Quantum Resistant Public Key Cryptography: A Survey

Ray A. Perlner
ray.perlner@nist.gov

David A. Cooper
david.cooper@nist.gov

National Institute of Standards and Technology
100 Bureau Drive
Gaithersburg, Maryland 20899–8930

ABSTRACT

Public key cryptography is widely used to secure transactions over the Internet. However, advances in quantum computers threaten to undermine the security assumptions upon which currently used public key cryptographic algorithms are based. In this paper, we provide a survey of some of the public key cryptographic algorithms that have been developed that, while not currently in widespread use, are believed to be resistant to quantum computing based attacks and discuss some of the issues that protocol designers may need to consider if there is a need to deploy these algorithms at some point in the future.

Categories and Subject Descriptors

E.3 [**Data**]: Data Encryption—*Public key cryptosystems*

General Terms

Algorithms, Security

Keywords

Quantum computers, public key cryptography

1. INTRODUCTION

Since its invention, public key cryptography has evolved from a mathematical curiosity to an indispensable part of our IT infrastructure. It has been used to verify the authenticity of software and legal records, to protect financial transactions, and to protect the transactions of millions of Internet users on a daily basis.

Through most of its history, including present day, public key cryptography has been dominated by two major families of cryptographic primitives: primitives whose security is believed to be contingent on the difficulty of the integer factorization problem, such as RSA [46] and Rabin-Williams [44, 55], and primitives whose security is believed to be contingent on the difficulty of the discrete logarithm problem, such as the Diffie-Hellman key exchange [14], El Gamal signatures [19], and the Digital Signature Algorithm (DSA) [17]. Also included within the second family is elliptic curve cryptography (ECC) [32, 40], which includes all known, practical identity-based encryption schemes [5] as well as pairing-based short signatures [6].

While both the integer factorization problem and the general discrete logarithm problem are believed to be hard in classical computation models, it has been shown that neither problem is hard in the quantum computation model. It has been suggested by Feynman [16] and demonstrated by Deutsch and Jozsa [13] that certain computations can be physically realized by quantum mechanical systems with an exponentially lower time complexity than would be required in the classical model of computation. A scalable system capable of reliably performing the extra quantum operations necessary for these computations is known as a quantum computer.

The possibility of quantum computation became relevant to cryptography in 1994, when Shor demonstrated efficient quantum algorithms for factoring and the computation of discrete logarithms [51]. It has therefore become clear that a quantum computer would render all widely used public key cryptography insecure.

While Shor demonstrated that cryptographic algorithms whose security relies on the intractability of the integer factorization problem or the general discrete logarithm problem could be broken using quantum computers, more recent research has demonstrated the limitations of quantum computers [47]. While Grover developed a quantum search algorithm that provides a quadratic speedup relative to search algorithms designed for classical computers [24], Bennet, Bernstein, Brassard, and Vazirani demonstrated that quantum computers cannot provide an exponential speedup for search algorithms, suggesting that symmetric encryption algorithms, one-way functions, and cryptographic hash algorithms should be resistant to attacks based on quantum computing [4]. This research also demonstrates that it is unlikely that efficient quantum algorithms will be found for a class of problems, known as NP-hard problems, loosely related to both search problems and certain proposed cryptographic primitives discussed later in this paper.

The above research suggests that there is no reason, at the moment, to believe that current symmetric encryption and hash algorithms will need to be replaced in order to protect against quantum computing based attacks. Thus, any effort to ensure the future viability of cryptographic protocols in the presence of large scale quantum computers needs to concentrate on public key cryptography. Given how vital public key trust models are to the security architecture of today's Internet, it is imperative that we examine alternatives to the currently used public key cryptographic primitives.

IDtrust '09, April 14-16, 2009, Gaithersburg, MD
ACM 978-1-60558-474-4

In this paper, we provide an overview of some of the public key cryptographic algorithms that have been developed that are believed to be resistant to quantum computing based attacks. The purported quantum-resistance of these algorithms is based on the lack of any known attacks on the cryptographic primitives in question, or solutions to related problems, in the quantum computation model. This does not mean that an attack will never be found, but it does yield some confidence. The same type of argument is used to justify the security of all but a handful or cryptographic primitives in the classical computation model. One-time pads [50, 53] and universal hash functions [8] are unconditionally secure in any computation model, if used properly, but they are usually impractical to use in a way that doesn't invalidate the proof. Other cryptography often comes with a "security proof," but these proofs are generally based on at least one unproved security assumption—virtually any proof of security in the classical or quantum computation model not based on an unproved assumption would resolve one of the best known unsolved problems in all of mathematics [10].

Section 2 lists some of the issues that should be considered in comparing public key cryptographic algorithms. Section 3 describes a one-time signature scheme known as Lamport signatures, and Section 4 describes techniques that have been developed for creating long-term signature schemes from one-time signature schemes. Section 5 covers public key cryptographic algorithms based on lattices. Section 6 describes the McEliece signature and encryption schemes. Other potential areas of research are mentioned in Section 7 and Section 8 discusses issues that may need to be considered by protocol designers if one or more of the public key cryptographic algorithms described in this paper become widely used at some point in the future.

