z}_q^n \times \{-1, 0, 1\}^* \to \{0, 1\}^k$, where the output vectors additionally satisfy that no more than κ entries are non-zero. It is easy to build H from a traditional hash function h by encoding inputs and outputs appropriately.

The signer starts by picking y from an m-dimensional distribution, typically a discrete Gaussian distribution. (A discrete Gaussian distribution is a distribution obtained by considering only integer values of the regular Gaussian distribution and normalizing appropriately.) The signer then computes $c = H(Ay \bmod q, \mu)$, where μ is the message, and $z = Sc + y$. The signature is the pair (c, z). To avoid leaking information about the private key S through the distribution of (c, z), Lyubashevsky uses rejection sampling to force an S-independent distribution. This means that the process is restarted with probability depending on (c, z).

The signature is accepted as valid if c and z are sufficiently small and if $H(Az - Tc \bmod q, \mu) = c$. The latter holds for valid signatures because $Az - Tc \equiv A(Sc + y) - ASc \equiv Ay \bmod q$.

Later proposals such as BLISS [41] improve the running time by reducing the frequency of rejection in the last step. A ring version with $k = n$ and $m = 2n$ signs in under half a millisecond and verifies about 10 times faster. Public keys and signatures each are between 5 and 7 kilobits, not much larger than RSA signatures.

Ongoing challenges include (1) generating the distribution in a way that does not leak [42] information on S through side channels and (2) analyzing the security of the underlying algorithmic problem, namely the problem of finding short integer solutions to a system of equations modulo q.

9 Multivariate-quadratic-equation signatures

Matsumoto and Imai [43] introduced a new signature system "C*" in 1988. Patarin [44] broke the C* system in 1995 but the next year [45] introduced a stronger system "HFE$^{\text{v}-}$" that remains unbroken today.

The HFE$^{\text{v}-}$ public key is a sequence of polynomials $p_1, \ldots, p_m$ in the n-variable polynomial ring $\mathbf{F}_2[x_1, \ldots, x_n]$ over the field $\mathbf{F}_2$, with $m \le n$. The polynomials are limited to quadratics and have no squared terms: each polynomial p_i has the form $a_i + \sum_j b_{i,j} x_j + \sum_{j<k} c_{i,j,k} x_j x_k$ with $a_i, b_{i,j}, c_{i,j,k} \in \mathbf{F}_2$. The coefficients have no obvious public structure.

A signature of a message is an n-bit string $(s_1, \ldots, s_n) \in \mathbf{F}_2^n$ such that the m-bit string $(p_1(s_1, \ldots, s_n), \ldots, p_m(s_1, \ldots, s_n)) \in \mathbf{F}_2^m$ equals a standard m-bit hash $(h_1, \ldots, h_m)$ of the message. An example of a reasonable parameter choice (including the internal parameters v, q described below) is $(m, n, v, q) = (240, 272, 16, 2^{256})$; then a signature is just 34 bytes. These very short signatures are an attractive feature of this signature system.

The signer chooses the polynomials with a secret structure that allows the signer to solve the simultaneous quadratic equations $p_1(s_1, \ldots, s_n) = h_1, \ldots, p_m(s_1, \ldots, s_n) = h_m$.

Specifically, HFE$^{\mathrm{v}-}$ exploits the fact that there are general methods to solve *univariate* polynomial equations of degree d over finite fields $\mathbf{F}_q$ in time $(d \log q)^{O(1)}$. We now explain how the multivariate polynomials $p_1, \ldots, p_m$ are secretly related to a univariate polynomial.

The signer views an n-bit signature $(s_1, \ldots, s_n)$ as a randomly chosen v-bit string $(r_1, \ldots, r_v) \in \mathbf{F}_2^v$, where $v \le n - m$, together with an $(n-v)$-bit element $S \in \mathbf{F}_q$, where $q = 2^{n-v}$. This view is secret: v and q can be standardized, but before the signature is partitioned into (r, S) it is passed through a secret invertible $n \times n$ matrix chosen by the signer. This means that S is *some* linear function of $s_1, \ldots, s_n$, but not a public linear function.

