**Squirrels (Square Unstructured Integer Euclidean Lattice Signature)**

Names: Daymon, Jaraad Hussain, Leonardo Yeung, Al Fahim

**1.1 Context and motivation**

NIST has recently standardized stateful hash-based signature schemes like XMSS and LMS, as well as three lattice-based schemes (Kyber KEM, Dilithium, and Falcon) and a stateless hash-based signature scheme called SPHINCS. These efforts underscore the growing significance and feasibility of post-quantum cryptographic primitives.

However, adopting entirely new cryptographic systems presents significant challenges due to the cost and time involved. Additionally, ensuring the longevity of these systems over decades is a concern. This is especially relevant given the unpredictable trajectory of quantum computing technology and quantum cryptanalysis in the coming years. In some application domains, taking a conservative approach to selecting post-quantum candidate schemes may be preferable.

In this context, the choice of the class of hard problems to rely on becomes crucial. Practitioners with valuable data requiring long-term confidentiality and authenticity guarantees may prioritize security and simplicity over premature optimization. As quantum computing technology evolves, so does the potential for it to break existing cryptographic systems. This calls for a strategic approach that balances innovation with the need for enduring security.

Basically in the field of post-quantum cryptography, there has been substantial progress, with NIST playing a pivotal role in standardization. The challenges of deploying entirely new cryptographic systems and ensuring their longevity must not be underestimated. For certain applications, a conservative approach in choosing cryptographic schemes may be wise. The choice of hard problems to base these schemes on should align with the need for long-term security in the face of the uncertain future of quantum computing and cryptanalysis.

**1.2 The choice of unstructured lattices**

The field of post-quantum cryptography faces uncertainties regarding the long-term viability of algebraically structured lattices, prompting the need for secure and resilient alternatives. Our proposal hinges on plain lattice problems, devoid of additional algebraic structures, as a foundational basis for cryptographic security. We have opted for conservative parameterizations to bolster security, even if this choice leads to certain efficiency trade-offs compared to algebraic variants. The rationale behind this decision is to ensure robustness against potential vulnerabilities that may emerge in the future.

In opting for plain lattices, we aim to create cryptographic solutions that are less reliant on specialized mathematical structures, which may become targets for advanced attacks by quantum computers. This approach prioritizes security and resilience over optimizing for performance.

A key advantage of using plain lattices is their flexibility in parameter selection. This allows for tailoring cryptographic security to specific requirements, achieving a fine-grained level of control over security levels. This adaptability is particularly valuable as the landscape of cryptography continually evolves, and the threat posed by quantum computing remains unpredictable.

In summary, our proposal in post-quantum cryptography leans on plain lattice problems, eschewing algebraic structures to enhance long-term security. While this choice might introduce some efficiency trade-offs when compared to algebraic variants, it is a calculated decision to fortify cryptographic robustness in the face of future uncertainties. The use of plain lattices not only minimizes reliance on potentially vulnerable structures but also offers the flexibility to fine-tune security levels according to specific needs, providing a versatile and adaptable approach to post-quantum cryptography.

**SECTION 2 Design Rationale**

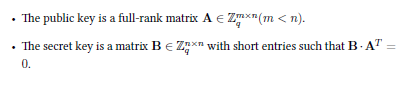
**Leonardo Yeung**

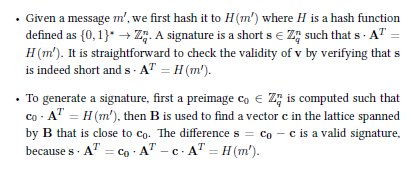
**2.1.1 Historical Context**

In 2008, Gentry, Peikert, and Vaikuntanathan unveiled a novel approach for crafting digital signatures, anchored in the SIS assumption's security (Short Integer Solution problem). This foundation has been instrumental in building various cryptographic systems. Central to their methodology is the "trapdoor basis" - a kind of special secret key. Leveraging this trapdoor basis, they apply the Gaussian sampling technique to pinpoint a lattice point close to a message's 'hash'. For recipients, the verification of a message necessitates assessing the signature's validity in relation to the lattice and ascertaining the proximity between the signature's lattice point and the message's hash. The shift from the Goldreich-Goldwasser-Halevi (GGH) signature blueprint to the GPV ushered in an improved paradigm. The GPV, instead of exposing the secret basis, emits signatures that remain cryptic about the secret key, enhancing its security profile.