2. GENERAL CONCERNS

A number of factors can be considered when examining the practicality of a public key cryptographic algorithm. Among these are:

- Lengths of public keys, key exchange messages, and signatures: For public key cryptographic algorithms commonly in use today, these are all roughly the same size, ranging from a few hundred to a few thousand bits, depending on the algorithm. This is not always the case for candidate quantum-resistant algorithms. If public keys, key exchange messages, or signatures are much larger than a few thousand bits, problems can be created for devices that have limited memory or bandwidth.

- Private key lifetime: A transcript of signed messages often reveals information about the signer's private key. This effectively limits the number of messages that can safely be signed with the same key. The most extreme example of this is the Lamport signature scheme, discussed below, which requires a new key for each signed message. Methods have been developed for creating a long-term signature scheme from a short-term or even single-use signature scheme, but these often require extra memory for managing and storing temporary keys, and they tend to increase the effective length of signatures. Private keys used for decryption do not generally have limited lifetime, since encryption does not use and therefore cannot leak information about the private key, and protocols can almost always be designed to prevent the decryptor from revealing information about his or her private key. This can be done by encrypting symmetric keys rather than the content itself, using integrity protection, and reporting decryption failures in a way that makes them indistinguishable from message authentication code (MAC) failures. This type of behavior is currently necessary for secure protocols using old RSA padding schemes, and is often considered good practice regardless of the key transfer mechanism.

- Computational cost: There are four basic public key operations: encryption, decryption, signing, and signature verification. On today's platforms, with currently used algorithms, these operations generally take a few milliseconds, except for RSA encryption and signature verification, which can be about 100 times faster due to the use of small public exponents. Key generation time may also be a concern if it is significantly more expensive than the basic cryptographic operations. Factoring based schemes such as RSA and Rabin-Williams tend to have this problem, as generation of the two high entropy prime factors requires several seconds of computation.

3. LAMPORT SIGNATURES

The basic idea behind Lamport signatures [33] is fairly simple. However, there is a wide variety of performance tradeoffs and optimizations associated with it. It derives its security strength from the irreversibility of an arbitrary one-way function, f. f may be a cryptographic hash function, although the scheme is secure even if f is not collision resistant. The Lamport scheme is a one-time signature scheme. In order for the scheme to be secure, a new public key must be distributed for each signed message.

In the simplest variant of Lamport signatures, the signer generates two high-entropy secrets, $S_{0,k}$ and $S_{1,k}$, for each bit location, k, in the message digest that will be used for signatures. These secrets ($2n$ secrets are required if the digest is n bits long) comprise the private key. The public key consists of the images of the secrets under f, i.e., $f(S_{0,k})$ and $f(S_{1,k})$, concatenated together in a prescribed order (lexicographically by subscript for example). In order to sign a message, the signer reveals half of the secrets, chosen as follows: if bit k is a zero, the secret $S_{0,k}$ is revealed, and if it is one, $S_{1,k}$ is revealed. The revealed secrets, concatenated together, comprise the signature. While the act of signing a message clearly leaks information about the private key, it does not leak enough information to allow an attacker to sign additional messages with different digests. Nonetheless, there is no way in general for the signer to use this type of public key to safely sign more than one message.

While conceptually the simplest, the above scheme is not the most efficient way to create a one-time signature scheme from a one-way function [20]. Firstly, the size of public keys and signatures can be reduced by nearly a factor of two, merely by using a more efficient method of choosing which secrets to reveal from a smaller pool. For each bit location, k, rather than creating two secrets, $S_{0,k}$ and $S_{1,k}$, the secret key may consist of only $S_{0,k}$, with the public key being $f(S_{0,k})$. In order to sign a message, the signer would reveal

	Digest								Counter	
Digest	6	3	F	1	E	9	0	B	3	D
Signature	$f^6(S_0)$	$f^3(S_1)$		$f(S_3)$	$f^{14}(S_4)$	$f^9(S_5)$	S_6	$f^{11}(S_7)$	$f^3(S_8)$	$f^{13}(S_9)$
Public Key	$f^{15}(S_0)$	$f^{15}(S_1)$	$f^{15}(S_2)$	$f^{15}(S_3)$	$f^{15}(S_4)$	$f^{15}(S_5)$	$f^{15}(S_6)$	$f^{15}(S_7)$	$f^{15}(S_8)$	$f^{15}(S_9)$