The signer similarly views an m-bit hash value, together with a randomly chosen $(n - v - m)$-bit string, as an element $H \in \mathbf{F}_q$. Again this view is not standardized: it is obscured by another secret matrix. Here is the overall signing process, starting from the hash value: choose the $v + (n - v - m) = n - m$ random bits mentioned above; construct H; try to solve for S as explained below (or, if no solution S exists, start over with another choice of random bits); and construct the resulting n-bit signature.

There is one more secret: a degree-d polynomial $P \in \mathbf{F}_q[x, y_1, \ldots, y_v]$ of the form $A + \sum_j B_j x^{2^j} + \sum_{j>k} C_{j,k} x^{2^j+2^k} + \sum_j D_j y_j + \sum_{j,k} E_{j,k} y_j x^{2^k} + \sum_{j>k} F_{j,k} y_j y_k$. This polynomial specifies a secret equation connecting S and H, namely $P(S, r_1, \ldots, r_v) = H$. To convert this equation into the public quadratic polynomials, the signer writes each bit of S^{2^j} as a linear combination of $s_1, \ldots, s_n$. To solve the equation for any particular signature, the signer simply observes that this is a univariate equation in S for any particular choice of random bits $r_1, \ldots, r_v$.

For comparison, C* takes $q = 2^m = 2^n$; takes the polynomial P as a monomial x^{2^j+1} with exponent coprime to $2^q - 1$; and solves the equation $S^{2^j+1} = H$ by computing $S = H^e$, where e is the reciprocal of $2^j + 1$ modulo $2^q - 1$. The core idea of Patarin's attack is that the *bilinear* equation $S^{2^{2j}} H = H^{2^j} S$ is equivalent to a secret bilinear equation E on the bits of hashes and signatures. Each hash-signature pair produces a linear equation for the secret coefficients of E, and after enough signatures the attacker simply solves for those coefficients, at which point signature forgery is easy. HFE$^{\mathrm{v}-}$ blocks this attack by including more monomials in P.

There is a vast literature on other multivariate-quadratic signature systems and on algorithms to attack these systems. For HFE$^{\mathrm{v}-}$ in particular, all known attacks take time exponential in approximately $(n - m + \lceil \log_2 d \rceil) \log_2 n$. The same type of analysis used for recent pre-quantum parameter choices [46] indicates that $m = 240$, $n - v = 256$, and $n = 272$ provide high post-quantum security against known attacks even if d is quite small.

10 Hash-based signatures

One of the design goals mentioned earlier for hash functions is that finding a preimage for a given output string is computationally hard. In 1975 Lamport realized that this could be used to build a one-time signature system [47, 8].

Lamport's one-time signatures work as follows. To generate her key pair, Alice chooses two random strings x_0 and x_1; these constitute her secret key. Her public key is $(h(x_0), h(x_1))$, where h is a hash function which is known to everybody. If she wants to sign 0 she reveals x_0; the verifier recomputes $h(x_0)$ and checks the result against the first half of the public key. To sign 1, Alice reveals x_1. More generally, to sign an m-bit message, Alice takes $2m$ strings as the secret key $X = (x_{10}, x_{11}, x_{20}, x_{21}, \ldots, x_{m0}, x_{m1})$ and their hash values as public key $Y = (h(x_{10}), h(x_{11}), h(x_{20}), h(x_{21}), \ldots, h(x_{m0}), h(x_{m1}))$. The signature of, e.g., $10110\ldots$ is $(x_{11}, x_{20}, x_{31}, x_{41}, x_{50}, \ldots)$. Security rapidly degrades if Alice signs more than one message under the same key.

