FALCON
A Post-Quantum Signature Scheme

Scott Furman

Abstract—This paper will serve as a case study on FALCON, a post-quantum signature scheme submitted to the NIST post-quantum competition. Falcon was designed by Thomas Prest, Pierrre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhengfei Zhang. After providing some background, we will go through the algorithms for key generation, signing, and verifying. We will also analyze some performance measures and compare them to other signature schemes.

1 Introduction

On December 20, 2016, the National Institute of Standards and Technology (NIST) officially put out the call for public key cryptographic algorithms that would be resistant to both classical and quantum computers. These will replace the current algorithms, many of which rely on integer factorization, that will be broken by sufficiently powerful quantum computers. In NISTs call for proposals, they say, "Some engineers even predict that within the next twenty or so years sufficiently large quantum computers will be built to break essentially all public-key schemes currently in use. Historically, it has taken almost two decades to deploy our modern public key cryptography infrastructure. Therefore, regardless of whether we can estimate the exact time of the arrival of the quantum computing era, we must begin now to prepare our information security systems to be able to resist quantum computing". The submission deadline was set to November 30, 2017 for digital signatures, public-key encryption, and key-establishment algorithms.

2 FALCON Overview

One such submission was FALCON, a post-quantum signature scheme created by a team of cryptographers led by Thomas Prest. FALCON is a lattice-based scheme (hence the "L" in "FALCON") that uses the short integer solution problem as its underlying hard problem. Lattices are thought to be quantum resistant, so lattice-based schemes are popular among NIST submissions. The short integer solution problem involves finding integer coefficients to a set of vectors that make up the lattice. It is hard on the average case.

The "FA" in FALCON stands for "Fast Fourier", an implementation of a trapdoor sampler used to generate short vectors. In general, a trapdoor sampler takes in a matrix, A, a trapdoor function, T, and a target, c. It then outputs a vector, s, such that $sA = c \mod q$. The shorter the vector outputted by the sampler, the more secure it is. FALCON uses a Fast Fourier sampler to accomplish this and generate vectors to be used as signatures. The specific implementation they use is the Ducas and Prest Fast Fourier sampler, which was co-created by Thomas Prest, the lead designer for FALCON.

The final "N" in FALCON stands for "NTRU", an existing lattice-based system. In NTRU, the secret key consists of four polynomials, f, g, F, and G, which verify the NTRU equation: $fG - gF = q \mod \phi$. The public key is a polynomial, h, such that $h = gf^{-1} \mod q$. Because of these properties, $\begin{bmatrix} 1 & h \\ 0 & q \end{bmatrix}$ and $\begin{bmatrix} f & g \\ F & G \end{bmatrix}$ generate the same lattice. It will become more apparent as to how FALCON uses this system in the following sections.

Lastly, the "C" in FALCON stands for "Compact" as in compact signatures. The quest for

compactness was a leading goal when designing FALCON. More specifically, they wished to minimize the size of the public key and the size of the signature: $\|pk\| + \|sig\|$ = (bit size of the public key) + (bit size of the signature). Hashed-base cryptographic schemes often have small public keys, but large signatures. Conversely, code-based schemes will often have small signatures but large public keys. FALCON uses lattices, which typically provide the best of both worlds. Another key aspect for the design was to encourage modularity; while FALCON was build over NTRU lattices, they could be substituted for any other lattice system. In total, FALCON stands for "Fast Fourier Lattice-Based Compact Signatures over NTRU."