Figure 1: A Sample Lamport Signature with $b = 16$

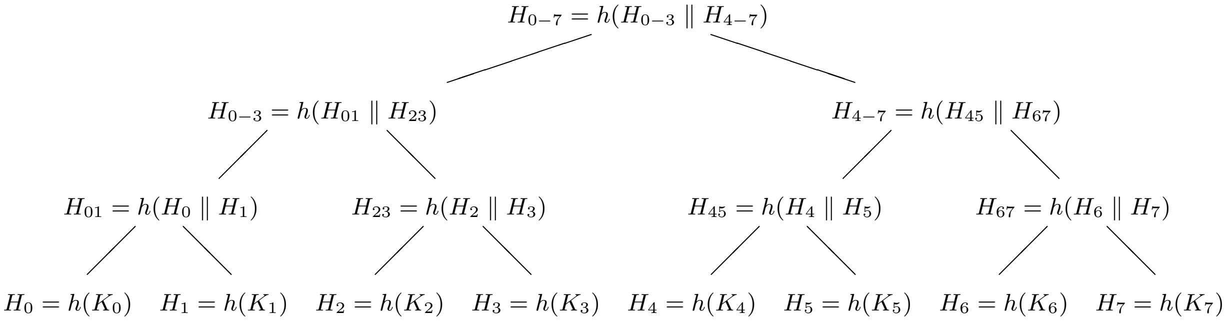
$H_{0-7} = h(H_{0-3} \parallel H_{4-7})$
$H_{0-3} = h(H_{01} \parallel H_{23})$
$H_{4-7} = h(H_{45} \parallel H_{67})$
$H_{01} = h(H_0 \parallel H_1)$
$H_{23} = h(H_2 \parallel H_3)$
$H_{45} = h(H_4 \parallel H_5)$
$H_{67} = h(H_6 \parallel H_7)$
$H_0 = h(K_0)$
$H_1 = h(K_1)$
$H_2 = h(K_2)$
$H_3 = h(K_3)$
$H_4 = h(K_4)$
$H_5 = h(K_5)$
$H_6 = h(K_6)$
$H_7 = h(K_7)$

Figure 2: Merkle Hash Tree

$S_{0,k}$ for each bit position, k, in the message digest that has a value of zero. Thus, the signature would be the concatenation of $S_{0,k}$ for each bit location in the message digest that has a value of zero. The problem with this scheme is that an attacker could try to change the value of a signature by withholding some of the $S_{0,k}$ values, thus changing some of the zero bits to one. In order to protect against this, a binary encoding of the total number of zero bits in the message digest may be appended to the message digest. This counter would be signed along with the message digest as described above. Since an attacker could only try to change zero bits to one, the attacker could not reduce the value of the counter, which would be necessary to successfully change some of the zero bits to one in the message digest itself.

The sizes of signatures and public keys can also be traded off against computation by using hash chains. In such a scheme, the message digests would be encoded using digits with a base b that is greater than two (e.g., using hexadecimal digits, which would correspond to $b = 16$). To sign the kth digit of the digest, N_k, the private key would be S_k, the public key would be the result of applying a one-way function, f, to the secret $b-1$ times, $f^{b-1}(S_k)$, and the signature value would be $f^{N_k}(S_k)$.[1] Thus if b were 4 and N_k were 1, then public key would be $f^3(S_k) = f(f(f(S_k)))$ and the signature value would be $f^1(S_k) = f(S_k)$. As with the binary scheme, there would be a need to append a "counter" to the message digest in order to prevent an attacker from increasing the values of any digits in the message digest. The value of the counter to be appended to the digest, for an n digit digest, would be $\sum_{k=0}^{n-1}(b-1-N_k)$. The reduction in signature size is logarithmic in the value of the base, while the cost of generating a one-time key pair is linear, so this process reaches diminishing returns fairly quickly, but using a base of 16 is often better than a base of 2. Figure 1 shows an example of a Lamport signature for a message digest that consists of eight hexadecimal digits.

Analysis of the performance of Lamport's one-time signatures is somewhat prone to confusion. As discussed above, the performance is dependent upon the choice of a one-way function and on the value of the base, b, used in generating the public key. Further, as the scheme is a one-time signature scheme the distinction between signing time and key generation time is not terribly useful, although it does provide a lot of opportunities for a signer to do precomputation. Nonetheless, with a fairly reasonable set of assumptions (e.g., f = SHA-256 with $b = 4$) one arrives at signature, verification, and key generation times that are similar to current schemes such as DSA.

4. LONG-TERM SIGNING KEYS FOR ONE-TIME SIGNATURE SCHEMES

If the signer can precompute a large number of single-use, public key - private key pairs, then at little additional cost, these keys can be used to generate signatures that can all be verified using the same public key [36]. Moreover, the long-term public key associated with this scheme need only be the size of a message digest. To do this, we use hash trees, a technique invented by Ralph Merkle in 1979 [35]. At the bottom of the tree, the one-time public keys are hashed once and then hashed together in pairs. Then those hash values are hashed together in pairs, and the resulting hash values are hashed together and so on, until all the public keys have been used to generate a single hash value, which will be used as the long-term public key. In this scheme, the signer can prove that a one-time public key was used in the computation that generated the long-term public key by providing just one hash value for each level of the tree—the overhead is therefore logarithmic in the number of leaves in the tree.