To overcome the problems of large public keys that could be used only once, Merkle proposed [48, 49] to combine 2^k public keys into one which can then be used to verify all 2^k signatures. For that, create 2^k key pairs for Lamport's one-time signature and arrange the public keys $Y_1, Y_2, \ldots, Y_{2^k}$ as the leaves of a binary tree with $k+1$ levels. A binary tree is one in which each node has exactly three edges, two going to a level closer to the leaves and one going closer to the root, except for the leaf nodes having only 1 and the root node having only 2 edges. Figure 2 shows an example of a Merkle tree with $2^3 = 8$ leaves. To compute the public key combining these 2^k keys, start from the leaves and compute the hash of each pair of public keys connected by edges in the tree, starting with $Y_{2^k+1} = h(Y_1, Y_2)$; continue iteratively through the levels, ending by computing the root $Y_{2^{k+1}-1} = h(Y_{2^{k+1}-3}, Y_{2^{k+1}-2})$. The value $Y_{2^{k+1}-1}$ at the root node is the public key of the system.

The public key is now a single hash value, but the signatures need to include more information to make it possible to check them. As before, a signature using secret key X_i reveals the x_{ij} matching the bit pattern of the message to be signed; in addition, the matching public key Y_i is included so that the Lamport signature can be verified. The signature also includes all siblings to the nodes encountered on the path from Y_i to the root; signature verification links Y_i to the public key by computing all hash values towards the root and comparing the value at the root with the public key.

Hash functions appear in all signature systems. Standard hash functions are affected only by Grover's attack, not by Shor's attack. This makes Merkle's very simple signatures prime candidates for post-quantum signatures: they have a clear security track record and computing hash functions is very fast.

Various improvements exist: using better one-time signatures [50, 51] to decrease the signature size, for example, or building trees of trees to reduce key-generation time. A system based on XMSS [52, 53] is currently in the final steps of adoption for Internet protocols by the Internet Research Task Force (IRTF). The U.S. National Institute for Standards and Technology (NIST) has indicated that they will fast-track a hash-based signature system.

It is important to never reuse a secret key X_i: each X_i is usable only one time. This means that the system described so far is *stateful*: the signer needs to remember which keys have been used. This might sound easy but has been described as a "huge foot-cannon" [54]: it poses problems for environments that use virtual machines, shared signing keys, etc. For such applications stateless systems exist [55] but this feature comes

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

Figure 2: Merkle tree with public key Y_{15} to sign 8 messages. The boxes show the verification path of the 6th signature, which reveals parts of X_6 (the secret key of Lamport's one-time signature), the matching public key Y_6, and includes Y_5, Y_{12}, and Y_{13} to compute the verification path to Y_{15}.

at the expense of longer signatures and longer signature-generation time.

11 Integration into the real world

Deploying a cryptographic system incurs physical costs: the time and energy consumed by cryptographic computations and by communication of keys, signatures, etc. Today's deployment of cryptography for billions of users relies on the fact that cryptography fits the users' budget. For comparison, some of the simplest goals of cryptography might also be achieved by couriers transporting locked briefcases, but this is so expensive that very few users can afford it.

Deploying a cryptographic system also raises questions of whether the real world matches the system's mathematical models of user capabilities and attacker capabilities. The most important examples are "side-channel attacks", in which the attacker learns extra information by observing timing [56], power consumption [57], etc. Another example is the stateful-signature-system "foot-cannon" mentioned above.

A large part of cryptographic research is aimed at finding the maximum real-world security achievable under various constraints on real-world costs. For example, side-channel attacks against cryptography are the largest topic at the immensely popular "Cryptographic Hardware and Embedded Systems" conference series, whereas there appears to have been far less public analysis of, e.g., the power of side-channel attacks against locked briefcases. As a final example, a state-of-the-art implementation [58] of McEliece's code-based system takes even less processing time than ECC; the only serious obstacle to wide deployment of this system is its key size.

12 Standardization

Several standardization bodies have recognized the urgency of switching to cryptosystems that remain secure against attacks by quantum computers. This is an important development because many applications of cryptography require all parties to use the same cryptographic system: standardization is thus a prerequisite for widespread deployment. Sometimes de-facto standards are set without standardization bodies, but formal standardization processes are widely viewed as reducing cryptographic risks.

