Implementing a Lattice-Based Post-Quantum Authenticated Key Exchange

Pedro Miguel Sosa

May 5, 2017

Abstract

In the past few years of PQC research, most of the focus has been on key exchanges (KEX) and digital signatures (DS), while little work has been done for authenticated key exchanges (AKE). However, AKEs are an essential construct for secure Internet communication, so as we move towards PQC standardization, it is important not to disregard them. On this project we have implemented the first freely available lattice-based authenticated key exchange (AKE) based on the designs of Del Pino et al. [1]. Furthermore, we have provided an in-depth analysis of its computational and communicational complexities, as well as comparisons against the popular New Hope KEX [2]. It is our hope that this work may serve as a stepping stone for further research and development of efficient AKEs.

1 Introduction

As we approach the dawn of Quantum Computing, the cryptographic community has started to shift its focus towards Post-Quantum Cryptography (PQC). Several standardization bodies such as IRTF, NIST, ESI, and ISO have already started encouraging PQC research and standardizationn. In the US, NIST has began a call for proposals, which has fueled the development of a wealth of quantum-resistant key exchanges (KEX), digital signatures (DS), encryption schemes, and many other. [3]

Although authenticated key exchanges (AKE) are an essential building block for Internet security today, most researchers have focused uniquely on key exchanges and digital signatures. These researchers claim that one can build a generic AKE by simply combining a key exchange followed by a Digital Signature scheme. Del Pino, Lybashevsky, and Pointcheval [1] argue that by considering both secrecy and authenticity requirements of the AKE at the same time, one can construct a more secure and communicationally efficient scheme. Furthermore, they developed the mathematical foundation for an AKE, but without providing any implementation of it.

In this project we aimed to produce an open-source software implementation for their AKE. Aside from being the first Post-Quantum Lattice-Based AKE

implementation, it is our hope that this work may serve as a stepping stone for further research and development of efficient quantum-resistant AKEs.

1.1 Recent Work

Post-Quantum research has been very active in the last few years. Particularly in the area of Lattice-Based Cryptoraphy[1] which has shown a lot of promise.

Key Exchanges Research on Lattice-Based key exchanges has been highly active and produced multiple efficient schemes. Many companies are starting to adopt and test some these schemes in actual consumer services. For example, Google's popular Chrome browser actually implements some of these schemes for testing purposes [4]. Some historically popular schemes have been NTRU [5], LWE [6], and R-LWE [7]. However, New Hope, the latest scheme developed by Alkim et al. [2], represents one of the most efficient and fast PQC-KEX currently available.

Digital Signatures Research on Lattice-based digital signatures has been similarly active, however most still rely on very large key sizes. Some of the most popular Digital Signature schemes in the Lattice-Based field are NTRUSign [8], BLISS [9], and more recently, TESLA [10].

AKE's roots The AKE we will be discussing on this paper bases itself on two main works: Ducas' et al. Identity-Based Encryption scheme [11], and Gentry's et al. GPV algorithm implementation [12]. These are particularly important for the Digital Signature portion of the AKE.

1.2 Our Contribution

In this project, we have implemented the AKE proposed by Del Pino et al. [1] as a C++ code, using minimal library dependencies. This AKE provides a more secure and communicationally efficient counterpart to a generic AKE built by combining a KEX and a DS. To the best of our knowledge, our code represents the first Post-Quantum Lattice-Based AKE implementation available, and it can be freely obtained for public use or further research at: `github.com/pmsosa/PQC`.

Moreover, on section 3.1 we present an in-depth analysis of the computational and communicational complexities of our implementation and compare it against the popular New Hope KEX [2].

[1] PQC can be subdivided into 5 main fields: Hash-Based, Multivariate-Based, Code-Based, Lattice-Based, and Supersingular Isogenies-Based.

2 AKE Scheme

On this section we will concisely explain the AKE developed by Del Pino et al. [1]. This AKE consists of two sections: A Key Encapsulation Mechanism (KEM) and a Digital Signature (DS) scheme.