**2.1.2-2.2 Main Mechanism**

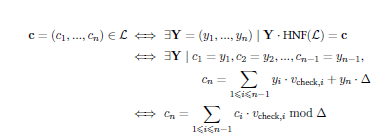
Squirrel ingeniously adopts the Gentry-Peikert-Vaikuntanathan Framework. It progresses from the preliminary GGH technique to the more refined GPV scheme, enhancing the security measures while producing lattice points essential for digital signatures. Squirrel emphasizes the Hermite Normal Form (HNF) of a lattice. This structure paves the way for streamlined lattice membership validations and efficient verification of the co-cyclic lattice's HNF.





**2.3 Security Assumptions**

Squirrel operates on the potent SIS-hash apparatus within the co-cyclic architecture. This tool exhibits flexibility, suitable for diverse lattice categories, marking a significant departure from the conventional uniform SIS-oriented lattices. The underpinning security of the system lies in the generalized SIS problems and their inherent hardness assumptions. Challenges such as GSIS underscore the system's cryptographic integrity, rooted in the robustness of these quandaries.



**2.4 Squirrel Family**

This section is an overview on what Squirrel contains. A more detailed look into each of these following topics will be in the next couple of sections like the algorithms used how it works.

\*\*Signature Schemes\*\*: Squirrels offers a suite of lattice-centric digital signature methods, with a focus on unstructured co-cyclic lattices, bringing unique cryptographic benefits to the fore.

\*\*Secret Keys\*\*: While Squirrels integrates the Klein sampler in signature creation, it introduces a unique trapdoor genesis pathway. The signature's caliber is intrinsically linked with the trapdoor basis's Gram-Schmidt norm, ensuring cryptographic resilience.

\*\*Public Key Derivation\*\*: The co-cyclicity of the sampled lattice is paramount for efficient public key synthesis, with the vector "vcheck" representing the Squirrels' public key.

\*\*Signature Generation\*\*: Squirrels amalgamate hashing mechanisms with a random nonce for signature generation, deriving a distinct vector. The lattice's secret basis and Klein's Gaussian sampler are instrumental in this phase.

\* \*\*Verification Mechanism\*\*: Verification is a priority in the Squirrels framework. Harnessing co-cyclic lattice attributes, the system ensures swift and streamlined validation processes.

**3.1 Advantages**

SQUIRREL has a big number of advantages, it does not rely on lattices that do not have strong geometric properties so it bases its security on generic lattice problems. Lattices with strong geometric properties often have recognizable patterns that make them more prone to attacks whereas lattices without strong geometric properties are more complex and random which make it more challenging for attackers to get through. Those problems were studied for years so they have strong mathematical proofs that makes SQUIRREL confident with their security. Complex problems and Squirrel go hand in hand but it is still able to create small signatures. The size of the signatures are in bytes, small enough to be compared to Falcon and Dilithium signatures, the two most well known methods for creating small signatures in post quantum cryptography. Squirrel is competitive in terms of signature generation and verification efficiency because even on a small laptop, Squirrel can create several dozens to hundreds of signatures in a single second. When checking if a signature is valid all the squirrel has to do is run a simple math equation in the form of a linear equation. The process of making and verifying signatures is simple and straightforward without making the security less strong.

**3.2 Limitations**

With SQUIRRELS making keys takes time because of how complicated the math involved with big matrices are, calculating determinants and using something called the Hermite Normal Form. HERMITE NORMAL FORM is used to simplify a matrix to perform mathematical operations on it.Hermite Normal Form is equivalent to Row Echelon Form in linear algebra. When producing keys it may take several seconds to make a key pair but when changing keys often it can be a problem because SQUIRREL uses unstructured lattices which mean it will take more time while getting bigger exponentially. The use of unstructured lattices impacts the making of public keys, it restricts the minimum size so it cannot be made smaller like systems such as falcon. This may impact limitations on storage and data transfer.

**4.1 Heuristic modelization of lattice reduction,GSA and beyond**

The text discusses the need to accurately assess the hardness of underlying problems in lattice-based cryptography, focusing on the Shortest Vector Problem (SVP). To do this, the authors propose modeling the behavior of a practical oracle that approximates the SVP, specifically using the Block Korkine-Zolotarev (BKZ) algorithm. They emphasize the conservative estimation of the "core-SVP hardness" to account for potential future improvements in lattice reduction techniques.