Algorithm 8 Keygen(ϕ, q)

Require: A monic polynomial $\phi \in \mathbb{Z}[x]$, a modulus q
Ensure: A secret key sk, a public key pk

$f, g, F, G, \gamma \leftarrow$ NTRUGen(ϕ, q)
$\mathbf{B} \leftarrow \left[\begin{array}{c|c} g & -f \\ \hline G & -F \end{array}\right]$
$\hat{\mathbf{B}} \leftarrow$ FFT$(\mathbf{B})$
$\mathbf{G} \leftarrow \hat{\mathbf{B}} \times \hat{\mathbf{B}}^{\star}$
$\mathsf{T} \leftarrow$ ffLDL$^{\star}(\mathbf{G})$
if ϕ is binary then
    $\sigma \leftarrow 1.55\sqrt{q}$
else if ϕ is ternary then
    $\sigma \leftarrow 1.32 \cdot 2^{1/4}\sqrt{q}$
for each leaf leaf of T do
    leaf.value $\leftarrow \sigma / \sqrt{\text{leaf.value}}$
sk $\leftarrow (\hat{\mathbf{B}}, \mathsf{T})$
$h \leftarrow g f^{-1} \bmod q$
pk $\leftarrow h$
return sk, pk

Fig. 1. FALCON's key generation algorithm.

3 FALCON Specification

3.1 Public Parameters

There are three public parameters to consider in FALCON: ϕ, q, and β. Additionally, n, refers to the degree of the polynomials used to generate the lattices. There are three FALCON-n variations: FALCON-512, FALCON-768, and FALCON-1024. ϕ refers to a monic and irreducible polynomial used as a modulus. In the case that n is a power of two, $\phi = x^n + 1$. If n is three times a power of two, $\phi = x^n - x^{n/2} + 1$. The option between binary or ternary opens up more options for values of n. Next, we have q, an integer modulus. In the binary case, q is equal to 12289, and in the ternary case, q is equal to 18433. Lastly, β is a bound for vector size.

3.2 Key Generation

The key generation for FALCON comes in two major steps: generating the polynomials that populate the lattices, and building the FALCON tree. The polynomials that must be generated are the same polynomials used in the NTRU equation. That is, we need to generate four polynomials, f, g, F, and G, such that $fG - gF = q \mod \phi$. To do this, f and g are first randomly generated with two constraints. We must ensure that our public key h can be computed by $h = gf^{-1} mod(\phi, q)$. In other words, f must be invertible. Additionally f and g must be capable of generating sufficiently short signatures. Once f and g have been generated, F and G can be computed to verify the NTRU equation.

Now that we have the polynomials, they need to be arranged into a secret key format. The FALCON secret key consists of $\hat{B}$ and T, where $\hat{B} = \begin{bmatrix} FFT(g) & -FFT(f) \\ FFT(G) & -FFT(F) \end{bmatrix}$ and FFT refers to the Fast Fourier transformation. T, also known as the FALCON tree, is the normalized LDL decomposition of $\hat{B} \times \hat{B}^*$. Again, their uses will become more apparent when analyzing the algorithms for signing and verifying.

3.3 Signing

The signing and verifying algorithms are rather simple, and the true challenge for implementation lies in the Fast Fourier sampling. Given a secret key, sk, and a message, m, first a random salt, r, is generated. The concatenation of r and m, $r\|m$ is hashed into a new value, c.

Algorithm 17 Sign $(\mathsf{m}, \mathsf{sk}, \beta)$

Require: A message m, a secret key sk, a bound β
Ensure: A signature sig of m

$\mathsf{r} \leftarrow \{0,1\}^{320}$ uniformly
$c \leftarrow \mathsf{HashToPoint}(\mathsf{r} \| \mathsf{m})$
$\mathbf{t} \leftarrow (\mathsf{FFT}(c), \mathsf{FFT}(0)) \cdot \hat{\mathbf{B}}^{-1}$
do
    $\mathbf{z} \leftarrow \mathsf{ffSampling}_n(\mathbf{t}, \mathsf{T})$
    $\mathbf{s} = (\mathbf{t} - \mathbf{z})\hat{\mathbf{B}}$
while $\|\mathbf{s}\| > \beta$
$(s_1, s_2) \leftarrow \mathsf{invFFT}(\mathbf{s})$
$\mathsf{s} \leftarrow \mathsf{Compress}(s_2)$
return $\mathsf{sig} = (\mathsf{r}, \mathsf{s})$

Fig. 2. FALCON's signature generation algorithm.