2.1 Key Encapsulation Mechanism

The author's KEM is based on the hardness of the NTRU problem, in which we attempt to find short solutions to polynomial equations over certain rings.

For this KEM, Assume the mathematical operations belong in the ring $R = \mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$ where $q = 12289$ and $n = 2^i$ (Usually $n = 512$ or $n = 1024$). This means that our elements will be represented as polynomials of degree at most n-1 and with reduction modulo an odd q that will map the coefficients into the range $[-(q-1)/2, (q-1)/2]$.

Furthermore, we will define two subsets of R: D_f and D_e. The distributions of D_e and D_f will depend on the security and failure probability we want to achieve. Namely, the l_2-norm of the elements in these distributions need to follow the parameters described in Section 2.3.1. Additionally, it is required that elements in D_f be invertible in both $R_q = \mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$ and $R_2 = \mathbb{Z}_2[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$.

$$\begin{aligned}
&\text{KEMKeyGen}\{\ f, g \xleftarrow{\$} D_f \\
&\qquad\qquad h \leftarrow f/g \bmod q \\
&\qquad\qquad \text{Return}(K_d, K_e) = (g, h) \qquad\qquad \}
\end{aligned}$$

$$\begin{aligned}
&\text{Enc}(K_e)\{\ r, e \xleftarrow{\$} D_e \\
&\qquad\qquad t \leftarrow 2hr + e \bmod q \\
&\qquad\qquad \text{Return}(c, k) = (t, e \bmod q) \qquad\qquad \}
\end{aligned}$$

$$\text{Dec}(K_d, c)\{\ \text{Return } k = \frac{gt \bmod q \bmod 2}{g} \bmod 2 \qquad \}$$

Note on Division It is important to notice that the divisions by g on the scheme can be achieved by multiplying by g^{-1}. Notice however, that on the keygen algorithm we are calculating g^{-1} as the inverse of g in R_q, whereas on Decapsulation algorithm g^{-1} is the inverse of g in R_2.

2.2 Digital Signature

Similar to the KEM, the authors present a Digital Signature scheme based on the hardness of finding short vectors in NTRU lattices. The scheme itself uses the hash-and-sign approach proposed by Gentry et al. [12].

For this DS, we will work with the ring $R = \mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$ where $q = 12289$. We will define a new distribution D_f and D_s, from where the secret keys and signatures are derived from, further described on [11].

$$
\begin{aligned}
\text{DSKeyGen}\{\ & f, g \xleftarrow{\$} D_f \\
& h \leftarrow f/g \bmod q \\
& \text{Return}(K_s, K_v) = ((f, g), h) \qquad \}
\end{aligned}
$$

$$
\begin{aligned}
\text{Sig}(K_s, m = (m_1||m_2))\{\ & \\
& t = (m_1 + F(H'(m)) \bmod q || H'(m)) \\
& s_1, s_2 \xleftarrow{\$} D_s \text{ s.t. } hs_1 + s_2 = t \bmod q \\
& \text{Return } \sigma = (s_1, s_2, m_2) \qquad \}
\end{aligned}
$$

$$
\begin{aligned}
\text{Ver}(K_v, \sigma)\{\ & (t_1||t_2) \leftarrow hs_1 + s_2 \bmod q \\
& m_1 \leftarrow t_1 - F(t_2) \bmod q \\
& \text{if } ||(s_1, s_2)|| < B \text{ and } H'(m_1||m_2) = t_2 \\
& \text{Then Accept} \\
& \text{Else Reject} \qquad \}
\end{aligned}
$$

Message Recovery One of the author's main contribution with this scheme is the introduction of Message-Recovery, a technique that substantially decreases the communicational complexity between signer and verifier. On a typical hash-and-sign scheme the signer would need to send both m and s_1, however this scheme avoids sending m and instead sends both s_1 and s_2, leaving the verifier to recover m. s_1 and s_2 are both drawn according to small Gaussians which means that unlike a uniformly random message m they follow a particular Gaussian distribution. This allows us to build a Huffman encoding, which in practice allows us to compress most coefficients to ≈ 9 bits.

