Using Hash-Based Signatures to Bootstrap Quantum Key Distribution

Sébastien Kunz-Jacques\textsuperscript{1} and Paul Jouguet\textsuperscript{1,2}

\textsuperscript{1}SeQureNet, 23, avenue d’Italie, 75013 Paris
\textsuperscript{2}Telecom ParisTech, 46, rue Barrault, 75013 Paris

Abstract

Quantum Key Distribution is a secret distribution technique that requires an authenticated channel. This channel is usually created on top of an un-authenticated communication medium using unconditionally secure Message Authentication Codes (MAC) and an initial common secret. We examine the consequences of replacing this MAC algorithm by a cryptographic hash-based signature algorithm, like the Lamport algorithm, and show that in practical settings it results in an increase of the security of QKD and ease its deployment.

1 QKD, session authentication, and Digital Signatures

Quantum Key Distribution (QKD) is a way to create shared and secret random values at both ends of a communication link, with a security guaranteed without computational hardness assumptions \cite{SBpC+09}. It requires however a classical authenticated channel, together with an untrusted ‘quantum’ channel (usually realized with an optical fiber or an free space optical transmission). This authenticated channel can be realized on top of an un-authenticated network connection using cryptographic primitives. The natural choice for these primitives is to use symmetric, unconditionally secure Message Authentication Codes like Wegman-Carter \cite{WC81}, Evaluation Hash \cite{MV84} or LFSR-based Toeplitz \cite{Kra94}. Being symmetric, these primitives require a common secret; this is not a problem as soon as enough secret is created by the QKD link, but it is an undesirable constraint for the first run, as it forces the user to dispatch securely a common secret at both ends of the link. A very common argument against QKD is that, instead of exchanging a short common secret and using QKD to amplify it, one may as well exchange initially a very large secret and use it in place of the QKD output; the latter solution is easily realized thanks to the very low current price of storage. While this argument is not entirely correct\textsuperscript{1} it is desirable to have alternatives to the pre-sharing of a common secret.

Another argument against methods based on a common secret is that they are very hard to operate securely in practice. Indeed, the right way to implement them would be to store the secret on a device providing hardware security like a smart card (acting as a safe for the secret), but for this to be of interest the whole authentication tag computation needs to be performed inside the secure device. Unfortunately, the complete computation

\textsuperscript{1}Indeed, QKD is forward secure, which means that each key produced is completely independent of past values; as a consequence, even an attacker having at some point in time a complete knowledge of the equipment state including its secrets, does not learn anything about future keys in a passive attack scenario. Contrary to the hard disk scenario where a one-time compromise is enough to obtain all the keys, QKD forces the attacker to perform a persistent active attack in order to obtain new keys, with a much higher risk of being detected.
by a smart card of an authentication tag for a large set of messages corresponding to a
QKD protocol run, typically consisting of several megabits, is unpractical. The secret
must therefore be allowed to go out of the secure device; but then the very purpose of the
secure device is defeated.

Mitigation measures include enabling the secret to go out of the secure device only in a
trusted environment, with a mechanism to authenticate the latter to the secure device like
a pin code, or splitting the secret into several parts handled by independent parties, but
the overall security assurance provided by these techniques does not compare favorably to
the one offered by the resistance of cryptographic primitives, even computationally secure.

Asymmetric cryptographic primitives which are used to negotiate keys in classical
encryption protocols usually employ computationally secure authentication means that
are themselves asymmetric, that is, digital signature algorithms. A common combination
(normalized as ISO 9798-3 [ISO98]) is to use the Signed Diffie-Hellman algorithm, where a
Diffie-Hellman key exchange is authenticated with digital signatures. The QKD protocol
and the Diffie-Hellman algorithm are very similar in function, in that they both enable to
create common secret values if a way to authenticate messages is available, although the
security guarantees they provide differ. Digital signatures require the communicating par-
ties to exchange public keys in an authentic way, contrary to symmetric MAC algorithms
which require a common secret value. This is a huge improvement because it is much easi-
er to ensure that a value is authentic than it is to guarantee its secrecy. This is because
a message alteration can be uncovered anytime after it occurred, whereas preventing a
loss of secrecy requires the perfect continuity of the protecting measures. Together with
the invention of Public Key Infrastructures, this is what sparked the success of public-key
encryption.