The BKZ algorithm, with a block size denoted as B, may require a polynomial number of calls to an SVP oracle in dimension B. However, for this analysis, they only consider the cost of a single call to the SVP oracle, taking a cautious approach. This is due to the fact that there are methods to amortize the cost of SVP calls within BKZ, particularly when sieving is employed as the SVP oracle.

Sieving is becoming the de facto standard for larger cryptographic block sizes due to its practical advantages. The authors introduce the concept of "Geometric series assumption (GSA)" for self-dual BKZ reduction. It assumes that the norm of Gram-Schmidt vectors in a reduced lattice basis decreases with geometric decay. This assumption is formalized to provide a more accurate estimation of the block size required for a successful attack, allowing for a more precise assessment of security.

The authors also mention a probabilistic simulation technique proposed in a previous work, which enhances the analysis by considering the "quadratic tail" phenomenon of reduced bases. This phenomenon is crucial for improving the precision of calculations, as it reflects real-world scenarios where lattice problems become harder as the lattice dimension increases.

The text highlights the need for a rigorous assessment of the security of lattice-based cryptographic schemes, with a focus on SVP. They opt for a conservative estimate of the hardness of SVP, incorporating the BKZ algorithm and the GSA. To refine their analysis further, they employ a probabilistic simulation technique that accounts for the quadratic tail phenomenon in reduced bases.

This analysis method is used to estimate the security of lattice-based cryptography, following the "core-SVP methodology." The bit complexity of lattice sieving, taken as the best SVP oracle, is considered in the classical and quantum settings for various block dimensions. This information is essential for understanding the security of lattice-based cryptographic schemes in both key recovery and forgery scenarios.

**4.2 Key Recovery Attack** The key recovery attack in lattice-based cryptography aims to find at least one short vector of the secret basis, based on knowledge of the public key. Traditional approaches involve lattice reduction on the public basis, but when the key is sparsely populated with ternary elements due to its length, hybrid attacks become a potential threat. This technique, introduced in the Falcon specification and used in Mitaka, involves examining the lattice formed by the public basis, encoded in the public key as the Hermite Normal Form of the lattice. The attack restricts the search space by projecting the lattice onto a subspace. After reducing the public basis using the DBKZ algorithm, the lattice is projected onto a subspace. If a projection of a secret vector is found in this subspace, it can be efficiently lifted to a vector of the desired norm. Using a classical sieve, all vectors of smaller norms are listed. Under certain assumptions, the attack condition for retrieving the projection among the sieved vectors is established. This technique helps in extracting information about the secret vectors from the public basis, posing a security concern in sparse ternary keys.

4.3 Hybridizing the Attack for sparse secrets

Sparse secrets are secret values or data that contain a lot of zeros or empty elements. These zeros can make certain types of attacks more difficult because the attacker has to find the position of non-zero elements in the secret. When hybridizing an attack for sparse attacks, if the first one of the short vectors has many zeros/high sparsity the accuracy for guessing other positions will be more accurate. A good guess will allow us to search for secret keys outside these positions by intersecting the public lattice with the complement set of l. The complete set of “l” will contain all the elements that are not in the set “l”. To hybridize the attack there are two main components, guessing sparse vector and lattice reduction. Guessing the sparse vector means you guess the positions of the zeros in the sparse vector in the lattice. When the guess is correct then it will reduce the search space. Lattice Reduction means the transformation of the basis into a more structured form making it easier to find shorter vectors. So in all, when formulating a method for trying to recover a secret vector from a lattice we can start by randomly guessing the positions of zeros. Then we would compute the L’ by taking the intersection of the original lattice L and removing the positions not in both sets. This would reduce the lattice to smaller dimensions which would speed up and make the attack process more efficient. Then we use the BKZ algorithm to reduce the lattice basis into a more structured form which would then make it easier to find shorter vectors. After that Gram-Schmidt orthogonalization process is used to get the result of the norm of the Gram-Schmidt vector from the basis vectors. If the norm of the vector is smaller than 4/3 times the norm of the Gram-Schmidt vector then we can include each vector into the list. Next we can use Babai’s nearest plan algorithm to find the closest lattice point within L’ for each of these vectors. We can calculate the length of the lifted vector in L’ and compare it to the gmax/ the specific needs of the problem. If the length of the lifted vector is shorter than gmax we can return it as a potential lattice vector.