The Internet Engineering Task Force (IETF) and its research branch IRTF are leading with having almost finalized standardization of a hash-based signature system. NIST has opened a call for submissions of candidates for standardization; the submission deadline is November 2017, and evaluation is expected to run 3–5 years. This call should result in the recommendation of a small portfolio of systems for encryption, signatures, and possibly other key-exchange mechanisms.

Other standardization bodies with post-quantum cryptography on the agenda are ETSI, with their "quantum-safe" working group; ISO, with SC27 WG2; and OASIS, with the KMIP standard.

One of the big European players in post-quantum cryptography is the EU-H2020 PQCRYPTO project. The logo of the project (see Figure 3) is illustrative of the state of post-quantum cryptography at the start of that project: confidence-inspiring proposals of long-lived systems are too big or too slow for casual deployment. The project covers research in design and implementation of new systems and analysis, including quantum cryptanalysis, of existing ones.

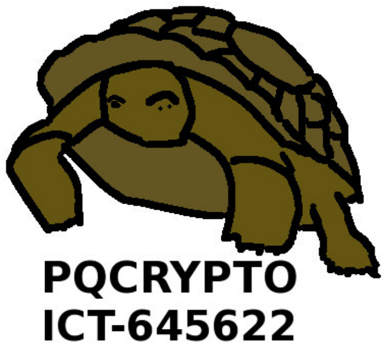

Figure 3: PQCRYPTO logo.

13 Ongoing and future work

These are exciting times for post-quantum cryptography. Researchers have identified many different ways to provide critical functions such as public-key encryption and public-key signatures. Some of these proposals have survived many years of scrutiny, but these proposals incur serious costs, especially in network traffic. Other proposals are more attractive for deployment, but their security is less clear, and it is likely that some of those proposals will be broken. Much more work is needed to unify these lines of research, building post-quantum systems that are widely deployable while at the same time inspiring confidence.

References

[1] Daemen, J. & Rijmen, V. *The Design of Rijndael: AES - The Advanced Encryption Standard.* Information Security and Cryptography (Springer, 2002). URL http:

//dx.doi.org/10.1007/978-3-662-04722-4.

[2] Bernstein, D. J. The Salsa20 family of stream ciphers. In Robshaw, M. J. B. & Billet, O. (eds.) *New Stream Cipher Designs - The eSTREAM Finalists*, vol. 4986 of *Lecture Notes in Computer Science*, 84–97 (Springer, 2008). URL http://dx.doi.org/10.1007/978-3-540-68351-3_8.

[3] McGrew, D. A. & Viega, J. The security and performance of the Galois/counter mode (GCM) of operation. In Canteaut, A. & Viswanathan, K. (eds.) *Progress in Cryptology - INDOCRYPT 2004, 5th International Conference on Cryptology in India, Chennai, India, December 20-22, 2004, Proceedings*, vol. 3348 of *Lecture Notes in Computer Science*, 343–355 (Springer, 2004). URL http://dx.doi.org/10.1007/978-3-540-30556-9_27.

[4] Bernstein, D. J. The Poly1305-AES message-authentication code. In Gilbert, H. & Handschuh, H. (eds.) *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*, vol. 3557 of *Lecture Notes in Computer Science*, 32–49 (Springer, 2005). URL http://dx.doi.org/10.1007/11502760_3.

[5] Information Technology Laboratory, National Institute of Standards and Technology. Secure hash standard (2012). URL http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf. Federal Information Processing Standards Publication 180-4.

[6] Bertoni, G., Daemen, J., Peeters, M. & Assche, G. V. Keccak. In Johansson, T. & Nguyen, P. Q. (eds.) *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, vol. 7881 of *Lecture Notes in Computer Science*, 313–314 (Springer, 2013). URL http://dx.doi.org/10.1007/978-3-642-38348-9_19.