Similarly to the case of Diffie-Hellman, it is appealing to use an asymmetric signature
algorithm to authenticate the first run of a pair of QKD equipments. Of course, doing
this makes the QKD security depend on the strength of the signing algorithm, which
reintroduces the very computational hardness assumptions QKD is supposed to be free
of.

In realistic deployments of QKD however, it will not be used stand-alone, encrypt-
ing traffic using one-time pad, and ensuring its integrity using unconditionally secure
Message Authentication Codes, but rather together with computationally secure symmetric
cryptography and authentication algorithms built on top of symmetric ciphers like the
Advanced Encryption Standard (AES) [FIP01]; in that setting, hardness assumptions
are required to ensure the confidentiality and integrity of the user traffic, and therefore
it makes sense to investigate the relationship between these assumptions and the ones
backing the security of asymmetric signature algorithms.

The existence of secure symmetric cryptography (stream ciphers, block ciphers, and
hash functions) is equivalent to the existence of one-way functions, that is, functions easy
to compute and hard to invert. Indeed, block ciphers are pseudorandom permutations, i.e.
permutations indexed by a key which are computationally indistinguishable from random
permutations when the key is secret; stream ciphers are pseudorandom number generators;
and hash functions are collision, 1st- and 2nd-preimage resistant functions. One-way

---

2 One could think of a 2-stage scheme where the secure device authenticates a small digest of the message,
but this can only be made secure in the computational sense: it must not be possible to find collisions for the
function that transforms the message into the digest, and such collisions exist since the digest is smaller than
the message.

3 When there are more than two users, the separation of the key in a public and a private part also reduces
the number of keys to distribute, since the same private-public key pair can be used to authenticate to several
parties, the public part being distributed to all of them. In the point-to-point setting of QKD however, we are
not concerned by this property.

4 a function is $n^{th}$-preimage resistant if it is difficult to compute a $n^{th}$ preimage of a value $x$ given $n-1$
different preimages of $x$. 

---

2
functions existence is known to be equivalent to the existence of pseudo-random number generators [HILL93], to the existence of pseudorandom functions [GGM86] and to the existence of pseudorandom permutations [LR88]. One-wayness is exactly 1\textsuperscript{st}-preimage resistance; it is implied by 2\textsuperscript{nd}-preimage resistance which is in turn implied by collision resistance; see [RS04]. There is no reverse implication, however a collision resistant, length-reducing function can be constructed from a one-way function [Rom90]. Hence 2\textsuperscript{nd}-preimage resistant or collision resistant functions exist iff one-way functions exist.

It is expected that one-way functions do exist, although this conjecture, implying the famous conjecture \( P \neq NP \), is not likely to be proven in the near future\(^5\). What happens in a world with quantum computers? Although some number-theory-based constructions used in asymmetric cryptography collapse, one-way functions may very well still exist. All we know is that with a quantum computer, key sizes for symmetric cryptography have to be doubled to retain the same security because of the Grover quantum algorithm enabling exhaustive search with square root complexity [Gro96].

Of course, the practical situation is more complex, since even if one-way functions and secure symmetric cryptography exist, it is not known whether the symmetric primitives used today are good approximations of their idealized counterparts. In fact, symmetric ciphers and cryptographic hash functions like the SHA family [FIP08], do not seem to rely on a small family of well-identified hypotheses of hardness of simple mathematical problems, unlike asymmetric algorithms.\(^6\) This lack of structure has two consequences: there is no provable security reduction between symmetric algorithms, but conversely their security is not likely to collapse because of some sudden theoretical advance. In fact, the last 30 years of cryptanalytic progress showed that the security of symmetric primitives of early designs like DES [FIP99] or hashing functions like SHA1 tend to erode slowly rather than abruptly, and that more mature designs (the AES competition contenders, the SHA2 family) exhibit a very good resistance to cryptanalysis.