Figure 2 depicts a hash tree containing eight single-use public keys. The eight keys are each hashed to form the leaves of the tree, the eight leaf values are hashed in pairs to create the next level up in the tree. These four hash

[1] As with the binary scheme above, the signer would not need to reveal the signature value for any digit k for which $N_k = b - 1$.

values are again hashed in pairs to create H_{0-3} and H_{4-7}, which are hashed together to create the long-term public key, H_{0-7}. In order for an entity to verify a message signed using K_0, the signer would need to provide H_1, H_{23}, and H_{4-7} in addition to K_0 and a certified copy of H_{0-7}. The verifier would compute $H'_0 = h(K_0)$, $H'_{01} = h(H'_0 \parallel H_1)$, $H'_{0-3} = h(H'_{01} \parallel H_{23})$, and $H'_{0-7} = h(H'_{0-3} \parallel H_{4-7})$. If H'_{0-7} is the same as the certified copy of H_{0-7}, then K_0 may be used to verify the message signature.

While the the number of additional hashes that need to be added to a public key grows logarithmically with the number of leaves in the tree, the cost of generating a hash tree is linear in the number of leaves. It may therefore be desirable to limit the size of hash trees. If the signer wishes to use a single public key to sign more messages than the number of single-use key pairs he or she is willing to generate in the process of generating a public key, then the signer may wish to use a certificate chain like construction where the longest term public key is used to sign a large number of shorter-term keys, which in turn are used to sign even shorter term keys and so on. The advantage of this is that short-term keys can be generated as needed, allowing the cost of generating new one-time keys to be distributed over the lifetime of the single long-term key. This technique can also be used for other signature schemes where the key has limited lifetime, not just those that are based on hash trees. One example is NTRUSign, which is discussed later in this paper.

One important point to note is that unlike current signature schemes, this scheme is not stateless. The signer needs to keep track of more than just a single long-term private key in order to sign messages. If the signer is using hash trees, the signer can save a lot of memory by using a pseudorandom number generator to generate one-time private keys from a seed and a counter rather than saving all of the one-time private keys in memory. The one-time private keys are large and are only used twice: once for the purpose of generating the hash tree, and again when the one-time private keys are needed to sign messages, so this makes fairly good sense. The hashes in the tree, however, are used more often, and they should therefore be saved in memory. If these management techniques are used, then the footprint of a signing module does not suffer terribly from the short lifetime of the underlying signature scheme, but the dynamic nature of the stored information does imply that read-only or write-once memory cannot be used to store it.

5. LATTICE BASED CRYPTOGRAPHY AND NTRU

Unlike Lamport signatures, most public key cryptographic schemes derive their security from the difficulty of specific mathematical problems. Historically, factorization and the discrete logarithm problem have been by far the most productive in this respect, but as previously noted, these problems will not be difficult if full scale quantum computers are ever built. Therefore, cryptographers have been led to investigate other mathematical problems to see if they can be equally productive. Among these are lattice problems.

An n-dimensional lattice is the set of vectors that can be expressed as the sum of integer multiples of a specific set of n vectors, collectively called the basis of the lattice—note that there are an infinite number of different bases that will all generate the same lattice. Two NP-hard problems related to lattices are the shortest vector problem (SVP) [1] and the closest vector problem (CVP) [52]. Given an arbitrary basis for a lattice, SVP and CVP ask the solver to find the shortest vector in that lattice or to find the closest lattice vector to an arbitrary non-lattice vector. In both the quantum and classical computation models, these problems are believed to be hard for high dimensional lattices, containing a large number of vectors close in length to the shortest lattice vector.

Of the various lattice based cryptographic schemes that have been developed, the NTRU family of cryptographic algorithms [25, 26, 27] appears to be the most practical. It has seen some degree of commercial deployment and effort has been underway to produce a standards document in the IEEE P1363 working group. NTRU-based schemes use a specific class of lattices that have an extra symmetry. While in the most general case, lattice bases are represented by an $n \times n$ matrix, NTRU bases, due to their symmetry, can be represented by an $n/2$ dimensional polynomial whose coefficients are chosen from a field of order approximately n. This allows NTRU keys to be a few kilobits long rather than a few megabits. While providing a major performance advantage, the added symmetry does make the assumptions required for NTRU-based schemes to be secure somewhat less natural than they would otherwise be, and many in the theory community tend to prefer schemes whose security follows more directly from the assumption that lattice problems are hard. Such schemes include schemes by Ajtai and Dwork [2], Micciancio [39], and Regev [45].

In all NTRU-based schemes, the private key is a polynomial representing a lattice basis consisting of short vectors, while the public key is a polynomial representing a lattice basis consisting of longer vectors. A desirable feature of NTRU and other lattice based schemes is performance. At equivalent security strengths, schemes like NTRU tend to be 10 to 100 times faster than conventional public key cryptography, with cryptographic operations taking about 100 microseconds on contemporary computing platforms.

A number of minor attacks have been discovered against NTRUEncrypt throughout its 10+ year history, but it has for the most part remained unchanged. Improvements in lattice reduction techniques have resulted in a need to increase key sizes somewhat, but they have remained fairly stable since 2001. NTRUEncrypt has also been found to be vulnerable to chosen ciphertext attacks based on decryption failures [18, 21, 31, 38], but a padding scheme [30], which has provable security against these attacks, has been developed. In addition to security concerns, the recommended parameter sets for NTRUEncrypt have been changed for performance reasons. In one case, this was done over-aggressively and this resulted in a security vulnerability that reduced the security of one of the parameter sets from 80 bits to around 60 [29].

