Hash-Based Signatures: State of Play

Denis Butin | Technische Universität Darmstadt

Quantum computers haven't yet arrived, but a history of inertia in the wide-scale adoption of new cryptographic schemes means that standardization of postquantum signature schemes—particularly hash-based ones—is both timely and urgent.

Although numerous digital signature schemes have been put forward since the late 1970s, only a small subset are used on a large scale. Deployed systems rely mostly on RSA-based signature schemes or on the digital signature algorithm (DSA) and its variant, ECDSA, which uses elliptic curve cryptography. The ubiquity of this handful of schemes means that Internet authentication relies on a small number of security assumptions. More precisely, these classical schemes rely on the hardness of two number-theoretic problems: integer factorization and computation of discrete logarithms.

These two computation categories will be capable of being performed efficiently once quantum computing becomes practical, due to the applicability of Peter Shor's 1994 quantum algorithm.[1] In effect, these number-theoretic problems will no longer be hard—leading to a catastrophic loss of security. Consequently, currently deployed authentication systems aren't future proof.

The Need for Postquantum Digital Signatures

One might argue that quantum computing is still in its infancy and that a full-scale quantum computer capable of running Shor's algorithm is far off. Furthermore, the issue of quantum-safe authentication seems much less pressing than the question of quantum-safe encryption. Sensitive data encrypted with a classical encryption algorithm can be captured and stored by an adversary in advance for future decryption (once quantum computing becomes practical). In contrast, authentication is a more immediate matter and not vulnerable to preemptive attack.

Although the deployment of quantum-safe encryption is more urgent, postquantum signature schemes must nevertheless be tackled now. The reason is inertia: network security's history repeatedly shows that fundamental transitions occur slowly and that severely outdated systems often continue to run longer than they should. Witness current issues with the SSL security protocol's ongoing use, despite the fact that its replacement, TLS, was put forward in 1999.

Furthermore, stakeholders are often prepared to update their systems only if the new schemes under consideration are fully specified and standardized. Standardization is a time-consuming process. For postquantum cryptography (both signature and encryption schemes), it started only recently, with processes getting under way at the Internet Engineering Task Force

(IETF), the European Telecommunications Standards Institute, and NIST.

Finally, quantum computing might not be as far off as it seems. Previsions mentioning a mere 15-year horizon are common. And secret agencies might have already made technological breakthroughs unbeknownst to the public, resulting in an even shorter time frame.

In the coming years, organizations will make decisions about the suitability of postquantum signature schemes; draft standards are expected to be published by NIST around 2025. Therefore, a discussion of current options for postquantum signing is timely. In this article, I focus on hash-based signatures, which feature numerous variants and present the advantage of minimal security requirements.

A Panorama of Postquantum Signature Schemes

Although the threat of quantum computer–supported attacks is much more serious now than it was 40 years ago, some postquantum signature schemes find their roots in the 1970s. Furthermore, the past decade has seen a flurry of improvements to previously introduced schemes. Here, I summarize the key characteristics of and differences between lattice-based, multivariate polynomial–based, code-based, isogeny-based, and hash-based postquantum signature schemes. Hash-based schemes will be described much more comprehensively in the rest of the article.

Lattice-Based Signatures

Lattice-based cryptography is a well-established and very active research field. A great number of signature schemes based on lattices have been introduced. The security of lattice-based cryptography in general relies on the hardness of finding short vectors in high-dimensional lattices. For instance, in the case of the shortest-vector problem, one must find the shortest nonzero vector in a lattice for which a basis is provided. A distinguishing feature of lattice-based schemes is worst-case hardness; that is, they can usually be shown to be secure as long as at least one instantiation of the underlying computational problem is hard. By contrast, for non-lattice-based cryptography, security is guaranteed only as long as an average instantiation of the underlying problem is hard.

To improve efficiency, lattice-based signature schemes are often instantiated over ideal lattices—a subset of all lattices. Benefits with respect to conventional lattice-based schemes include smaller key sizes and improved running times. The tradeoff for these advantages is that the problem space—ideal lattices—is smaller than the normally used set of all lattices. Lattice-based signature schemes include ring-TESLA (Tightly Secure, Efficient Signature Scheme from Standard Lattices), an efficient scheme with a tight security proof.[2] A typical ring-TESLA parameter set for 128-bit security yields competitive sizes: public keys of 3 Kbytes, private keys of 2 Kbytes, and a signature of 2 Kbytes.

