# Introduction to post-quantum cryptography

Daniel J. Bernstein

Department of Computer Science, University of Illinois at Chicago.

## 1 Is cryptography dead?

Imagine that it's fifteen years from now and someone announces the successful construction of a large quantum computer. The *New York Times* runs a front-page article reporting that all of the public-key algorithms used to protect the Internet have been broken. Users panic. What exactly will happen to cryptography?

Perhaps, after seeing quantum computers destroy RSA and DSA and ECDSA, Internet users will leap to the conclusion that cryptography is dead; that there is no hope of scrambling information to make it incomprehensible to, and unforgeable by, attackers; that securely storing and communicating information means using expensive physical shields to prevent attackers from seeing the information—for example, hiding USB sticks inside a locked briefcase chained to a trusted courier's wrist.

A closer look reveals, however, that there is no justification for the leap from "quantum computers destroy RSA and DSA and ECDSA" to "quantum computers destroy cryptography." There are many important classes of cryptographic systems beyond RSA and DSA and ECDSA:

- **Hash-based cryptography.** The classic example is Merkle's hash-tree public-key signature system (1979), building upon a one-message-signature idea of Lamport and Diffie.
- **Code-based cryptography.** The classic example is McEliece's hidden-Goppa-code public-key encryption system (1978).
- **Lattice-based cryptography.** The example that has perhaps attracted the most interest, not the first example historically, is the Hoffstein–Pipher–Silverman "NTRU" public-key-encryption system (1998).
- **Multivariate-quadratic-equations cryptography.** One of many interesting examples is Patarin's "HFE$^{v-}$" public-key-signature system (1996), generalizing a proposal by Matsumoto and Imai.

- **Secret-key cryptography.** The leading example is the Daemen–Rijmen "Rijndael" cipher (1998), subsequently renamed "AES," the Advanced Encryption Standard.

All of these systems are believed to resist classical computers *and* quantum computers. Nobody has figured out a way to apply "Shor's algorithm"—the quantum-computer discrete-logarithm algorithm that breaks RSA and DSA and ECDSA—to any of these systems. Another quantum algorithm, "Grover's algorithm," does have some applications to these systems; but Grover's algorithm is not as shockingly fast as Shor's algorithm, and cryptographers can easily compensate for it by choosing somewhat larger key sizes.

Is there a better attack on these systems? Perhaps. This is a familiar risk in cryptography. This is why the community invests huge amounts of time and energy in cryptanalysis. Sometimes cryptanalysts find a devastating attack, demonstrating that a system is useless for cryptography; for example, every usable choice of parameters for the Merkle–Hellman knapsack public-key encryption system is easily breakable. Sometimes cryptanalysts find attacks that are not so devastating but that force larger key sizes. Sometimes cryptanalysts study systems for years without finding any improved attacks, and the cryptographic community begins to build confidence that the best possible attack has been found—or at least that real-world attackers will not be able to come up with anything better.

Consider, for example, the following three factorization attacks against RSA:

- 1978: The original paper by Rivest, Shamir, and Adleman mentioned a new algorithm, Schroeppel's "linear sieve," that factors any RSA modulus $n$—and thus breaks RSA—using $2^{(1+o(1))(\lg n)^{1/2}(\lg\lg n)^{1/2}}$ simple operations. Here $\lg = \log_2$. Forcing the linear sieve to use at least $2^b$ operations means choosing $n$ to have at least $(0.5 + o(1))b^2/\lg b$ bits.

  Warning: $0.5 + o(1)$ means something that *converges* to 0.5 as $b \to \infty$. It does not say anything about, e.g., $b = 128$. Figuring out the proper size of $n$ for $b = 128$ requires looking more closely at the speed of the linear sieve.

- 1988: Pollard introduced a new factorization algorithm, the "number-field sieve." This algorithm, as subsequently generalized by Buhler, Lenstra, and Pomerance, factors any RSA modulus $n$ using $2^{(1.9\ldots+o(1))(\lg n)^{1/3}(\lg\lg n)^{2/3}}$ simple operations. Forcing the number-field sieve to use at least $2^b$ operations means choosing $n$ to have at least $(0.016\ldots + o(1))b^3/(\lg b)^2$ bits.

  Today, twenty years later, the fastest known factorization algorithms for classical computers still use $2^{(\text{constant}+o(1))(\lg n)^{1/3}(\lg\lg n)^{2/3}}$ operations. There have been some improvements in the constant and in the details of the $o(1)$, but one might guess that $1/3$ is optimal, and that choosing $n$ to have roughly $b^3$ bits resists all possible attacks by classical computers.