A comparatively greater number of problems have been found in NTRU-based signature schemes. The first NTRU-based signature scheme, NSS [28], was broken in 2001 by Gentry, Jonsson, Stern, and Szydlo a year after its publication [22]. A new scheme called NTRUSign [25] was introduced in 2002, based on the Goldreich-Goldwasser-Halevi signature scheme [23]. In this scheme, the signer maps the message digest to a vector, and proves knowledge of the private key by finding the nearest lattice point to that vector. Since the set of vectors to which a given lattice point is the nearest is non-spherical, it was known that a large number of messages signed with the same key would leak information about the private key. Because of this, the original signature scheme included an option, called perturbation, that would allow the signer to systematically choose a lattice point which was not necessarily the closest lattice point, but which was still closer than any point that could be found without knowledge of the private key. In 2006, it was shown by Nguyen that the unperturbed NTRUSign could be broken given only 400 signed messages [42]. The developers of NTRUSign estimate that with perturbation, it is safe to use the same NTRUSign key to sign at least one billion messages [54], but recommend rolling over to a new signing key after 10 million signatures [43].

6. MCELIECE

An additional hard problem that has been used to construct public key schemes is the syndrome decoding problem, which asks the solver to correct errors that have been introduced to an arbitrary, redundant linear transformation of a binary vector. There are, of course, easy instances of this problem, namely error correction codes, but in the general case, this problem is known to be NP-hard. One of the oldest of all public key cryptosystems, McEliece encryption [34], works by disguising an easy instance of the decoding problem as a hard instance. The security of McEliece therefore relies upon the presumed fact that it is difficult to distinguish between the disguised easy code and an arbitrary hard code.

The easy instance of the decoding problem used by McEliece is a family of error correction codes known as Goppa Codes. An (n, k) Goppa code takes a k-bit message to an n-bit code word in such a way that the original message can be reconstructed from any string that differs from the code word at fewer than $t = (n - k)/\log_2(n)$ bits. There are approximately n^t/t such codes. To disguise the code, it is written as an $n \times k$ matrix, then left-multiplied by an n-bit permutation matrix, and right multiplied by an arbitrary invertible binary matrix. The resulting $n \times k$ binary matrix is the public key, while the three matrices used to generate it remain private.

To encrypt a k-bit message, the encryptor treats the message as a binary vector, left-multiplies the public key, and randomly changes t of the resulting n bits. The private key holder can then decode the message stepwise. First the private key holder undoes the private permutation—this does not change the number of errors. The errors can now be corrected using the private Goppa code, allowing the private key holder to reconstruct the k-bit linear transformation of the original message. Since the private linear transformation used to construct the public key is invertible, the private key holder can now reconstruct the message.

McEliece has remained remarkably resistant to attack during its 30 year history, and it is very fast, requiring only a few microseconds for encryption and 100 microseconds for decryption on contemporary platforms. The primary drawback is that in order for the scheme to be secure, n and k need to be on the order of 1000, making the total size of the public key about a million bits.

It was recently demonstrated by Courtois, Finiasz, and Sendrier that there was a corresponding signature scheme [11], but this scheme is less desirable than the encryption scheme. To sign a message, the signer decrypts a string derived by padding the message digest. However, since most strings will not decrypt, the signer will typically have to try thousands of different paddings before finding a string that will decrypt. As a result, signing times are on the order of 10 to 30 seconds. It is, however, possible to make the signatures reasonably short.

7. OTHER AREAS OF RESEARCH

In addition to hash based signatures and lattice based and code based cryptography, a number of additional approaches have been used as an alternative basis for public key cryptography [7]. While most of the resulting schemes are currently poorly understood or have been broken, it is still possible that breakthroughs in these areas could one day lead to practical, secure, and quantum-resistant public key schemes.

One of the first NP-complete problems used in public key cryptography was the knapsack problem. Merkle and Hellman first proposed a knapsack based cryptosystem in 1978 [37], but this was soon shown to be vulnerable to approximate lattice reduction attacks [49]. Many similar schemes were subsequently broken, with the last, Chor-Rivest [9], being broken in 1995 [48].

More complex algebraic problems have also been proposed as successors to the factoring and discrete logarithm problems. These include the conjugacy search problem and related problems in braid groups, and the problem of solving multivariate systems of polynomials in finite fields. Both have been active areas of research in recent years in the mathematical and cryptographic communities. The latter problem was the basis for the SFLASH signature scheme [12], which was selected as a standard by the New European Schemes for Signatures, Integrity and Encryption (NESSIE) consortium in 2003 but was subsequently broken in 2007 [15]. It remains unclear when these or other algebraic problems will be well enough understood to produce practical public key cryptographic primitives with reliable security estimates.