It turns out that a family of signature algorithms, Lamport signatures [Lam79], and their derivatives, only require a function \( f \) with 1\textsuperscript{st}-preimage resistance (i.e. a one-way function) and a function with collision resistance \( g \) (which could be built from a one-way function as stated above\(^7\)). Let us sum up the properties of QKD when the public channel authentication is realized with the help of Lamport signatures. If the underlying functions \( f, g \) are indeed 1\textsuperscript{st}-preimage resistant and collision resistant, then naturally the combination has the same security properties than QKD with unconditional authentication methods. This includes unconditional forward secrecy. But the security properties of \( f, g \) need to last only as long as the (first) authenticated session itself: hence unlike the classical encryption setting where ciphertexts may be recorded for later decryption, only the attacks known at the moment of the QKD run, and that can be performed within the timeframe of a QKD session, are relevant.

In the rest of this paper, we investigate in details this solution, examine some variants and discuss the security properties obtained.

\(^5\)For a general presentation of these issues, see chapter 9 of [MvOV96], and in particular remark 9.12
\(^6\)For instance, the RSA hypothesis - related to, and not stronger than factoring - for RSA [RSA78], the discrete logarithm in finite fields or elliptic curves for DSA/ECDSA [FIP09, JM99] and Schnorr Signatures [Sch90], or related problems like the Computational Diffie-Hellman problem, etc.
\(^7\)This is why the existence of one-way functions implies the existence of digital signatures, which is the main result of [Rom90].
2 Lamport Signatures

2.1 Description

For this paragraph, the main reference is the chapter 3 of the book [BBD08].

Lamport signatures are \( n \times \ell \)-bit strings, where \( n \) and \( \ell \) are chosen according to the security requirements as explained in section 2.3.

As usual for digital signatures, a message to be signed is first transformed into a fixed-length string, its digest, by a collision-resistant hash function \( g : \{0,1\}^* \rightarrow \{0,1\}^\ell \). The rest of the algorithm uses a preimage-resistant function \( f : \{0,1\}^n \rightarrow \{0,1\}^n \).

The public key consists in \( 2\ell \) values \( y_i[j] = f(x_i[j]), i = 0, \ldots, \ell - 1, j = 0, 1 \). The values \( x_i[j] \) are the private key of the algorithm and are chosen uniformly at random.

The signature of a message \( M \) of digest \( m = g(M) = m_0, \ldots, m_{\ell-1} \) is the bit string \( x_0[m_0], x_1[m_1], \ldots, x_{n-1}[m_{\ell-1}] \). It is of size \( n \times \ell \).

The signature check of a signature \( s_0, \ldots, s_{\ell-1} \) is performed by verifying that \( f(s_i) = y_i[m_i] \) for \( i = 0, \ldots, \ell - 1 \).

2.2 The Case of QKD; one-time Signatures Usability

The Lamport algorithm is not widely used because a key pair can only sign one message. Indeed, its security degrades very quickly when several messages are signed with the same key pair: this is to be expected since a signature is really just a part of the private key. More precisely, given \( k \) signatures of messages whose digests are \( m'_i, i = 0, \ldots, \ell - 1, j = 0, \ldots, k - 1 \), a signature for any message of digest \( m'_0, \ldots, m'_{\ell-1} \) s.t.

\[
\forall i, m'_i \in \{m'_j \mid j = 0, \ldots, k - 1\}
\]

(1)

can be signed using the previous signatures. As soon as the signed messages digests differ on more that one bit (which occurs with overwhelming probability as soon as \( k \geq 2 \) since \( \ell \gg 1 \)), it suffices to create new combinations of the differing bits to create digests that can be signed and that are different from the original message digests.\[8\]