Moreover, we can only recover messages of up to $n\log q - 256$bits. Thus when we split our message $m = (m_1||m_2)$, we will have to send m_2 (of size 256) along s_1 and s_2 to the verifier.

This technique allows us to drastically cut the data sent from signer to verifier. Without message recovery we would have to send $|m| + |s_1| = n \cdot \log(12289) + n \cdot \log(12289) = 28n$ bits, whereas with message recovery we only need $|s_1| + |s_2| + |m_2| = n \cdot 9 + n \cdot 9 + 32 = 18n + 256$ bits.

Signing Messages When signing keys, the s_1 and s_2 generation relies on the GPV algorithm and is explained in-depth on Ducas et al. IBE-Scheme work [11]. This algorithm essentially allows us to generate draw two polynomials s_1 and s_2 from a small gaussian distribution such that $s_1 \cdot h + s_2 = t \bmod q$. The strength of this schemes centeres around the fact that it is impossible to predict

whether s_1 and s_2 where generated given t, or that t was generated given s_1 and s_2 [1].

Verifying Messages To verify a message signature we are calculating the Centered Eucledian Norm $||(s_1, s_2)|| = \sqrt{||s_1||^2 + ||s_2||^2}$ (where $||s_i||$ is the l_2-norm of s_i) [8]. Furthermore, B is defined as $\sigma\sqrt{2N}$, where $\sigma = 1.17$ refers to the variance of the D_s distribution [11, 13].

2.3 AKE

Using the KEM and DS previously described, we can build a 2-round foward-secure AKE to authenticate both users, as shown in Figure 1.

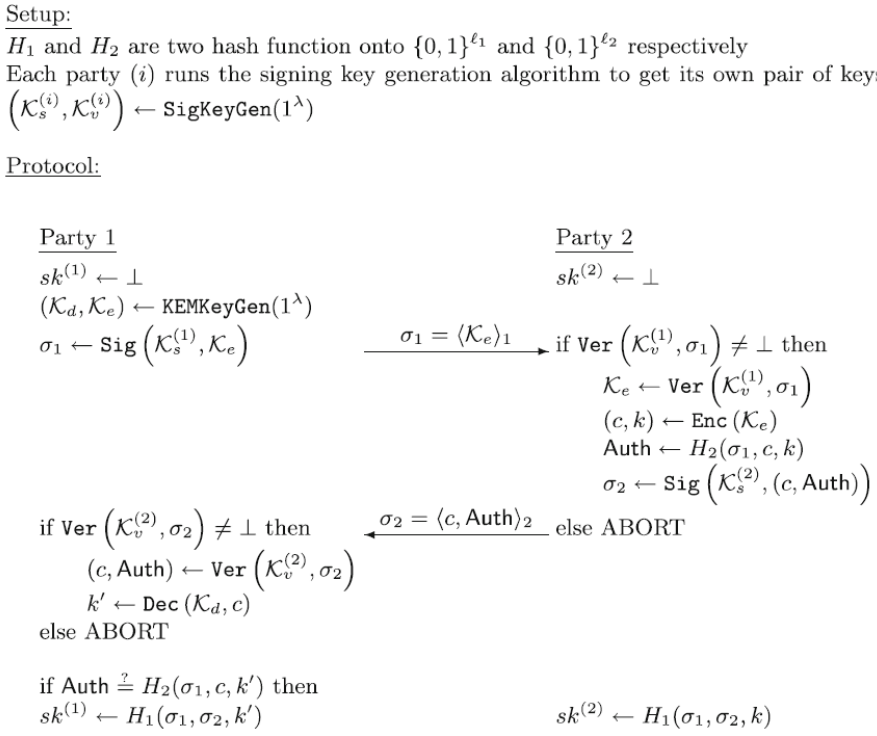

Figure 1: Generic 2-round foward-secure AKE

2.3.1 Parameters

The parameters for the AKE are chosen depending on the security level and failure probability that one wants to achieve. We experimented with the following parameters that the authors recommended for different setups. In practice

parameter sets **I** and **II** are sufficient, however we include **III** and **IV** to counter any overly-optimistic adversary.