Multivariate Polynomial–Based Signatures

Multivariate polynomial–based signature systems offer the benefit of short signatures (normally less than 1 Kbyte) and are generally efficient. Based on simple arithmetic operations, they're fast with respect to other types of signing schemes. However, public keys tend to be comparatively large. This can be problematic for embedded devices. Their security is based on the hardness of solving sets of nonlinear equations over a finite field and relates to the structure of polynomial ideals.

As in the case of lattice-based signatures, numerous schemes exist. One example is the Rainbow signature scheme,[3] for which typical parameters for the underlying field F_{31}—targeting a security level adequate for use in the year 2020—yield a 45-Kbyte public key and a 31-Kbyte private key. A tradeoff between key pair generation time and size is available for Rainbow: keys and signatures can be smaller if one accepts slower key generation. The provable security of multivariate polynomial–based signature schemes isn't clear at the moment, resulting in unwelcome uncertainty.

Code-Based Signatures

The coding theory field was the basis for another category of postquantum signature schemes. Computational problems such as the syndrome-decoding problem are related to error-correcting codes used in electronic communication and storage. Code-based signature schemes are often constructed from identification schemes via a cryptographic technique called the Fiat-Shamir heuristic. Unlike digital signing, identification requires interaction between a prover and a verifier, not just a signer. For instance, the Stern identification protocol can be converted into a code-based signature scheme using Fiat-Shamir.[4] However, the resulting signatures are large at more than 100 Kbytes. A 2001 code-based signature scheme, CSF (for its authors Nicolas Courtois, Matthieu Finiasz, and Nicolas Sendrier), features very short signatures and efficient verification but huge public keys and inefficient signing. A code-based signing scheme suffering from none of these drawbacks has yet to be found.

Isogeny-Based Signatures

This year, Youngho Yoo and his colleagues introduced a digital signature scheme based on isogenies, a special kind of structure-preserving map.[5] In this signature scheme, a postquantum security level of 128 bits yields signatures of approximately 141 Kbytes and a 1-Kbyte public key. Concerning speed, signing is estimated to require approximately 3×10^{10} clock cycles on a desktop computer (that is, three orders of magnitude slower than hash-based signatures). Signing is therefore slow, but it can be readily parallelized and the costly part of the signing computations can be performed in advance. Unfortunately, verification speed is of the same order of magnitude.

Isogeny-based cryptography's security relies on the hardness of finding unknown isogenies between pairs of elliptic curves—a number-theoretic problem to which no efficient quantum algorithm is known to apply. The link between such schemes and elliptic curves might sound misleading, because classical cryptography based on elliptic curves relies on the hardness of discrete logarithm computation on elliptic curves, which isn't quantum safe. However, the computation of isogenies between a specific category of elliptic curves—supersingular—presents computation hardness that appears quantum safe.

The novelty of isogeny-based schemes incites caution, because they have not yet stood the test of time. On the other hand, the emergence of a new category of postquantum cryptographic schemes is a welcome diversification.

Hash-Based Signatures

Hash-based signatures have undergone numerous improvements since their introduction by Ralph Merkle in 1979.[6] The central idea is that many one-time signature key pairs are combined into a single structure using a hash tree. A hash tree is a hierarchical data structure that repeatedly uses a hash function and concatenation to compute nodes. Two key arguments support the use of hash-based signatures:

- *Minimal security assumptions.* Hash-based signature schemes such as the Extended Merkle Signature Scheme (XMSS) require only a secure cryptographic hash function to be secure. It's been proven that as long as any secure signature algorithm exists, a secure instantiation of XMSS exists.[7] In that sense, XMSS and other hash-based signature schemes display minimal security requirements.
- *Generic nature.* Hash-based signatures schemes can be instantiated with any hash function that fulfills simple criteria, resulting in tremendous flexibility. These schemes can be viewed as templates, with the underlying hash function as a piece that can be replaced without changing the overall structure. This is significant for long-term security, because new hash-function vulnerabilities might emerge over time.