As we shall see in section 3.1 there are hash-based signature algorithms that are able to sign several messages with a unique set of keys. However, in the QKD setting, the limitation to one signature is not an issue for two reasons:

- The algorithm is only used to authenticate the first protocol run of a pair of QKD devices; subsequent executions are authenticated normally with a symmetric MAC algorithm using some of the common secret produced by the QKD link itself.
- To enable a recovery when this run was not successful, it is possible to include in the signed message a new public key that will be used to authenticate a new execution if needed. Additionally, a computational symmetric MAC can be used to check for message authenticity before using the Lamport mechanism: this will eliminate most failures before resorting to the signature and consuming the Lamport key, while not needing to put too much trust in the symmetric key used by the computational MAC mechanism.\[9\]

---

\[8\] available on-line at http://www.cdc.informatik.tu-darmstadt.de/~dahmen/papers/hashbasedcrypto.pdf

\[9\] for values of \( k \ll n \), it may still be hard to find a message whose digest lies in the set of digests that can be signed using the revealed part of the private key, i.e. messages satisfying equation (1), but this property holds only for very small values of \( k \) and cannot be used in practice.
2.3 Security Properties

The security of a slightly simplified Lamport signature scheme is stated in theorem 8.1 of [BRD08]. In this simplified scheme, messages are \( \ell \)-bit long and the transformation of a message \( m \) into its fixed-size digest \( g(m) \) is skipped. Therefore only the one-way function \( f \) plays a role in the security of the scheme. The stated security result is computational: it states there exists no attacker able with probability \( \epsilon \) to forge a Lamport signature for some public Lamport key \( PK \) and some new message \( m \) after it has obtained a signature for 1 message of its choice valid with respect to \( PK \), unless there also exists an attacker with the almost same running time and success probability \( \epsilon/4\ell \) against the underlying one-way function.

One says that, given a one-way function \( f \), the (modified) Lamport signature scheme for \( \ell \)-bit messages is \((1, \epsilon)\)-existentially unforgeable under an adaptive chosen message attack.

Going back to the unmodified scheme using a collision-resistant function \( g \), it can be proven that an attacker (1-adaptive existential forger as above) against the Lamport signature scheme cannot have success probability greater than \( \epsilon \) where there there exists an attacker against the one-wayness of \( f \) with success probability \( \epsilon'/4\ell \), and another attacker that produces a collision for \( g \) with probability \( \epsilon'' \) with

\[
\epsilon' + \epsilon'' \geq \epsilon
\]  

Indeed, if the attacker against the signature scheme produces a forged signature of a message \( m' \neq m \) after being given the signature of \( m \), either \( g(m) \neq g(m') \) which constitutes an attack against the simplified scheme, or \( g(m) = g(m') \) and \( (m, m') \) is a collision against \( g \). Remark that one could have \( \epsilon' = 0 \) or \( \epsilon'' = 0 \), as long as equation (2) is satisfied: an attacker against one of the two functions yields an attacker against the signature scheme.

The running time of the attacker against the Lamport signature signature scheme is the minimum of the running times of the attackers against \( f \) and \( g \).

Generic (non quantum) attacks against a hash function producing \( \ell \)-bit hashes enable to find collisions in \( O(2^{\ell/2}) \) hash function computations. For a \( n \)-bit hash function, preimages are found in a generic way in \( O(2^n) \) hash function computations. \( k \)-bit classical security (i.e. best non-quantum attack in \( O(2^k) \) operations) for Lamport signatures therefore requires \( n \geq k \) and \( \ell \geq 2k \). Typically we want 128-bit security, which yields \( n \geq 128 \) bits and \( \ell \geq 256 \) bits.