8. CONSIDERATIONS FOR PROTOCOL DESIGNERS

In order to enable a comparison of the costs associated with various algorithms, Table 1 presents information about key sizes, message sizes, and the amount of time required to perform certain operations for several public key cryptographic algorithms. The table includes the algorithms that are described in this paper that are believed to be quantum resistant (Lamport signatures, McEliece encryption and signatures, NTRUEncrypt, and NTRUSign) as well as some of the public key cryptographic algorithms commonly in use today that are vulnerable to Shor's algorithm (RSA, DSA, Diffie-Hellman, and ECC). The numbers presented in the table are rough estimates, not benchmark results, but should be sufficiently accurate to enable comparison of the strengths and weaknesses of the different algorithms.

Compared to public key cryptographic algorithms commonly in use today, the algorithms presented in this paper differ in two ways that may be significant to protocol designers: key size and limited lifetime. Of the algorithms listed in Table 1, limited key lifetime is only an issue for Lamport signatures and NTRUSign. In the case of these two algorithms, the limited lifetimes should not pose significant

Table 1: A Comparison of Public Key Cryptographic Algorithms at the 80 Bit Security Level

	Estimated Time (PC)						
	Setup (ms)	Public Key Operation (ms)	Private Key Operation (ms)	Limited Lifetime?	Public Key Size (kbits)	Private Key Size (kbits)	Message Size (kbits)
Lamport Signature	1	1	1	1 signature	~10	~10	~10
Lamport w/Merkle	1	1	1	2^{40} signatures	0.08	~250	~50
McEliece Encryption	0.1	0.01	0.1	no	500	1000	1
McEliece Signature	0.1	0.01	20,000	no	4000	4000	0.16
NTRUEncrypt	0.1	0.1	0.1	no	2	2	2
NTRUSign	0.1	0.1	0.1	2^{30} signatures	2	2	4
RSA	2000	0.1	5	no	1	1	1
DSA	2	2	2	no	2	0.16	0.32
Diffie-Hellman	2	2	2	no	2	0.16	1
ECC	2	2	2	no	0.32	0.16	0.32

problems, but more consideration will need to be used in deploying these algorithms in order to ensure that keys are not used too many times.

When Lamport signatures are used in conjunction with Merkle hash trees as described in Section 4, the number of signatures that may be created from a given long-term public key is strictly limited, but that limit may be set to any value that the creator of the key chooses. If public keys have expiration dates, as they do today, then the maximum can always be set to a value that will ensure that the long-term public key will expire before all of the one-time keys have been used. Even a high volume server creating a few thousand signatures a second would take several years to create 2^{40} signatures. For most key holders, the maximum number of signatures per long-term public key could be set at a much smaller value, which would allow for smaller private keys and signatures.

The situation with NTRUSign is less clear since there is no fixed limit on the number of times that a key may be used. While the developers of NTRUSign recommend rolling over keys after 10 million signatures in order to be conservative, they believe that a key may be safely used to sign at least a billion messages [43]. For most key holders, even a limit of 10 million signatures would not be an issue. For some high volume servers, however, obtaining a new key pair and certificate after every 10 million signatures would be unreasonable, whereas a new certificate could be obtained after every billion signatures if the process were automated and relatively fast. If NTRUSign is to be used in the future, and further research indicates a need to impose key lifetimes that are closer to 10 million signatures than to 1 billion signatures, then high volume servers may need to employ one of the techniques described in Section 4 in order to reduce the frequency with which new certificates need to be obtained.

Table 1 shows the estimated key sizes that would be required to achieve 80-bits of security (i.e., a security level comparable to that provided by an 80-bit symmetric key). While 80-bits of security may be considered adequate at the moment, it is recommended that within the next few years all such keys be replaced with keys that provide 112 to 128 bits of security [3]. For the McEliece algorithms, this would imply 1 megabit public encryption keys and 8 megabit public signature keys. With key sizes this large, the ways in which public keys are distributed must be carefully considered.

With many protocols in use today, it is common to include a copy of the sender's certificate(s) in the message. For example, the server's encryption certificate is usually sent to the client during the key establishment phase of the Transport Layer Security (TLS) protocol. Also, email clients typically include copies of the sender's signature and encryption certificates in all digitally signed messages. Since most public key certificates that have been issued are less than 2 kilobytes, this is a reasonable practice at the moment, as the amount of bandwidth wasted by sending a copy of a certificate to a recipient that has previously received a copy is minimal. However, if the need to switch to quantum resistant algorithms were to lead to the use of public key cryptographic algorithms with key lengths comparable to those required by the McEliece signature and encryption schemes, this practice would need to be avoided and other means would need to be used to ensure that relying parties could obtain copies of the public keys that they need.

The most straightforward solution to this problem would be to avoid sending certificates in protocol messages, except in cases in which the recipient has requested a copy of the certificate. Instead, the protocol message could include a pointer to the certificate, which could be used by the recipient to obtain a copy of the certificate if it does not already have a copy in its local cache. For privacy reasons, many organizations prefer not to place end user certificates in publicly accessible directories. However, if the directories that hold certificates are not searchable and the URLs that point to the certificates are not easily guessable, this should provide an adequate amount of privacy protection.

An alternative solution would be to not include a copy of the public key in the certificate, but instead include a pointer to the public key along with a hash of the key. In this case, since the directory would only include the public key, there would be fewer privacy concerns with respect to the data in the directory. This would also allow the relying party to validate the certificate before downloading the public key, in which case the relying party could avoid the cost of downloading a very large public key if the certificate could not be validated, and thus the public key could not be used.