	I	**II**	**III**	**IV**
n	512	512	1024	1024
q	12289	12289	12289	12289
KEM D_f, D_e polynomial norms	≈ 93.21	≈ 67.17	≈ 110.42	≈ 79.54
KEM failure prob	$\approx 2^{-30}$	$\approx 2^{-128}$	$\approx 2^{-30}$	$\approx 2^{-128}$
KEM sieving complexity *(log # ops.)*	>128	>115	>269	>246
KEM sieving space *(log bits)*	>114	>104	>227	>209
KEM enumeration complexity *(log # ops.)*	>185	>157	>645	>503
DS D_f, D_s variance	1.17	1.17	1.17	1.17
DS sieving complexity *(log # ops.)*	>102	>102	>237	>237
DS sieving space *(log bits)*	>85	>237	>85	>216
DS enumeration complexity *(log # ops.)*	>130	>130	>520	>520

Table 1: Parameter set for different AKE setups. [1]

3 Implementation

The AKE previously described was implemented on C++ using the NLP library. Code for the GPV and FTT implementations were repurposed from Prest's implementation [14] of an IBE Scheme [11]. FFT was used to allow for quicker multiplications of large polynomials, whilist the GPV code was used to efficiently generate s_1 and s_2 to sign messages.

Furthermore, we compared our implementation to the latest implementation of the New Hope KEX to better understand the computational and communicational efficiency.

3.1 Results and Comparison

In this section we compare our AKE's implementation against the New Hope key exchange [2]. These observations were taken on a computer with an Intel i7 processor and 8 GB RAM running Ubuntu Linux.

3.1.1 Computational Efficiency

To compare the computational efficiency of our AKE implementation, we measured its speed and number of CPU cycles. We ran both our AKE and the New Hope KEX 100 times, and present the averaged results on Table 2.

	New Hope	AKE	
n	1024	512	1024
Avg. Speed (ms)	0.311	48.078	120.320
Avg. # CPU Cycles	$2.09\text{x}10^6$	$3.543\text{x}10^8$	$7.702\text{x}10^8$

Table 2: Comparison of average speed and # of CPU Cycles for a New Hope KEX and our AKE. Notice the KEX is not authenticating users as our AKE. Notice that our AKE $n = 512$ is of comparable security with New Hope $n = 1024$.

Security New Hope needs to use n =1024 to be able to guarantee at least 128 bits of security against a Quantum adversary. Meanwhile, $n = 512$ is enough for our AKE to guarantee 128 bits of security. So, for a fair comparison of between both schemes, we must compare New Hope using $n = 1024$ against our AKE using $n = 512$.

Additionally, although our implementation is substantially slower in comparison with New Hope, it is important to note that New Hope is not actually authenticating its users; it is simply completing a key exchange.

Speed Breakdown On Table 3 we present the average speed and cycles it took to run different sections of the code. This allows us to visualize the expense of each operation, and pin-point where future optimization work should focus on. The AKE described on Table 3 used n=512 and D_f/D_e norm = 93.21.