If quantum generic attacks are taken into account, the picture changes a bit. Finding preimages of an \( n \)-bit hash function can be performed in \( O(2^{n/2}) \) operations using Grover algorithm, and there is a quantum algorithm with complexity \( O(2^{\ell/3}) \) able to find with good probability a collision of \( \ell \)-bit hashes [BH1997]. However it requires a (quantum) memory of size \( O(2^{\ell/3}) \) [CIR94], so that we rather include its analysis to the next paragraph about parallel methods. With a constant or log amount of memory, the best known quantum attacks for \( n \)-bit hash preimage and \( \ell \)-bit hash collision have complexity \( O(2^{n/2}) \) and \( O(2^{\ell/2}) \), respectively. For \( k \)-bit security, one should therefore choose \( \ell, n \geq 2k \).

Assuming some parallelism, results are again different. With a (classical or quantum, computation or memory) resource size of \( O(2^\mu) \), and realistic communication models, the best known generic preimage complexity for a \( n \)-bit hash function is \( O(2^{(n-\mu)/2}) \) and the best collision attack for a \( \ell \)-bit hash function, \( O(2^{\ell/2-\mu}) \) [Ber09]. Hence one should choose \( n \geq 2k + \mu \) and \( \ell \geq 2(k + \mu) \). Assuming \( k = 128 \) and \( \mu = 64 \) (which is an extremely large security margin), this gives \( n \geq 320 \) bits and \( \ell \geq 384 \) bits.

The complexities above are given for an attacker with success probability one. Going back to the \( \epsilon', \epsilon'' \) above, the success probability of an attacker using only a fraction \( \eta \) of the resources indicated has a success probability \( \eta^2 \), except for the classical preimage search algorithm where the probability scales linearly with the effort.
2.4 Operation with a Secure Device

A secure device provides facilities to store data in a confidential and/or authentic way and to perform operations using this data. A typical cheap secure device is a smart card. It has a limited computational power, but is designed to store keys securely and enable to operate some cryptographic algorithms making use of these keys, typically a hashing algorithm (usually SHA1 or SHA2) and a block cipher algorithm (3DES or AES). More powerful secure devices ('Hardware Security Modules') can sit in computers but usually offer security assurances lower than smart cards.

The right way to implement a public-key algorithm such as Lamport is to never let private keys go out of secure devices. Therefore each secure device, at each end of the QKD link, generates its own private key and discloses the corresponding public key. During a trusted initialization phase, each device then receives the public key of the other device, or some digest of it, enabling it to later authenticate the signature of the other party.

Both the collision-resistant function and the preimage-resistant function are implemented with the hash function provided by the secure device. Since the first step in computing a signature (applying $g$) does not involve the private key, and results into a small-fixed-length string, this step can be performed outside the secure device. The hash is then provided to the secure device. With notations of section 2.1, the latter then picks the values $x_0[m_0], x_1[m_1], \ldots, x_{\ell-1}[m_{\ell-1}]$ corresponding to the digest $m_0, \ldots, m_\ell$ provided.

To ensure a correct usage of the algorithm as well as to effectively protect the private key, the sequencing of the instructions to the secure device should prevent multiple signing with the same private key. For instance, if the values $x_i[m_i]$ are output successively by the secure device, as it is the case with a smart card that cannot output the signature all-at-once, the private key should be erased as soon as the last signature part is output, or even better, $x_i[1-m_i]$ can be erased as soon as $m_i$ is known by the secure device.

2.5 Private and Public Key Size

An issue with the Lamport algorithm is its large key size: for instance, for $\ell = 256$ and $n = 128$, the public key size is $2 \times 256 \times 128 = 64$Kb. With the more conservative parameters of paragraph 2.3 the size becomes $2 \times 384 \times 320 = 240$Kb. Such a key cannot be stored on most smart cards.

A standard way to overcome this is to generate the private key from a pseudo-random number generator (PRNG) and to store only the secret $s$ of the PRNG; it is then possible to compute the public and the private key on-the-fly. The public key and signature are then computed and output piece by piece, typically one hash at a time.

