Implementation of a Post-Quantum Cryptographic Algorithm

- STUDENTS / UNIVERSITIES

Aleyna GÜNEŞ/YILDIZ TECHNICAL UNIVERSITY
Arda YÜKSEL /BİLKENT UNIVERSITY
Emre Batuhan BALOĞLU/BİLKENT UNIVERSITY
Gülçin URAS/SABANCI UNIVERSITY

- SUPERVISOR(S)

Erkay SAVAŞ

Sabancı Üniversitesi

PURE PROGRAM FOR UNDERGRADUATE RESEARCH

ABSTRACT

In this project, we have implemented the lattice-based digital signature scheme Dilithium, which is a component of the CRYSTALS (Cryptographic Suite for Algebraic Lattices). Cryptography involves usage of hard problems and this implementation uses Learning with Errors (LWE) and Short Integer Solution (SIS) on Lattices. The hardness of the problem allows it to be used in post-quantum cryptography, in other words, it will be able to resist attacks coming from quantum computers.

Digital Signatures, one of the most crucial elements of the digital world, are a scheme of cryptography that aims to provide verification of integrity, authentication and non-repudiation on digital messages and documents.

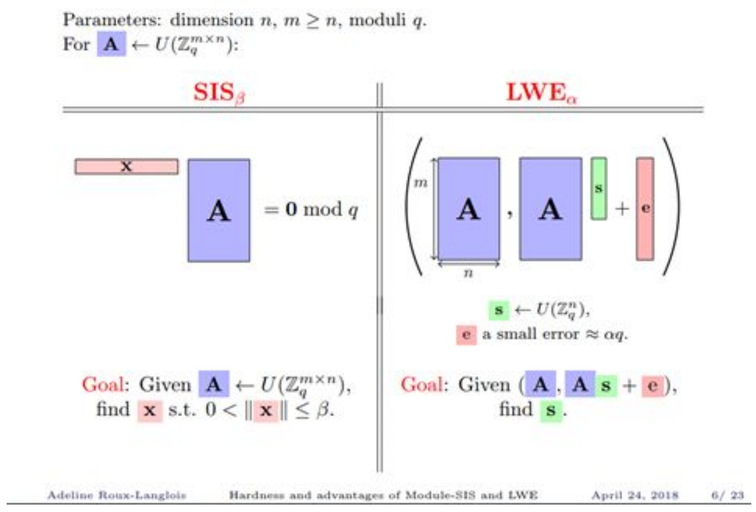

Figure 1: An overview of hard problems, LWE and SIS

Objectives

1- Implementing Dilithium Digital Signature Scheme with Python

2- Ensuring that algorithm works in constant time using discrete Uniform sampling

3- Improving the running time of the scheme with number theoretic transform (NTT).

Project Details

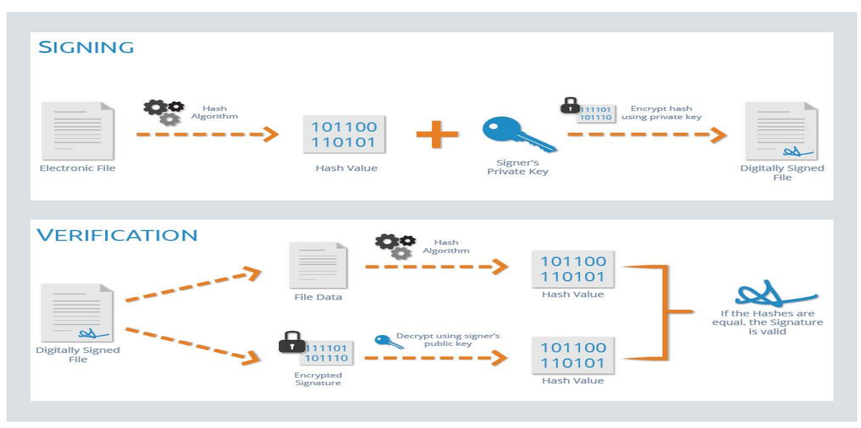

Figure 2: An overview of signature and verification process

Cryptography based on the hardness of lattice problems is seen as a very promising replacement of traditional cryptography after the eventual coming of quantum computers.

In this project, we have implemented the digital signature scheme Dilithium, whose security is based on the hardness of finding short vectors in lattices. We also used its complete optimized implementation, as well as a detailed security analysis of the underlying problems upon which it is based. This scheme was designed with the following pseudocode:

Gen
01 $\rho \leftarrow \{0,1\}^{256}$
02 $K \leftarrow \{0,1\}^{256}$
03 $(\mathbf{s}_1, \mathbf{s}_2) \leftarrow S_\eta^\ell \times S_\eta^k$
04 $\mathbf{A} \in R_q^{k \times \ell} := \mathsf{ExpandA}(\rho)$ // $\mathbf{A}$ is stored in NTT Domain Representation
05 $\mathbf{t} := \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$
06 $(\mathbf{t}_1, \mathbf{t}_0) := \mathsf{Power2Round}_q(\mathbf{t}, d)$
07 $tr \in \{0,1\}^{384} := \mathsf{CRH}(\rho \,\|\, \mathbf{t}_1)$
08 **return** $(pk = (\rho, \mathbf{t}_1), sk = (\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0))$