Algorithm 20 Verify $(\mathsf{m}, \mathsf{sig}, \mathsf{pk}, \beta)$

Require: A message m, a signature $\mathsf{sig} = (\mathsf{r}, \mathsf{s})$,
Ensure: Accept or reject

$c \leftarrow \mathsf{HashToPoint}(\mathsf{r} \| \mathsf{m}, q, n)$
$s_2 \leftarrow \mathsf{Decompress}(\mathsf{s})$
$s_1 \leftarrow c - s_2 h \bmod q$
if $\|(s_1, s_2)\| \leq \beta$ then
    accept
else
    reject

Fig. 3. FALCON's verification algorithm.

A preimage, t, of c is computed and given as input to the Fast Fourier sampler. The sampler outputs two short polynomials, s_1 and s_2 such that $s_1 + s_2 h = c \mod q$. The polynomial s_2 is then compressed into a bit string, s. The signature is the pair of the salt and the compressed polynomial, (r, s).

3.4 Verifying

Verification is even more simple, both logically and computationally, because it can evaluate a signature without reverting the Fast Fourier sampling. Given the secret key, sk, the message, m, and the signature, (r, s), the concatenation of r and m are again hashed into a value, c. The bitstring, s, is then decompressed back into the polynomial s_2. The value of s_1 can then be recomputed because $s_1 = c - s_2 h \mod q$. If the two polynomials are shorter than the specified bound, β, then the signature is accepted as valid. Otherwise, it is rejected.

4 Performance

The FALCON team provided data on the performance of each FALCON variant, using what they referred to as a "common laptop computer." See TABLE 1 for a full listing of performance for key generation, signing, and verifying, as well as information on the average size of public keys and signatures. Using the C implementation, we also tested FALCON and got the results found in TABLE 2. Additionally, using the OpenSSL speed tool, we gathered information on the performance of two existing digital signature algorithms: RSA and DSA. FALCON is claimed to have security comparable to RSA-2048. RSA-2048 was able to generate 730.8 signatures per second, and perform 25399.9 verifications per second. DSA-2048 generated 2215.7 signatures per second, and performed 2087.8 verifications per second.

5 Conclusion

FALCON outperforms classical signature schemes like RSA and DSA in both speed and signature size. In fact, the FALCON design team claims that FALCON generates more compact signatures than any other scheme they were aware of. Additionally, there are no glaring security concerns, and the most effective known attacks involve lattice-reduction techniques that any lattice-based system would be vulnerable against.

TABLE 1
Provided Performance Results Using Intel i7 @ 3.30 GHz

Variant	Keygen (ms)	Sign/s	Verify/s	Public Key Size (bytes)	Signature Size (bytes)
FALCON-512	6.98	6081.9	37175.3	897	617.38
FALCON-768	12.7	3547.9	20637.7	1441	993.91
FALCON-1024	19.6	3072.5	17697.4	1793	1233.3

TABLE 2
Experimental Results Using AMD FX-9370 @ 4.40 GHz

Variant	Keygen (ms)	Sign/s	Verify/s	Signature Size (bytes)
FALCON-512	8.85	4832.147	25115.593	617.36
FALCON-768	16.2	2864.962	15087.424	993.92
FALCON-1024	25.6	2319.858	11848.316	1233.29

However, they also note that resistance against side-channel attacks is largely unstudied. Comparatively, FALCON appears to be a strong contender for the NIST competition for digital signatures.

REFERENCES

[1] An Overview of the NTRU Cryptographic System. (2014). [pdf] San Diego: San Diego State University. Available at: https://sdsu-dspace.calstate.edu/bitstream/handle/10211.3/135700/Nguyen_sdsu_0220N_10605.pdf?sequence=1.

[2] Csrc.nist.gov. (2016). *Post-Quantum Cryptography*. [online]. Available at: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography.

[3] Fouque, P., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W. and Zhang, Z. (2017). *FALCON: Fast-Fourier Lattice-based Compact Signatures over NTRU*. 1st ed. [pdf]. Available at: https://falcon-sign.info/falcon.pdf.

[4] Fouque, P., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W. and Zhang, Z. (2018). *Falcon*.