[7] Rivest, R. L., Shamir, A. & Adleman, L. M. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **21**, 120–126 (1978). URL http://doi.acm.org/10.1145/359340.359342.

[8] Diffie, W. & Hellman, M. E. New directions in cryptography. *IEEE Trans. Information Theory* **22**, 644–654 (1976). URL http://dx.doi.org/10.1109/TIT.1976.1055638.

[9] ElGamal, T. A public key cryptosystem and a signature scheme based on discrete logarithms. In Blakley, G. R. & Chaum, D. (eds.) *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, vol. 196 of *Lecture Notes in Computer Science*, 10–18 (Springer, 1984). URL http://dx.doi.org/10.1007/3-540-39568-7_2.

[10] Schnorr, C.-P. Efficient identification and signatures for smart cards. In Brassard, G. (ed.) *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, vol. 435 of *Lecture Notes in Computer Science*, 239–252 (Springer, 1989). URL `http://dx.doi.org/10.1007/0-387-34805-0_22`.

[11] Miller, V. S. Use of elliptic curves in cryptography. In Williams, H. C. (ed.) *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*, vol. 218 of *Lecture Notes in Computer Science*, 417–426 (Springer, 1985). URL `http://dx.doi.org/10.1007/3-540-39799-X_31`.

[12] Koblitz, N. Elliptic curve cryptosystems. *Mathematics of Computation* **48**, 203–209 (1987).

[13] Bernstein, D. J. Curve25519: New Diffie-Hellman speed records. In Yung, M., Dodis, Y., Kiayias, A. & Malkin, T. (eds.) *Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography, New York, NY, USA, April 24-26, 2006, Proceedings*, vol. 3958 of *Lecture Notes in Computer Science*, 207–228 (Springer, 2006). URL `http://dx.doi.org/10.1007/11745853_14`.

[14] Johnson, D., Menezes, A. & Vanstone, S. A. The elliptic curve digital signature algorithm (ECDSA). *Int. J. Inf. Sec.* **1**, 36–63 (2001). URL `http://dx.doi.org/10.1007/s102070100002`.

[15] Bernstein, D. J., Duif, N., Lange, T., Schwabe, P. & Yang, B.-Y. High-speed high-security signatures. *J. Cryptographic Engineering* **2**, 77–89 (2012). URL `http://dx.doi.org/10.1007/s13389-012-0027-1`.

[16] Shor, P. W. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, 124–134 (IEEE Computer Society, 1994). URL `http://dx.doi.org/10.1109/SFCS.1994.365700`.

[17] Beauregard, S. Circuit for Shor's algorithm using $2n+3$ qubits. *Quantum Information & Computation* **3**, 175–185 (2003). URL `http://portal.acm.org/citation.cfm?id=2011525`.

[18] Grover, L. K. A fast quantum mechanical algorithm for database search. In Miller, G. L. (ed.) *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, 212–219 (ACM, 1996). URL `http://doi.acm.org/10.1145/237814.237866`.

[19] Grassl, M., Langenberg, B., Roetteler, M. & Steinwandt, R. Applying Grover's algorithm to AES: quantum resource estimates. In Takagi, T. (ed.) *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February*

24-26, 2016, Proceedings, vol. 9606 of *Lecture Notes in Computer Science*, 29–43 (Springer, 2016). URL http://dx.doi.org/10.1007/978-3-319-29360-8_3.

[20] Rostovtsev, A. & Stolbunov, A. Public-key cryptosystem based on isogenies (2006). URL https://eprint.iacr.org/2006/145.

[21] Couveignes, J.-M. Hard homogeneous spaces (2006). URL https://eprint.iacr.org/2006/291.

[22] Jao, D. & Feo, L. D. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In Yang, B.-Y. (ed.) *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings*, vol. 7071 of *Lecture Notes in Computer Science*, 19–34 (Springer, 2011). URL http://dx.doi.org/10.1007/978-3-642-25405-5_2.

[23] Kuperberg, G. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM J. Comput.* **35**, 170–188 (2005). URL http://dx.doi.org/10.1137/S0097539703436345.