Sign(sk, M)
09 $\mathbf{A} \in R_q^{k \times \ell} := \mathsf{ExpandA}(\rho)$ // $\mathbf{A}$ is stored in NTT Domain Representation
10 $\mu \in \{0,1\}^{384} := \mathsf{CRH}(tr \,\|\, M)$
11 $\kappa := 0, (\mathbf{z}, \mathbf{h}) := \perp$
12 **while** $(\mathbf{z}, \mathbf{h}) = \perp$ **do**
13 $\quad \mathbf{y} \in S_{\gamma_1 - 1}^\ell := \mathsf{ExpandMask}(K \,\|\, \mu \,\|\, \kappa)$
14 $\quad \mathbf{w} := \mathbf{A}\mathbf{y}$
15 $\quad \mathbf{w}_1 := \mathsf{HighBits}_q(\mathbf{w}, 2\gamma_2)$
16 $\quad c \in B_{60} := \mathsf{H}(\mu \,\|\, \mathbf{w}_1)$
17 $\quad \mathbf{z} := \mathbf{y} + c\mathbf{s}_1$
18 $\quad (\mathbf{r}_1, \mathbf{r}_0) := \mathsf{Decompose}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)$
19 $\quad$ **if** $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ or $\|\mathbf{r}_0\|_\infty \geq \gamma_2 - \beta$ or $\mathbf{r}_1 \neq \mathbf{w}_1$, then $(\mathbf{z}, \mathbf{h}) := \perp$
20 $\quad$ **else**
21 $\quad\quad \mathbf{h} := \mathsf{MakeHint}_q(-c\mathbf{t}_0, \mathbf{w} - c\mathbf{s}_2 + c\mathbf{t}_0, 2\gamma_2)$
22 $\quad\quad$ **if** $\|c\mathbf{t}_0\|_\infty \geq \gamma_2$ **or** the # of 1's in $\mathbf{h}$ is greater than ω, then $(\mathbf{z}, \mathbf{h}) := \perp$
23 $\quad \kappa := \kappa + 1$
24 **return** $\sigma = (\mathbf{z}, \mathbf{h}, c)$

Verify$(pk, M, \sigma = (\mathbf{z}, \mathbf{h}, c))$
25 $\mathbf{A} \in R_q^{k \times \ell} := \mathsf{ExpandA}(\rho)$ // $\mathbf{A}$ is stored in NTT Domain Representation
26 $\mu \in \{0,1\}^{384} := \mathsf{CRH}(\mathsf{CRH}(\rho \,\|\, \mathbf{t}_1) \,\|\, M)$
27 $\mathbf{w}_1' := \mathsf{UseHint}_q(\mathbf{h}, \mathbf{A}\mathbf{z} - c\mathbf{t}_1 \cdot 2^d, 2\gamma_2)$
28 **return** $[\![\|\mathbf{z}\|_\infty < \gamma_1 - \beta]\!]$ **and** $[\![c = \mathsf{H}(\mu \,\|\, \mathbf{w}_1')]\!]$ **and** $[\![\#$ of 1's in $\mathbf{h}$ is $\leq \omega]\!]$

Figure 3: Pseudocode of our Digital Signature Scheme

Our implementation consists of three main functions: Gen, Sign and Verify.

Gen function -which stands for key generation- generates public and secret keys in order to be used in signature and verification protocols. It generates them by using Uniform Distribution on NTT domain, with a random seed, which is randomly selected.

Sign function gets secret key and a message, which will be digitally signed. The function maps the message and secret key to a binary string, which is 48 bytes. Then, rejection sampling is performed on this 48 bytes in order to avoid dependency on secret key; otherwise we may have a security leak. The function returns the digitally signed message.

Verify function gets public key, the message in its first form and the digitally signed message. Then the function performs a zero knowledge proof on these inputs in order to confirm or reject if the message is really signed or not. The function returns a boolean value which indicates this confirmation.

Results and Conclusions

In this project, Dilithium's initial algorithm is implemented successfully. From the tests, Digital Signature Scheme is observed to be working flawlessly. Due to lack of usage of NTT, timing issue affected the progression. However, results are gathered correctly regardless of the computational time issues.

From the tests that involve 1000 samples of the algorithm, constant time aspect is proven to be correct. Following Results are obtained:

Average Generation Time(s)	Average Signing Cycle Time(s)	Average Verification Time(s)	Average Signing Cycle(s)	Average Run Time(s)
0.79	1.33	1.25	6.82	11.15

In conclusion, the implementation has achieved most of the objectives and issues regarding the time and memory management can be further developed after the usage of NTT in later stages of the projects.

References

- Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Stehlé, D.: Crystals - Kyber: a CCA-secure module-lattice-based KEM. Cryptology ePrint Archive, Report 2017/634 (2017). http://eprint.iacr.org/2017/634
- Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Dilithium: a lattice-based digital signature scheme. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2018(1), 238–268 (2018). 554, 555, 557, 573, 575, 579, 580, 581, 582, 583