Hash-Based Signature Basics

Here, I provide a high-level overview of hash-based signatures in terms of their building blocks, historical improvements, and overall structure. Detailed treatment can be found elsewhere.[8–14]

One-Time Signature Schemes

One-time signature schemes are the cornerstone of hash-based signatures. They can be seen as miniature digital signatures that can be used only once for a given key pair. Their security relies entirely and exclusively on the security of the underlying hash function. Although this underlying hash function entails certain technical assumptions, many hash functions fulfill these conditions, with many more expected to be developed in the future. Therefore, one-time signature schemes already display the template-like quality characteristic of hash-based signature schemes. Furthermore, as long as the chosen hash function fulfills the required criteria, the one-time signature scheme's structure remains identical.

The most commonly used one-time signature schemes are Leslie Lamport and Whitfield Diffie's, Robert Winternitz's, and W-OTS$^+$ (a Winternitz-type one-time signature scheme).[8] Because signatures actually reveal part of the private key, each can be used to sign only a single message with a given key pair. The Winternitz scheme and its variants are much more flexible than the seminal Lamport-Diffie scheme because they use a value—the Winternitz parameter—to choose the number of bits to be signed at once. Larger values result in shorter signatures and keys, but the tradeoff is speed: signing and verifying become slower. A typical value for this parameter is 16.

One-time signature schemes are structurally very simple. This simplicity is an advantage, because their security can readily be reduced to the security of the underlying hash function. Without going into the details of the corresponding security requirements, these requirements are unremarkable for usual hash functions.

An alternative to a one-time signature scheme as a building block to hash-based signature schemes is a few-time signature scheme. These schemes are used to sign a limited number of messages for a given key pair, but the limit is larger than one. A typical example of a few-time signature scheme is Hash to Obtain Random Subset (HORS), introduced in 2002.

The Merkle Scheme and Its Variants

Because one-time signature schemes require a new key pair for each new signature, they aren't practical for use by themselves. Instead, a large number of one-time signature key pairs are often combined into a single structure, with aggregated public and private keys constructed from these many key pairs. This is the gist of Merkle's signature scheme, introduced in the late 1970s,

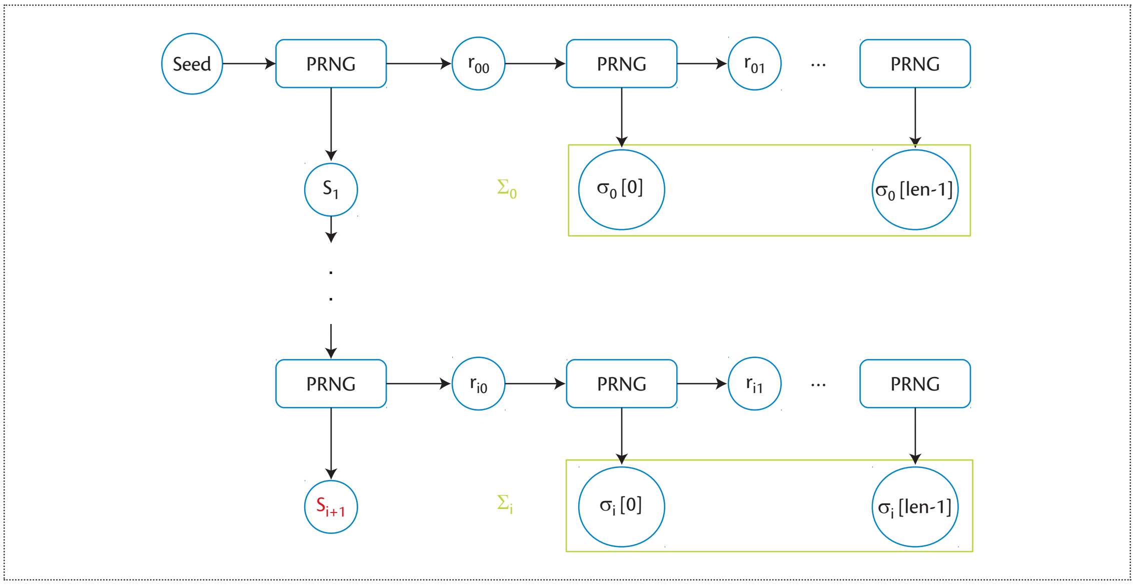

Figure 1. Iterative generation of successive pseudorandom number generator (PRNG) seeds (s_i) from an initial one (seed), and one-time signature signing key generation. The one-time signature signing keys are $\Sigma_i = \sigma_i$ [0]… σ_i [len – 1], $i \leq 2^h$, where len is the number of n-byte string elements in a W-OTS$^+$ secret key, public key, and signature. After the generation of the ith one-time signature signing key, the next seed s_{i+1} (in red) must be stored.