[24] McEliece, R. J. A public-key cryptosystem based on algebraic coding theory (1978). URL http://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF. Deep Space Network progress report 42-44.

[25] Bernstein, D. J., Lange, T. & Peters, C. Attacking and defending the McEliece cryptosystem. In Buchmann, J. A. & Ding, J. (eds.) *Post-Quantum Cryptography, Second International Workshop, PQCrypto 2008, Cincinnati, OH, USA, October 17-19, 2008, Proceedings*, vol. 5299 of *Lecture Notes in Computer Science*, 31–46 (Springer, 2008). URL http://dx.doi.org/10.1007/978-3-540-88403-3_3.

[26] Bernstein, D. J. Grover vs. McEliece. In Sendrier, N. (ed.) *Post-Quantum Cryptography, Third International Workshop, PQCrypto 2010, Darmstadt, Germany, May 25-28, 2010. Proceedings*, vol. 6061 of *Lecture Notes in Computer Science*, 73–80 (Springer, 2010). URL http://dx.doi.org/10.1007/978-3-642-12929-2_6.

[27] Niederreiter, H. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory* **15**, 159–166 (1986).

[28] Hoffstein, J., Pipher, J. & Silverman, J. H. NTRU: A ring-based public key cryptosystem. In Buhler, J. (ed.) *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, vol. 1423 of *Lecture Notes in Computer Science*, 267–288 (Springer, 1998). URL http://dx.doi.org/10.1007/BFb0054868.

[29] Campbell, P., Groves, M. & Shepherd, D. Soliloquy: a cautionary tale (2014). URL http://docbox.etsi.org/Workshop/2014/201410_CRYPTO/S07_Systems_and_Attacks/S07_Groves_Annex.pdf.

[30] Biasse, J.-F. & Song, F. Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields. In Krauthgamer, R. (ed.) *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, 893–902 (SIAM, 2016). URL http://dx.doi.org/10.1137/1.9781611974331.ch64. **An extension of Shor's algorithm breaks some lattice-based systems.**

[31] Cramer, R., Ducas, L. & Wesolowski, B. Short Stickelberger class relations and application to Ideal-SVP (2016). URL https://eprint.iacr.org/2016/885.

[32] Bernstein, D. J. A subfield-logarithm attack against ideal lattices (2014). URL https://blog.cr.yp.to/20140213-ideal.html.

[33] Bernstein, D. J., Chuengsatiansup, C., Lange, T. & van Vredendaal, C. NTRU Prime (2016). URL https://eprint.iacr.org/2016/461.

[34] Laarhoven, T. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In Gennaro, R. & Robshaw, M. (eds.) *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, vol. 9215 of *Lecture Notes in Computer Science*, 3–22 (Springer, 2015). URL http://dx.doi.org/10.1007/978-3-662-47989-6_1.

[35] Laarhoven, T. & de Weger, B. Faster sieving for shortest lattice vectors using spherical locality-sensitive hashing. In Lauter, K. E. & Rodríguez-Henríquez, F. (eds.) *Progress in Cryptology - LATINCRYPT 2015 - 4th International Conference on Cryptology and Information Security in Latin America, Guadalajara, Mexico, August 23-26, 2015, Proceedings*, vol. 9230 of *Lecture Notes in Computer Science*, 101–118 (Springer, 2015). URL http://dx.doi.org/10.1007/978-3-319-22174-8_6.

[36] Becker, A., Ducas, L., Gama, N. & Laarhoven, T. New directions in nearest neighbor searching with applications to lattice sieving. In Krauthgamer, R. (ed.) *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, 10–24 (SIAM, 2016). URL http://dx.doi.org/10.1137/1.9781611974331.ch2.

[37] Goldreich, O., Goldwasser, S. & Halevi, S. Public-key cryptosystems from lattice reduction problems. In Kaliski Jr., B. S. (ed.) *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, vol. 1294 of *Lecture Notes in Computer Science*, 112–131 (Springer, 1997). URL http://dx.doi.org/10.1007/BFb0052231.