Alice (Party 1)				
Section	Speed (ms)		CPU Cycles	
	Average	St. Dev	Average	St. Dev
DS Keygen*	1,425.070	183.615	$4.0\text{x}10^9$	$5.1\text{x}10^8$
KEM Keygen	31.464	9.572	$8.8\text{x}10^7$	$2.6\text{x}10^7$
DS Sign	6.008	0.942	$1.6\text{x}10^7$	$2.6\text{x}10^6$
DS Verify	1.113	0.447	$3.1\text{x}10^6$	$1.2\text{x}10^6$
KEM Decapsulate	1.814	0.920	$5.1\text{x}10^6$	$2.5\text{x}10^6$
Hash H2	0.150	0.040	$4.2\text{x}10^5$	$1.1\text{x}10^5$
Hash H1	0.157	0.038	$4.3\text{x}10^5$	$1.0\text{x}10^5$
Total AKE	40.708	10.25	$1.1\text{x}10^8$	$2.8\text{x}10^7$

Bob (Party 2)				
Section	Speed (ms)		CPU Cycles	
	Average	St. Dev	Average	St. Dev
DS Keygen*	1,403.050	212.354	$3.9\text{x}10^9$	$5.96\text{x}10^8$
DS Verify	1.065	0.113	$2.9\text{x}10^6$	$3.2\text{x}10^5$
KEM Encapsulate	1.373	0.296	$3.8\text{x}10^6$	$8.2\text{x}10^5$
H2	0.133	0.028	$3.7\text{x}10^5$	$8.3\text{x}10^4$
DS Sign	5.992	1.277	$1.6\text{x}10^7$	$3.5\text{x}10^6$
H1	0.158	0.041	$4.4\text{x}10^5$	$1.1\text{x}10^5$
Total AKE	8.723	1.622	$2.4\text{x}10^7$	$4.5\text{x}10^6$

Table 3: Breakdown of speed and CPU Cycles taken by different sections of Alice and Bob's code running the AKE with n=512 and D_f/D_e norm = 93.21. *Since DS Keygen needs to be done on first boot only, we did not include it on the Total AKE.

The first thing we notice is that generating the public and private signature keys for Alice (Party 1) and Bob (Party 2) are quite expensive operations, however since this is only done once on first boot, we do not include this for our total AKE calculation.

The KEM Keygen is the AKE's most expensive operation. This is because we must randomly choose a combination of two polynomials and check if they are invertible in two different groups R_q and R_2. These inversions are quite expensive, and we must repeat them at the very least 4 times. However, we also run the risk of having to retest polynomials if we chose some non-invertible ones.

On a positive note, notice that Alice's work is done by the 'Client' who is requesting connection to a 'Server'. It is usually more acceptable for the client to do the heavier load as servers often need to maintain connections with multiple clients at the same time.

3.1.2 Communicational Efficiency

One main benefit of using this AKE, is that the communicational complexity (The amount of information we need to share among parties) is substantially less. Table 4 shows the comparisons done by the authors of our AKE (with and without message-recovery) to the New Hope KEX with a generic hash-and-sign digital signature.

	New Hope [2]		Our AKE	
Hash-and-Sign	Naive	Message-recovery	Naive	Message-recovery
1^{st} flow (bits)	$\approx$ 24000	$\approx$ 19600	$\approx$ 23300	$\approx$ 18900
2^{nd} flow (bits)	$\approx$ 25800	$\approx$ 21400	$\approx$ 23600	$\approx$ 19200
Total flow (bits)	$\approx$ 49800	$\approx$ 41000	$\approx$ 46900	$\approx$ 38100

Table 4: Theoretical comparison of our AKE and New Hope (n=1024). For this comparison, we are considering the AKE produced by combining New Hope alongside the Digital Signature scheme previously presented. (This table is taken from [1]).

In real-world scenarios Internet communication tends to be slower than matching computation speed, thus we believe this reduction in communicational complexity is extremely important for any network related applications.

3.2 Future Work

This implementation provides a good stepping stone for future development of efficient AKEs. We believe there is still code optimization that could be done

on our code, particularly when calculating polynomial inversions which are the most expensive operations on our code. It would also be beneficial to come up with method of generating f, g on our KEM in such a way that we could guarantee its invertibility in R_q and R_2 without having to test for it. Moreover, while NTL claims to use Intel's AVX/AVX2 instructions, perhaps natively using these could also cut down the processing speed.