in which many one-time signature key pairs are combined into a binary tree structure. The way in which the tree's nodes relate to one other also involves a hash function. To this end, it makes sense to use the same hash function that was already selected as a basis for the underlying one-time signature scheme. This so-called Merkle tree, or hash tree, yields a small global public key, corresponding to the single tree node at the very top of the tree. This value is an output of the selected hash function; as a consequence, it's short. A typical public key length for a hash-based signature scheme is 32 bytes. To relate the validity of a one-time public key to that of the global public key, signatures store a sequence of tree nodes called the authentication path, allowing reconstruction of the path from the relevant one-time public key to the tree's top.

The global private key can be handled in different ways. The simplest is to concatenate all one-time private keys. Many one-time key pairs must be combined in a Merkle structure to result in a practical scheme, resulting in a very large global private key. An efficient alternative is to use a pseudorandom number generator. In this case, it suffices to store a seed value, and to successively derive one-time secret keys from this initial value using the generator (see Figure 1). Using this approach, the seed value to be stored is as small as the public key, 32 bytes in a typical situation.

In addition to decisive space savings, using a pseudorandom number generator provides the functional advantage of forward security. If a private key is compromised, no previous signatures can be forged, only upcoming ones.

Because each one-time private key can be used only once, it's critical for security to prevent multiple use of any of these keys. As a result, the Merkle scheme and its variants use one-time private keys sequentially and track the most recently used key. This index is stored as part of each signature, and also as part of the secret key. Thus, the secret key is dynamic; such schemes are said to be stateful. (The technical aspects of the management of this state are discussed elsewhere.[13])

Merkle's seminal hash-based signature scheme has undergone many improvements over the past decade. Although the issue of oversized private keys has been resolved through the pseudorandom number generator method, other efficiency aspects are more difficult to address and have thus been tackled incrementally. Most notably, the tree-traversal problem has proven central to performance considerations and has been the focus of increasingly complex approaches. Tree traversal is crucial because trees grow large when a high maximum number of signatures is required. This is the case for scenarios involving frequent signing, such as TLS. Generally speaking, securely generating up to 2^h signatures requires a total tree height h. Although some Merkle scheme variants use multiple, interdependent hash trees, this statement remains true with regard to the total height of the construction.

Some variants, such as XMSS^{MT},[10] use multiple tree layers. This makes it practical to combine a very large number (up to 2^{80}) of one-time key pairs in a single structure. In this case, only the lowest tree layer is used for actual message signing, with the other trees being used to sign the root values of trees on lower layers.

The Leighton-Micali Scheme

The Leighton-Micali signature scheme combines Merkle's general approach with a modified version of the Winternitz one-time signature scheme.[11] IETF is considering it for standardization.[12] The proposed standard includes a hierarchical variant.

SPHINCS

Hash-based signature schemes that don't require state tracking are also possible. The recent SPHINCS scheme is stateless and builds on a few-time signature scheme.[14] A hypertree (a tree of Merkle trees) is used, and on signing, the few-time key pair index is selected randomly, as opposed to sequentially. A typical parameter set results in a 1-Kbyte private key, a similarly sized public key, and a 41-Kbyte signature—which is significantly larger than the signatures of most stateful hash-based signature schemes. Performance is also worse for signature generation and verification. Consequently, the pros and cons of stateful and stateless hash-based signature schemes must be carefully weighed against use case requirements. Embedded systems, which feature limited resources, are better served by stateful schemes. But when performance isn't an issue, the advantages of statelessness are more profitable.