[38] Hoffstein, J., Pipher, J. & Silverman, J. H. NSS: an NTRU lattice-based signature scheme. In Pfitzmann, B. (ed.) *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, vol. 2045 of *Lecture Notes*

in Computer Science, 211–228 (Springer, 2001). URL http://dx.doi.org/10.1007/3-540-44987-6_14.

[39] Hoffstein, J., Howgrave-Graham, N., Pipher, J., Silverman, J. H. & Whyte, W. NTRUSIGN: digital signatures using the NTRU lattice. In Joye, M. (ed.) *Topics in Cryptology - CT-RSA 2003, The Cryptographers' Track at the RSA Conference 2003, San Francisco, CA, USA, April 13-17, 2003, Proceedings*, vol. 2612 of *Lecture Notes in Computer Science*, 122–140 (Springer, 2003). URL http://dx.doi.org/10.1007/3-540-36563-X_9.

[40] Lyubashevsky, V. Lattice signatures without trapdoors. In Pointcheval, D. & Johansson, T. (eds.) *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, vol. 7237 of *Lecture Notes in Computer Science*, 738–755 (Springer, 2012). URL http://dx.doi.org/10.1007/978-3-642-29011-4_43.

[41] Ducas, L., Durmus, A., Lepoint, T. & Lyubashevsky, V. Lattice signatures and bimodal Gaussians. In Canetti, R. & Garay, J. A. (eds.) *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, vol. 8042 of *Lecture Notes in Computer Science*, 40–56 (Springer, 2013). URL http://dx.doi.org/10.1007/978-3-642-40041-4_3.

[42] Groot Bruinderink, L., Hülsing, A., Lange, T. & Yarom, Y. Flush, Gauss, and reload - A cache attack on the BLISS lattice-based signature scheme. In Gierlichs, B. & Poschmann, A. Y. (eds.) *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, vol. 9813 of *Lecture Notes in Computer Science*, 323–345 (Springer, 2016). URL http://dx.doi.org/10.1007/978-3-662-53140-2_16. **First successful side-channel attacks against lattice-based signatures.**

[43] Matsumoto, T. & Imai, H. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In Günther, C. G. (ed.) *Advances in Cryptology - EUROCRYPT '88, Workshop on the Theory and Application of of Cryptographic Techniques, Davos, Switzerland, May 25-27, 1988, Proceedings*, vol. 330 of *Lecture Notes in Computer Science*, 419–453 (Springer, 1988). URL http://dx.doi.org/10.1007/3-540-45961-8_39.

[44] Patarin, J. Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt'88. In Coppersmith, D. (ed.) *Advances in Cryptology - CRYPTO '95, 15th Annual International Cryptology Conference, Santa Barbara, California, USA, August 27-31, 1995, Proceedings*, vol. 963 of *Lecture Notes in Computer Science*, 248–261 (Springer, 1995). URL http://dx.doi.org/10.1007/3-540-44750-4_20.

[45] Patarin, J. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): two new families of asymmetric algorithms. In Maurer, U. M. (ed.) *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, vol. 1070 of *Lecture Notes in Computer Science*, 33–48 (Springer, 1996). URL http://dx.doi.org/10.1007/3-540-68339-9_4.

[46] Petzoldt, A., Chen, M.-S., Yang, B.-Y., Tao, C. & Ding, J. Design principles for HFEv- based multivariate signature schemes. In Iwata, T. & Cheon, J. H. (eds.) *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part I*, vol. 9452 of *Lecture Notes in Computer Science*, 311–334 (Springer, 2015). URL http://dx.doi.org/10.1007/978-3-662-48797-6_14. **Optimizes conservative multivariate-quadratic signatures.**

[47] Lamport, L. Constructing digital signatures from a one way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory (1979). URL http://lamport.azurewebsites.net/pubs/pubs.html#dig-sig.