With very large public signature keys, the organization of public key infrastructures (PKI) would also need to be carefully considered. Today, even a very simple PKI may consist of a hierarchy of certification authorities (CA), with a root CA that issues certificates to subordinate CAs that in turn issue end user certificates. While the relying party would have already obtained the public key of the root CA through some secure, out-of-band means, the public key of one of the subordinate CAs would need to be downloaded in order to verify the signature on an end user certificate. If responses from Online Certificate Status Protocol (OCSP) [41] responders were needed to verify that neither the intermediate nor the end user certificate had been revoked, this could require the relying party to download two more public keys in order to verify the responses from the two OCSP responders. So, validating an end user certificate in a simple two-level hierarchy could require the relying party to download three public keys in addition to the end user's public key. In some PKIs today, certification paths involving four or more intermediate certificates are not uncommon. While this is reasonable with the public key algorithms that are in use today, which use public keys that are smaller than one kilobyte, such PKI architectures will need to be reconsidered if there is a need in the future to move to public key algorithms that require the use of very large public keys.

9. CONCLUSION

While factoring and discrete logarithm based cryptography continue to dominate the market, there are viable alternatives for both public key encryption and signatures that are not vulnerable to Shor's Algorithm. While this is no guarantee that they will remain impervious to classical or quantum attack, it is at least a strong indication. When compared to current schemes, these schemes often have similar or better computational performance, but usually require more bandwidth or memory. While this should not be a major problem for PCs, it may pose problems for more constrained devices. Some protocols may also have problems with increased packet sizes.

It does not appear inevitable that quantum computing will end cryptographic security as we know it. Quantum computing is, however, a major threat that we probably will need to deal with in the next few decades, and it would be unwise to be caught off guard when that happens. Protocol designers should be aware that changes in the underlying cryptography may and almost certainly will be necessary in the future, either due to quantum computing or other unforeseen advances in cryptanalysis, and they should be at least passably familiar with the algorithms that are most likely to replace current ones. Cryptanalysts will also need to scrutinize these algorithms before they are urgently needed. While some work has been done already, more work is needed to convince the cryptographic community that these algorithms will be as safe, in the future, as factoring and discrete logarithm based cryptography are today.

10. REFERENCES

[1] M. Ajtai. The shortest vector problem in L_2 is *NP*-hard for randomized reductions (extended abstract). In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing*, pages 10–19, 1998.

[2] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 284–293, 1997.

[3] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid. Recommendation for key management – part 1: General. NIST special publication 800-57, National Institute of Standards and Technology, Mar. 2007.

[4] C. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computation. *Special Issue on Quantum Computation of the Siam Journal of Computing*, Oct. 1997.

[5] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. of Computing*, 32(3):586–615, 2003.

[6] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In *Advances in Cryptology – ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security*, pages 514–532, 2001.

[7] J. Buchmann, C. Coronado, M. Döring, D. Engelbert, C. Ludwig, R. Overbeck, A. Schmidt, U. Vollmer, and R.-P. Weinmann. Post-quantum signatures. Cryptology ePrint Archive, Report 2004/297, 2004.

[8] J. L. Carter and M. N. Wegman. Universal classes of hash functions (extended abstract). In *STOC '77: Proceedings of the ninth annual ACM symposium on Theory of computing*, pages 106–112, 1977.

[9] B. Chor and R. L. Rivest. A knapsack type public key cryptosystem based on arithmetic in finite fields. *IEEE Transactions on Information Theory*, 34(5):901–909, Sept. 1988.

[10] S. Cook. The importance of the P versus NP question. *Journal of the ACM*, 50(1):27–29, 2003.

[11] N. Courtois, M. Finiasz, and N. Sendrier. How to achieve a McEliece-based digital signature scheme. In *Advances in Cryptology – ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security*, pages 157–174, 2001.

[12] N. T. Courtois, L. Goubin, and J. Patarin. SFLASH^{v3}, a fast asymmetric signature scheme. Cryptology ePrint Archive, Report 2003/211, 2003.

[13] D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. *Proc Roy Soc Lond A*, 439:553–558, Oct. 1992.

[14] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, Nov. 1976.

[15] V. Dubois, P.-A. Fouque, A. Shamir, and J. Stern. Practical cryptanalysis of SFLASH. In *Advances in Cryptology – CRYPTO 2007, 27th Annual International Cryptology Conference*, pages 1–12, 2007.

[16] R. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6&7):467–488, 1982.

[17] FIPS 186-2. *Digital Signature Standard (DSS)*.

National Institute of Standards and Technology, Jan. 2000.

[18] N. Gama and P. Q. Nguyen. New chosen-ciphertext attacks on NTRU. In *Public Key Cryptography - PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography*, pages 89–106, 2007.

[19] T. E. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology, Proceedings of CRYPTO '84*, pages 10–18, 1984.

[20] L. C. C. García. On the security and the efficiency of the Merkle signature scheme. Cryptology ePrint Archive, Report 2005/192, 2005.