Another avenue of studies could also focus at how to further encode or compress the data sent, such that the AKE would have even smaller communicational complexity.

We are particularly interested in the prospect of bringing this AKE into the embedded computing field. As we move into the world of IoT devices, we are seeing more and more embedded devices requiring Internet connectivity. On future research, we could target soft-FPGA devices, where one would implement the Gaussian Sampling, FFT and other computing-intesive mathematical operations on hardware, while keeping less intense operations on softwate. Work done by [15] suggests that one could gain substantial speed-ups using this method.

4 Conclusion

In this project we have provided a software implementation for the AKE designed by Del Pino et al. and compared its performance against the popular New Hope key exchange. We have shown that although the AKE is computationally slower, it does provide an increased efficiency in terms of communication. We have also presented viable paths for future work that could help optimize this AKE even further. Finally, we have publicly released this implementation (available at: `https://github.com/pmsosa/PQC`), and it is our hope that this work may serve as a stepping stone for further research and development of efficient AKEs.

References

[1] R. Del Pino, V. Lyubashevsky, and D. Pointcheval, "The whole is less than the sum of its parts: Constructing more efficient lattice-based AKEs," in *International Conference on Security and Cryptography for Networks*, pp. 273–291, Springer, 2016.

[2] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange-a new hope.," *IACR Cryptology ePrint Archive*, vol. 2015, p. 1092, 2015.

[3] NIST, "Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process." `csrc.nist.gov/groups/ST/post-quantum-crypto/documents/call-for-proposals-final-dec-2016.pdf`, 2016. [Online; accessed 20-April-2017].

[4] Google, "Experimenting with Post-Quantum Cryptography." `security.googleblog.com/2016/07/experimenting-with-post-quantum.html`, 2016. [Online; accessed 20-April-2017].

[5] J. Hoffstein, J. Pipher, and J. Silverman, "NTRU: A ring-based public key cryptosystem," *Algorithmic number theory*, pp. 267–288, 1998.

[6] J. Ding, X. Xie, and X. Lin, "A simple provably secure key exchange scheme based on the learning with errors problem.," *IACR Cryptology EPrint Archive*, vol. 2012, p. 688, 2012.

[7] J. W. Bos, C. Costello, M. Naehrig, and D. Stebila, "Post-quantum key exchange for the TLS protocol from the ring learning with errors problem," in *Security and Privacy (SP), 2015 IEEE Symposium on*, pp. 553–570, IEEE, 2015.

[8] J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. H. Silverman, and W. Whyte, "NTRUSIGN: Digital signatures using the NTRU lattice," in *Cryptographers Track at the RSA Conference*, pp. 122–140, Springer, 2003.

[9] L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky, "Lattice signatures and bimodal gaussians," in *Advances in Cryptology–CRYPTO 2013*, pp. 40–56, Springer, 2013.

[10] E. Alkim, N. Bindel, J. A. Buchmann, Ö. Dagdelen, and P. Schwabe, "TESLA: Tightly-secure efficient signatures from standard lattices.," *IACR Cryptology ePrint Archive*, vol. 2015, p. 755, 2015.

[11] L. Ducas, V. Lyubashevsky, and T. Prest, "Efficient identity-based encryption over NTRU lattices," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 22–41, Springer, 2014.

[12] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pp. 197–206, ACM, 2008.

[13] Y.-H. Hung, Y.-M. Tseng, and S.-S. Huang, "Revocable ID-based signature with short size over lattices," *Security and Communication Networks*, vol. 2017, 2017.

[14] T. Prest, "Lattice-IBE." `https://github.com/tprest/Lattice-IBE`, 2016. [Online; accessed 20-April-2017].

[15] C. Rentería-Mejía, A. López-Parrado, and J. Velasco-Medina, "Hardware design of FFT polynomial multipliers," in *Circuits and Systems (LASCAS), 2014 IEEE 5th Latin American Symposium on*, pp. 1–4, IEEE, 2014.