Security wise, using this construction requires to take into account the PRNG security. It is measured by the success probability $p_{PRNG}$ of an attacker that is presented with either a true random sequence or an output of the PRNG and must distinguish between the two cases. With $\epsilon_{PRNG} = |2 \times p_{PRNG} - 1|$, the bound on the success probability of an attacker against the signature scheme using the PRNG becomes \cite{10}

\[
\epsilon' + \epsilon'' + \epsilon_{PRNG} \geq \epsilon
\]

The generic distinguishing attack consists in finding whether there is a value $s$ that enables to reproduce the output sequence. This is similar to the preimage attacks we discussed in paragraph 2.3 and the bounds given for $n$ apply for the size of $s$. Secure
PRNG constructions using hashing and satisfying these size constraints exist (see for instance the annex of [FIP09]).

3 Variants

3.1 Enabling Multiple Signatures: Merkle trees

Lamport signatures, or, for the matter, any one-time signature (OTS) scheme, can be paired with Merkle trees, which are a construction using only preimage-resistant functions and enabling the authentication of a large family of public signature keys with only one short value. We will omit a complete description of Merkle trees, which is detailed for instance in [BBD08], but will only describe their role and associated cost.

A Merkle tree of depth $H$ uses a collision-resistant function $T : \{0, 1\}^s \rightarrow \{0, 1\}^t$, and enables to perform $2^H$ signatures. The tree ‘public key’ is a $t$-bit value.

Each message signature consists in a signature by the underlying one-time signature scheme, with some added information value enabling to authenticate the OTS public key. This added information consists in $H$ $t$-bit values. The signature verification, aside from the underlying OTS signature verification, requires $H$ computations of $T$.

The signature algorithm calls the underlying OTS algorithm once and requires additionally $\mathcal{O}(H)$ computations of $T$ and the storage of $\mathcal{O}(H)$ values of $T$.

Initially, $2^H$ OTS public-private key pairs must be generated. This is usually too large to store; instead, a PRNG can be used, with ideas similar to those of paragraph 2.5, the computation of the tree public key requires $2^H - 1$ computations of $T$ and uses the $2^H$ underlying OTS public keys, which are themselves computed in the case of the Lamport algorithm through $2^H$ calls to $f$.

The success probability of an attacker against this scheme, attempting to forge a signature after requesting at most $2^H$ signatures, is bounded by $2^H \epsilon_{OTS} + \epsilon_{CR}$ where $\epsilon_{OTS}$ is the success probability of an attacker against the underlying OTS scheme, and $\epsilon_{CR}$ the success probability of an attacker against the collision resistance of $T$. The output size $t$ of $T$ should therefore be chosen as $\ell$ for the Lamport signature scheme. For a proof of this result and more discussion on the attackers running times, see theorem 8.2 of [BBD08].

In the usage context of quantum key distribution, the typical instantiation of Merkle trees would use a rather small parameter $H$ (say $H < 10$) to avoid sending a new public key in each signature, while retaining a small signature and verification overhead (this is especially true on the verification side where the Lamport verification step already requires $\ell$ hash computations). This also ensures the provable security loss factor incurred, $2^H$, is small.

3.2 Combination with Unconditional Methods

It is always possible to combine a computational signature mechanism with an unconditional MAC scheme: then the signature scheme is seen as a failsafe in case of a compromise of the common secret. Of course, this combination is computationally costly since it requires to process the messages to be authenticated once for each scheme.

4 Conclusion

We have seen that provided some minimal hypotheses are fulfilled, namely the existence of preimage-resistant and collision-resistant functions, some signatures schemes well known in classical cryptography are secure and can be used to authenticate communications on a classical channel without any common secret, and thereby to bootstrap key production on
a quantum key distribution link. In practical scenarios, collision-resistant and preimage-resistant functions are instantiated with a cryptographic hash function. We contend that the security gained by the removal of the initial common secret far outweighs the loss caused by the dependence to the preimage and collision resistance of the hash function used, notably since these properties are required to hold only until the initial QKD protocol run finishes, and not as long as the keys produced by the QKD link are supposed to remain secret.