[48] Merkle, R. C. *Secrecy, authentication, and public key systems*. Ph.D. thesis, Stanford University (1979). http://www.merkle.com/papers/Thesis1979.pdf.

[49] Merkle, R. C. A certified digital signature. In Brassard, G. (ed.) *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, vol. 435 of *Lecture Notes in Computer Science*, 218–238 (Springer, 1989). URL http://dx.doi.org/10.1007/0-387-34805-0_21.

[50] Dods, C., Smart, N. P. & Stam, M. Hash based digital signature schemes. In Smart, N. P. (ed.) *Cryptography and Coding, 10th IMA International Conference, Cirencester, UK, December 19-21, 2005, Proceedings*, vol. 3796 of *Lecture Notes in Computer Science*, 96–115 (Springer, 2005). URL http://dx.doi.org/10.1007/11586821_8.

[51] Hülsing, A. W-OTS+ - shorter signatures for hash-based signature schemes. In Youssef, A., Nitaj, A. & Hassanien, A. E. (eds.) *Progress in Cryptology - AFRICACRYPT 2013, 6th International Conference on Cryptology in Africa, Cairo, Egypt, June 22-24, 2013. Proceedings*, vol. 7918 of *Lecture Notes in Computer Science*, 173–188 (Springer, 2013). URL http://dx.doi.org/10.1007/978-3-642-38553-7_10.

[52] Buchmann, J. A., Dahmen, E. & Hülsing, A. XMSS - A practical forward secure signature scheme based on minimal security assumptions. In Yang, B.-Y. (ed.) *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings*, vol. 7071 of *Lecture Notes*

in Computer Science, 117–129 (Springer, 2011). URL http://dx.doi.org/10.1007/978-3-642-25405-5_8. **Conservative stateful hash-based signatures are small and fast.**

[53] Hülsing, A., Rausch, L. & Buchmann, J. A. Optimal parameters for XMSS MT. In Cuzzocrea, A., Kittl, C., Simos, D. E., Weippl, E. R. & Xu, L. (eds.) *Security Engineering and Intelligence Informatics - CD-ARES 2013 Workshops: MoCrySEn and SeCIHD, Regensburg, Germany, September 2-6, 2013. Proceedings*, vol. 8128 of *Lecture Notes in Computer Science*, 194–208 (Springer, 2013). URL http://dx.doi.org/10.1007/978-3-642-40588-4_14.

[54] Langley, A. Hash based signatures (2013). URL https://www.imperialviolet.org/2013/07/18/hashsig.html.

[55] Bernstein, D. J., Hopwood, D., Hülsing, A., Lange, T., Niederhagen, R., Papachristodoulou, L., Schneider, M., Schwabe, P. & Wilcox-O'Hearn, Z. SPHINCS: practical stateless hash-based signatures. In Oswald, E. & Fischlin, M. (eds.) *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, vol. 9056 of *Lecture Notes in Computer Science*, 368–397 (Springer, 2015). URL http://dx.doi.org/10.1007/978-3-662-46800-5_15. **Conservative stateless hash-based signatures are practical.**

[56] Kocher, P. C. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Koblitz, N. (ed.) *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, vol. 1109 of *Lecture Notes in Computer Science*, 104–113 (Springer, 1996). URL http://dx.doi.org/10.1007/3-540-68697-5_9.

[57] Kocher, P. C., Jaffe, J. & Jun, B. Differential power analysis. In Wiener, M. J. (ed.) *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, vol. 1666 of *Lecture Notes in Computer Science*, 388–397 (Springer, 1999). URL http://dx.doi.org/10.1007/3-540-48405-1_25.

[58] Bernstein, D. J., Chou, T. & Schwabe, P. McBits: Fast constant-time code-based cryptography. In Bertoni, G. & Coron, J.-S. (eds.) *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, vol. 8086 of *Lecture Notes in Computer Science*, 250–272 (Springer, 2013). URL http://dx.doi.org/10.1007/978-3-642-40349-1_15. **Conservative code-based encryption is faster than ECC.**