Minimal Security Assumptions

A hallmark of hash-based signatures is the minimal number of required security assumptions. Because all digital signature schemes sign digests (computed with a cryptographic hash function) of messages rather than full messages, the selected hash function's security is required for any digital signature scheme, independent of other potential requirements. In general, signature schemes also rely on other assumptions. For classical (prequantum) signature schemes, number-theoretic hardness assumptions are usually necessary. This isn't the case with hash-based signature schemes: no security assumptions are required other than the one for the hash function. This simplicity is very attractive, because the opposite situation implies whole classes of potential attacks—risks linked to additional security assumptions.

Challenges and Tradeoffs

Hash-based signatures bring with them several challenges. None are insurmountable, but they must be addressed carefully. Here, I take a closer look at three of these challenges. First, the issue of statefulness, already briefly mentioned, is discussed in more detail. Strategies for the adequate management of both state and stateless schemes are available, which entail benefits as well as drawbacks. Another nontrivial issue is the selection of parameters for different use cases. Although many prequantum signature schemes feature only a few parameters, this isn't so for modern hash-based signatures schemes. Especially if multiple trees are used, numerous parameter combinations are possible and users require guidance as to which parameter sets are best for the security protocol being considered. Last, I discuss the need for standardization and current steps in that direction.

Statefulness

The issue of statefulness results directly from the use of one-time signature key pairs as the building blocks of hash-based signatures. Security breaks down completely if a one-time signing key is used more than once, so tracking which one-time signing pairs were already used is critical for overall security. One-time signing keys are used sequentially, that is, in a predefined order, and an index or counter is stored in the global secret key to indicate which one-time signing keys can still be used. Size requirements for the index depend on the overall tree structure. As an example, for XMSS, a 4-byte value is sufficient. The index is part of the state that must be maintained; however, in general, the state contains more than just this index. In particular, hash-based signatures usually include an authentication path, which is a sequence of tree nodes. A verifier uses these nodes to reconstruct the path to the tree's root to validate a one-time public key against the overall public key. This authentication path must be updated with each signature, and elements related to it, such as precomputed nodes to equalize signing time, can be stored as part of the state. In this case, a performance benefit is derived.

David McGrew and his colleagues recently described strategies for state management.[13] One widely applicable method is state reservation. The underlying idea is that the global secret key is normally read from a hard disk (or another kind of nonvolatile storage) into RAM, so security is endangered if the updated private key isn't updated on the disk when it should be. To prevent this situation, several signatures are reserved for use in advance; that is, a private key with a number of signatures ahead of the current one is written to disk. This approach has the additional benefit of making disk access less frequent, improving overall signing performance as a result.

Another recently suggested mitigation strategy is a stateful–stateless hybrid approach.[13] It uses a hierarchical hash-based scheme with exactly one stateless level: the root level. The other (lower) levels are stateful (see Figure 2). This approach features increased

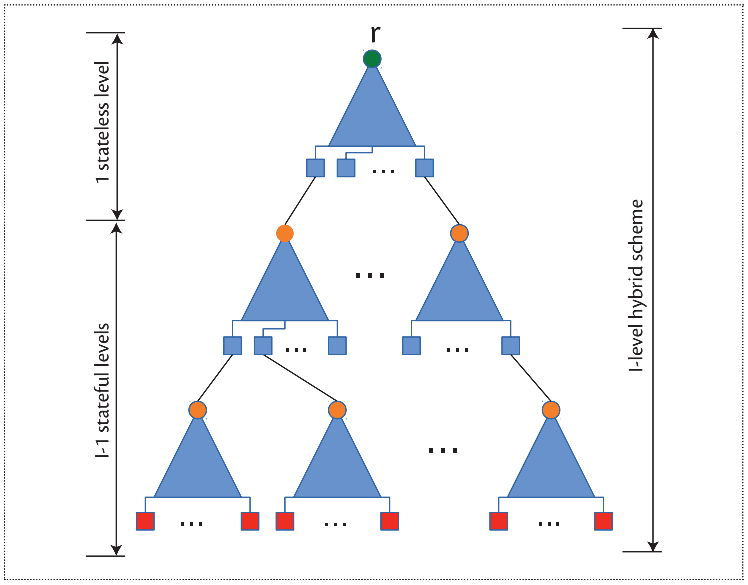

Figure 2. A hybrid approach that combines a stateless signature scheme at the root level and a stateful scheme, such as LMS (Leighton-Micali Scheme) or XMSS (Extended Merkle Signature Scheme), at the lower levels.

implementation complexity due to the two kinds of schemes used, but combines the functional advantages of statelessness and the performance benefits of stateful schemes. The total number of signatures that can be generated with this system remains limited, but no state synchronization is required at the root level, avoiding both the risks related to cloning and the performance penalty of synchronization delays.