We believe that there are other ways to use properties related to one-wayness, like pseudo-randomness, to improve the practicality and/or security of QKD protocols, without sacrificing its most fundamental property, the forward secrecy.

References

[BBD08] D. J. Bernstein, J. Buchmann, and E. Dahmen. Post Quantum Cryptography. Springer-Verlag, first edition, 2008.

[Ber09] D. J. Bernstein. Cost analysis of hash collisions: Will quantum computers make SHARCS obsolete? In Workshop Record of SHARCS'09: Special-purpose Hardware for Attacking Cryptographic Systems, 2009.

[BHT97] G. Brassard, P. Høyer, and A. Tapp. Quantum cryptanalysis of hash and claw-free functions. SIGACT News, 28:14–19, June 1997.

[FIP99] Data Encryption Standard (DES). FIPS PUB 46-3, National Institute for Standards and Technology, 1999.

[FIP01] Advanced Encryption Standard (AES). FIPS PUB 197, National Institute for Standards and Technology, 2001.

[FIP08] Secure Hash Standard (SHS). FIPS PUB 180-3, National Institute for Standards and Technology, 2008.

[FIP09] Digital Signature Standard (DSS). FIPS PUB 186-3, National Institute for Standards and Technology, 2009.

[GGM86] O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. Journal of the ACM, 33(4):210 – 217, 1986.

[GR04] L. Grover and T. Rudolph. How significant are the known collision and element distinctness quantum algorithms? QUANTUM INFORMATION AND COMPUTATION, 4:201, 2004.

[Gro96] L. K. Grover. A Fast Quantum Mechanical Algorithm for Database Search. In Annual ACM Symposium on Theory of Computing, pages 212–219. ACM, 1996.

[HILL93] J. Hästad, R. Impagliazzo, L. A. Levin, and M. Luby. Construction of a Pseudo-Random Generator From Any One-Way Function. SIAM Journal on Computing, 28:12–24, 1993.

[ISO98] ISO/IEC 9798-3: Entity Authentication Mechanisms - Part 3: Entity Authentication using Asymmetric Techniques, 1998.

[JM99] D. Johnson and A. Menezes. The Elliptic Curve Digital Signature Algorithm (ECDSA). Technical Report CORR 99-34, University of Waterloo, 1999.

[Kra94] H. Krawczyk. LFSR-based Hashing and Authentication. In Advances in Cryptology – Crypto’94, LNCS, pages 129–139. Springer-Verlag, 1994.

[Lam79] L. Lamport. Constructing Digital Signatures from a One Way Function. CSL-98, SRI International Computer Science Laboratory, 1979.
[LR88] M. Luby and C. Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. *SIAM Journal of Computing*, 17(2):373 – 386, 1988.

[MV84] K. Mehlhorn and U. Vishkin. Randomized and Deterministic Simulations of PRAMs by Parallel Machines with Restricted Granularity of Parallel Memories. *Acta Informatica*, 21:339–374, 1984.

[MvOV96] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. Available at http://www.cacr.math.uwaterloo.ca/hac/.

[Rom90] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, STOC ’90, pages 387–394, New York, NY, USA, 1990. ACM.

[RS04] P. Rogaway and T. Shrimpton. Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. In *FSE’04*, pages 371–388, 2004.

[RSA78] R. 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, 1978.

[SBpC+09] V. Scarani, H. Bechmann-pasquinucci, N. J. Cerf, M. Dušek, N. Lütkenhaus, and M. Peev. The security of practical quantum key distribution, 2009.

[Sch90] C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In B. Brassard, editor, *Advances in Cryptology – Crypto’89*, volume 435 of LNCS, pages 239 – 252. Springer-Verlag, 1990.

[WC81] M. N. Wegman and L. Carter. New Hash Functions and Their Use in Authentication and Set Equality. *Journal of Computer and System Sciences*, 22(3):265–279, June 1981.