4.4 Signature Forgery by BDD Reduction

Faking signatures can be done with different methods. Faking signatures involves introducing a specific point in a grid-like structure that's not too far from a random value. We can fake signatures with a framework called Bounded Distance Decoding (BDD). It makes assumptions and meets a specific condition about the distance between two points to solve the problem.

4.4.2 Additional “BUFF” Security Properties

**Specification:**

**5.3.1 - Private Key**

The private key in Squirrels is represented as a n × n matrix. This matrix's rows each stand for a vector of B. This private key matrix was created according to certain mathematical standards. The first one is that every vector in B has to satisfy the condition that its norm, length, is between the bounds set by g-min and g-max. In other words, the vector lengths must not be too small or too big. The second one is that the determinant of B needs to equal that of delta, denoted as det(B) = ∆. The third one is that the private key, matrix B, is co-cyclic, which results in the existence of a particular Hermite Normal Form (HNF) called "1." The private key can also be dynamically recalculated using a seed. This is a very time-consuming process, so it is usually okay to keep at least the last vector of the private key matrix (B) with the seed. When the seed is stored, it allows for more efficient and faster key regeneration when needed.

**5.3.2 - Public Key**

The public key in Squirrels cryptography is denoted by this formula,

pk = (v-check,i mod p)1⩽i⩽n−1,p∈P∆. It is first represented by a value called vector check. The formula then takes the modulo for every prime number, p, in the set P∆. This modulo operation yields a result for each "i" ranging from 1 to "n - 1." It's worth noting that the last coordinate of v-check does not need to be stored in the public key because it represents the determinant, which is a fixed and known parameter. In short, the public key is created by taking the value v-check and performing modulo operations on various prime numbers from P, with the exception of the last coordinate, which is associated with a fixed parameter.

**5.4 - Key Pair Generation**

There are three main steps that must be completed in order to generate a key pair.. The first step is generating the first n - 1 vectors of the secret basis. These vectors are essential parts of the private key. It achieves this with Gram-Schmidt norms between g-min and g-max. In order to bound the norm of the last Gram-Schmidt vector, it uses a formula that controls the value of each step using a variable, *i*, so that it can remain small in absolute value. This process moves forward one step at a time and is sequential. Each step of the procedure involves selecting a candidate vector and checking that the orthogonal component, denoted as vB, is evenly distributed among the vectors of B and has a norm contained within the regions defined by "b-low" and "b-up." The vector v is then rounded to an integral vector by rounding each of its coordinates. The second step is to compute the final secret basis vector. This vector is calculated in such a way that the entire basis reaches the desired target determinant. In the last step, the public key is derived from the row Hermite Normal Form (HNF) of the secret basis. The norms of the Gram-Schmidt vectors are carefully controlled throughout this key pair generation process, as well as the expected separation from the target determinant. To ensure that the last vector's norm stays within established limits, this is crucial.

**5.5 - Hashing**

In a hash-and-sign signature scheme, the message is hashed before being signed or verified. To sign or verify a message in Squirrels, it is hashed, which converts it to a vector denoted as, Zn. This transformation is based on a valid extendable-output hash function (XOF). The chosen XOF must meet or exceed the signature scheme's security level. An algorithm is then introduced called HashToPoint, which converts a message into a vector whose values fall within the specified range [0, q - 1] for n - 1 dimensions. Also, the last coordinate is fixed at 0.

**5.6 - Signature Generation**

In order to sign a message "m," Squirrels first hashes the message along with a salt value "r." This hashing operation returns a point "h" that is located in the space [0, q − 1] n−1 × {0}. This transformation is carried out by the "HashToPoint" function, as mentioned in the hashing section. Squirrels will then use Klein's trapdoor sampler to find a lattice point that is close to "h" after obtaining the point "h."

**5.8 - Signature Verification**

Verifying the signature requires a few steps. First we recompute the hash point “h” which we get from the message “m” and salt “r” using the HashToPoint algorithm. Next, we do a Euclidean Norm Check to verify that the signature, “s”, falls within the appropriate change. The last step is using the public key to verify that the sum of the signature, "s", and the recalculated "h", denoted as (s + h), is part of the lattice structure. This will ensure the accuracy of the signature and its relationship to the original message.

References:

<https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/Squirrels-spec-web.pdf>