Parameter Set Guidance for Specific Use Cases

Unlike many other types of digital signatures, hash-based signatures feature numerous parameters that enable tradeoffs. This makes their use by nonspecialists more complicated if no guidance is offered. No one parameter set can be recommended universally, because constraints on performance aspects such as signing speed and key size highly depend on use cases. For instance, software update authentication doesn't require high-frequency signing; the converse is true for HTTP over TLS. Individual users' email signing doesn't entail frequent signing either, but usability considerations might lead to the choice of a parameter set that prioritizes signature size to limit message expansion. In contrast, message expansion is bound to be negligible for HTTPS, considering the average size of contemporary webpages. Architectural complexities also vary considerably between use cases. HTTPS immediately raises the question of changes to current public-key infrastructure and certificate handling, while software update authentication entails more localized environmental modifications.

As a result, users would benefit from parameter set recommendations tailored to specific use cases. The specifications put forward for current standardization attempts suggest concrete parameter sets, but they don't address their adequacy for specific applications.[9,12] However, these specifications do discuss the crucial element of security levels for the proposed parameter sets. The choice of the underlying hash function is also normally not made explicit by hash-based signature schemes as described in the academic literature. Only security requirements on this underlying hash function are given. For concrete instantiations, guidance should be given in standards, with, for instance, SHA-256 specified as mandatory in current IETF Internet-Drafts.[9,12] In addition, state management strategies must be evaluated in a nuanced way depending on the intended use case. Complications arise mainly from high-volume signing and distributed environments.

Standardization

Standardized schemes enjoy broader adoption. (However, some popular cryptographic schemes and security protocols aren't standardized but are still widely used; take, for instance, the Noise Protocol used by a billion WhatsApp users despite only appearing in an informal specification.) Furthermore, the renewed scrutiny initiated by the standardization process leads to higher confidence in the claimed security properties and could bring about previously unexplored enhancements.

The standardization of hash-based signature schemes is desirable for these reasons, but it's also timely. In the US, federal organizations such as NSA and NIST are preparing the switch to postquantum cryptography. Calls for proposals are pending, such as a NIST competition currently under way. Such selection processes share many aspects of the standardization process, in particular the requirement for clear, down-to-earth specifications written in a much more concrete style than the academic articles that usually introduce new cryptography. Implementations following these specifications exactly are often required. Because these requirements also exist for standardization, it makes sense to combine the processes. In fact, NIST encourages this approach: in its postquantum cryptography competition information, it declares that stateful hash-based signature candidate schemes will be handled in coordination with an existing IETF standardization process.

Regarding the specification of hash-based signature schemes, two currently active IETF Internet-Drafts target the stateful schemes of XMSS and LMS (Leighton-Micali Scheme).[9,12] The Internet-Draft on XMSS uses W-OTS$^+$ as its basic building block. Both a single- (XMSS) and a multitree (XMSS^{MT}) scheme are described on top of W-OTS$^+$. Unlike the initial XMSS description in the academic literature, the Internet-Draft on XMSS includes a tweak domain separation that prevents multitarget attacks.[15]

The minimality of the required security assumptions is a strong argument in favor of hash-based signatures. This argument, as well as these schemes' template-like quality, make a compelling case for them to be part of the future portfolio of deployed postquantum signature schemes.

For smooth deployment, uphill work must be done now. Even though quantum computers aren't here just yet, the well-known inertia in the wide-scale adoption of new cryptographic schemes must be mitigated by advancing now on specification, standardization, and prototyping. The experience gained from small-scale deployments, particularly for state management strategies and parameter selections, will prove useful in the preparations for the larger transition.