[21] C. Gentry. Key recovery and message attacks on NTRU-composite. In *Advances in Cryptology – EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques*, pages 182–194, 2001.

[22] C. Gentry, J. Jonsson, J. Stern, and M. Szydlo. Cryptanalysis of the NTRU signature scheme (NSS) from Eurocrypt 2001. In *Advances in Cryptology – ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security*, pages 1–20, 2001.

[23] O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. In *Advances in Cryptology – CRYPTO '97, 17th Annual International Cryptology Conference*, pages 112–131, 1997.

[24] L. K. Grover. A fast quantum mechanical algorithm for database search. In *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[25] J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. H. Silverman, and W. Whyte. NTRUSign: Digital signatures using the NTRU lattice. In *Topics in Cryptology – CT-RSA 2003, The Cryptographers' Track at the RSA Conference 2003*, pages 122–140, 2003.

[26] J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. H. Silverman, and W. Whyte. NTRUEncrypt and NTRUSign: efficient public key algorithms for a post-quantum world. In *PQCrypto 2006: International Workshop on Post-Quantum Cryptography*, pages 141–158, May 2006.

[27] J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: A ring-based public key cryptosystem. In *Algorithmic Number Theory (ANTS-III): Proceedings of the Third International Symposium on Algorithmic Number Theory*, pages 267–288, June 1998.

[28] J. Hoffstein, J. Pipher, and J. H. Silverman. NSS: An NTRU lattice-based signature scheme. In *Advances in Cryptology – EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques*, pages 211–228, 2001.

[29] N. Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In *Advances in Cryptology – CRYPTO 2007, 27th Annual International Cryptology Conference*, pages 150–169, 2007.

[30] N. Howgrave-Graham, J. H. Silverman, A. Singer, and W. Whyte. NAEP: Provable security in the presence of decryption failures.

[31] É. Jaulmes and A. Joux. A chosen-ciphertext attack against NTRU. In *Advances in Cryptology – CRYPTO 2000, 20th Annual International Cryptology Conference*, pages 20–35, 2000.

[32] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[33] L. Lamport. Constructing digital signatures from a one-way function. Technical Report CSL-98, SRI International, Oct. 1979.

[34] R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. Deep Space Network Progress Report 42–44, Jet Propulsion Laboratory, California Institute of Technology, pages 114–116, 1978.

[35] R. C. Merkle. *Security, Authentication, and Public Key Systems*. PhD thesis, Stanford University, June 1979.

[36] R. C. Merkle. A certified digital signature. In *Advances in Cryptology – CRYPTO '89, 9th Annual International Cryptology Conference*, pages 218–238, 1989.

[37] R. C. Merkle and M. E. Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory*, 24(5):525–530, Sept. 1978.

[38] T. Meskanen and A. Renvall. A wrap error attack against NTRUEncrypt. *Discrete Applied Mathematics*, 154(2):382–391, Feb. 2006.

[39] D. Micciancio. Improving lattice based cryptosystems using the Hermite normal form. In *Cryptography and Lattices Conference — CaLC 2001*, pages 126–145, Mar. 2001.

[40] V. S. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology – CRYPTO '85*, pages 417–426, 1986.

[41] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP. RFC 2560 (Proposed Standard), June 1999.

[42] P. Q. Nguyen. A note on the security of NTRUsign. Cryptology ePrint Archive, Report 2006/387, 2006.

[43] NTRU Announces Signature Algorithm, NTRUSign, viewed November 12, 2008, ⟨http://www.ntru.com/cryptolab/intro_ntrusign.htm⟩.

[44] M. O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, MIT Laboratory for Computer Science, Jan. 1979.

[45] O. Regev. New lattice-based cryptographic constructions. *Journal of the ACM*, 51(6):899–942, Nov. 2004.

[46] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, Feb. 1978.

[47] S. Robinson. Emerging insights on limitations of quantum computing shape quest for fast algorithms. *SIAM News*, 36(1), January/February 2003.

[48] C.-P. Schnorr and H. H. Hörner. Attacking the Chor-Rivest cryptosystem by improved lattice

reduction. In *Advances in Cryptology – EUROCRYPT '95, International Conference on the Theory and Application of Cryptographic Techniques*, pages 1–12, 1995.

[49] A. Shamir. A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem. In *Advances in Cryptology: Proceedings of CRYPTO '82*, pages 279–288, 1982.

[50] C. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1949.

[51] P. W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[52] P. van Emde Boas. Another NP-complete problem and the complexity of computing short vectors in a lattice. Technical Report 81-04, University of Amsterdam, Department of Mathematics, Netherlands, 1981.

[53] G. S. Vernam. US patent #1,310,719: Secret signaling system, July 1919.

[54] W. Whyte. NTRUSign and P1363.1, Apr. 2006. http://grouper.ieee.org/groups/1363/WorkingGroup/presentations/P1363.1-2006-04.ppt.

[55] H. C. Williams. A modification of the RSA public-key encryption procedure. *IEEE Transactions on Information Theory*, IT-26(6):726–729, Nov. 1980.