Specification and standardization force cryptographers to come up with concrete parameter sets and foster a more focused discussion on topics such as adequate parameters for specific use cases. On top of the ongoing standardization of hash-based schemes themselves, extending security protocol standards, including the support of these schemes, is a desirable next step. Besides these practical considerations, specification encourages deeper analysis and public scrutiny of the proposed schemes, leading to higher confidence in their solidity. For this reason, all candidate schemes for postquantum signing, not just hash-based ones, would benefit greatly from standardization attempts. ■

References

1. P.W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM J. Computing*, vol. 26, no. 5, 1997, pp. 1484–1509.
2. S. Akleylek et al., "An Efficient Lattice-Based Signature Scheme with Provably Secure Instantiation," *Proc. 9th Int'l Conf. Cryptology in Africa* (AFRICACRYPT 16), LNCS 9646, Springer, 2016, pp. 44–60.
3. J. Ding and D. Schmidt, "Rainbow, a New Multivariable Polynomial Signature Scheme," *Proc. Applied Cryptography and Network Security* (ACNS 05), LNCS 3531, Springer, 2005, pp. 164–175.
4. P.-L. Cayrel, P. Gaborit, and E. Prouff, "Secure Implementation of the Stern Authentication and Signature Schemes," *Proc. IFIP WG 8.8/11.2 Int'l Conf. Smart Card Research and Advanced Applications* (CARDIS 08), LNCS 5189, Springer, 2008, pp. 191–205.
5. Y. Yoo et al., "A Post-Quantum Digital Signature Scheme Based on Supersingular Isogenies," Cryptology ePrint Archive, report 2017/186, 2017; eprint.iacr.org/2017/186.
6. R.C. Merkle, "A Certified Digital Signature," *Proc. 9th Ann. Int'l Cryptology Conf.* (CRYPTO 89), LNCS 435, Springer, 1989, pp. 218–238.
7. J. Buchmann, E. Dahmen, and A. Hülsing, "XMSS—A Practical Forward Secure Signature Scheme Based on Minimal Security Assumptions," *Proc. 4th Int'l Workshop Post-Quantum Cryptography* (PQCrypto 11), LNCS 7071, Springer, 2011, pp. 117–129.
8. A. Hülsing, "W-OTS$^+$—Shorter Signatures for Hash-Based Signature Schemes," *Proc. 6th Int'l Conf. Cryptology in Africa* (AFRICACRYPT 13), LNCS 7918, Springer, 2013, pp. 173–188.
9. A. Hülsing et al., "Internet-Draft: XMSS: Extended Hash-Based Signatures," Internet Engineering Task Force, 2017; datatracker.ietf.org/doc/draft-irtf-cfrg-xmss-hash-based-signatures.
10. A. Hülsing, L. Rausch, and J. Buchmann, "Optimal Parameters for XMSS^{MT}," *Proc. Int'l Conf. Availability, Reliability, and Security* (CD-ARES 13), LNCS 8128, Springer, 2013, pp. 194–208.
11. F.T. Leighton and S. Micali, "Large Provably Fast and Secure Digital Signature Schemes from Secure Hash Functions," US Patent 5,432,852, 1995.
12. D. McGrew, M. Curcio, and S. Fluhrer, "Internet-Draft: Hash-Based Signatures," Internet Engineering Task Force, 2017; datatracker.ietf.org/doc/draft-mcgrew-hash-sigs.
13. D. McGrew et al., "State Management for Hash-Based Signatures," *Proc. 3rd Int'l Conf. Security Standardization Research* (SSR 16), LNCS 10074, Springer, 2016, pp. 244–260.
14. D.J. Bernstein et al., "SPHINCS: Practical Stateless Hash-Based Signatures," *Proc. Int'l Conf. Theory and Applications of Cryptographic Techniques* (EUROCRYPT 15), LNCS 9056, Springer, 2015, pp. 368–397.
15. A. Hülsing, J. Rijneveld, and F. Song, "Mitigating Multi-target Attacks in Hash-Based Signatures," *Proc. 19th IACR Int'l Conf. Practice and Theory in Public-Key Cryptography* (PKC 16), LNCS 9614, Springer, 2016, pp. 387–416.

Denis Butin is a researcher in the Cryptography and Computer Algebra group at Technische Universität Darmstadt. His current research focuses on practical aspects of hash-based signatures. Contact him at dbutin@cdc.informatik.tu-darmstadt.de.