Post-Quantum Cryptography: SQUIRRELS

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

Intro

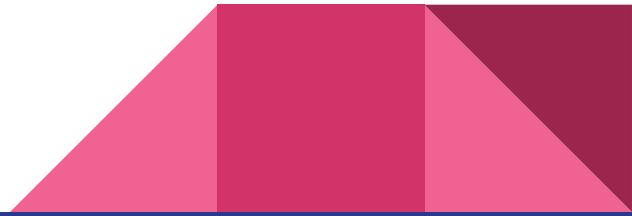
SQUIRRELS

Square Unstructured Integer Euclidean Lattice Signature

- Current post-quantum cryptography faces uncertainties
- Long-term viability of algebraically structured lattices uncertain
- Plain lattice is the future
- No more unnecessary additional algebraic structures
- Conservative parameterizations to bolster security
- Ensures robustness

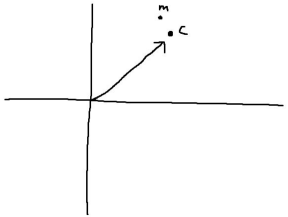
Design Rationale

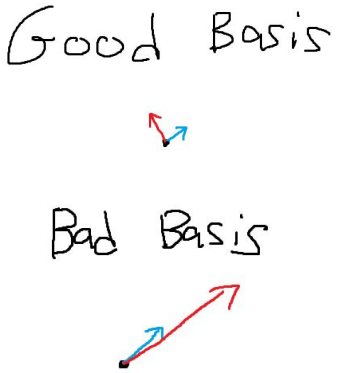
- History
- Main Mechanism
- Squirrel Family



Main Mechanism

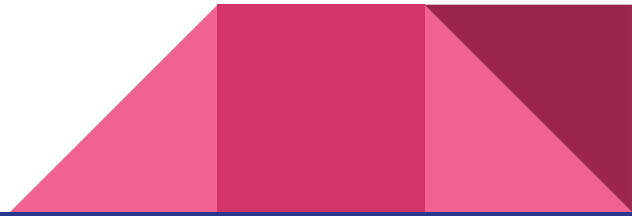
- Uses GPV improvement of GGH
  - GGH (Goldreich-Goldwasser-Halevi signature)
    - Operation
    - Good vs. Bad Basis
    - Flaw
  - GPV Signature Scheme
    - Improvements
    - Key Components
      - Public Key
      - Secret Key
      - Hashing message
      - Signature generation





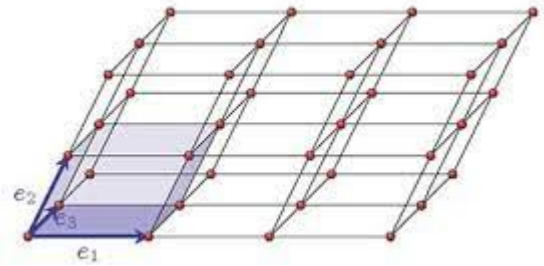
Squirrel Family

- Construction & Reliance
- Public and Secret keys
  - Key pair generation using public key
- Signature Generation
- Verification



D 3.1 Advantages

- Squirrel bases it’s security on generic lattice problems
- It does not rely on lattices that do not have strong geometric properties

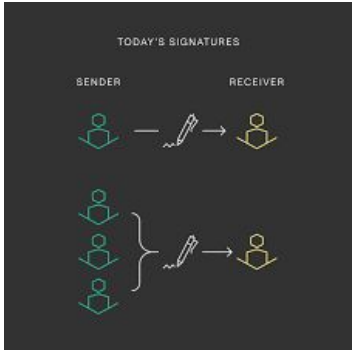


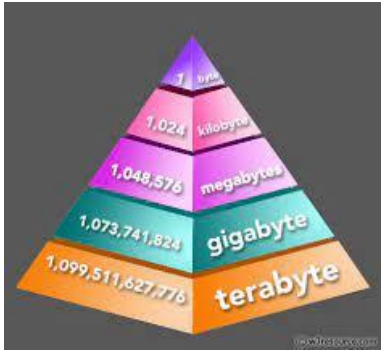
Lattices without strong geometric properties: Is more irregular and chaotic in its point distribution. This randomness and complexity make it more resistant to attacks

Lattices with strong geometric properties: Often have regular, easily discernible patterns, making them more vulnerable to certain types of attacks

D 3.1 Advantages

- Allows compact Signatures - Even though squirrels uses a complex problems it is still able to create small signatures.
- The size of the signatures are in bytes similar to the Falcon and Dilithium signatures (These two signatures are well known methods for creating the smallest signatures in post-quantum world)





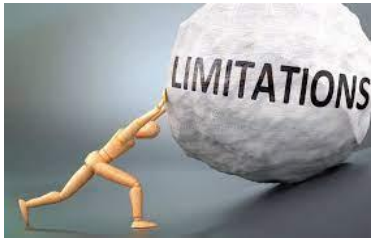
D 3.1 Advantages

- Squirrel is competitive in terms of signature generation and verification efficiency
- Even on a small laptop it is capable of generating several dozens to hundreds of signatures in a single second
- Checking if a signature is valid is easy, SQUIRREL does a quick math equation in the form of a single linear equation
- The process of verifying signatures is simple and straightforward without making the security less strong

D 3.2 Limitations

- With SQUIRRELS making keys takes time because of the complicated math involved with big matrices like calculating determinants and using something called the Hermite Normal Form.
  - HERMITE NORMAL FORM is used to simplify a matrix. Hermite Normal Form is equivalent to Row Echelon Form in linear algebra.

$$A = \begin{pmatrix} 3 & 3 & 1 & 4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 19 & 16 \\ 0 & 0 & 0 & 3 \end{pmatrix} \quad H = \begin{pmatrix} 3 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 19 & 1 \\ 0 & 0 & 0 & 3 \end{pmatrix} \quad U = \begin{pmatrix} 1 & -3 & 0 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -5 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



D 3.2 Limitations

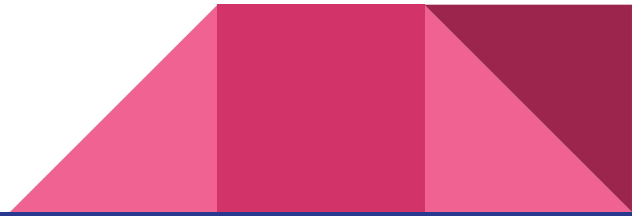
- Depending on how secure you want the keys it will take several seconds to just make 1 key pair but if you change keys often it can be a problem because SQUIRREL works with unstructured lattices so it takes more and more time as the lattice and gets bigger

Unstructured lattices have points arranged in a more random or irregular fashion within a multi-dimensional space. So it is believed to offer stronger security against attacks.

SVP - BKZ

- Assess the problem
- SVP (shortest vector problem)
- Oracle SVP (Database)
- Relies on BKZ algorithm
- BKZ can access oracle

- Block Korkine-Zolotarev (BKZ) algorithm
  - Lattice Reduction
    - Purpose is to reduce the basis of a lattice.
    - Aim to transform this basis into a more "structured" form
    - More apparent properties.
  - Block Algorithm
    - Operates on a block of vectors at a time
    - Determines the number of vectors to be processed in each step
  - Efficiency and Complexity
    - Time complexity depends on the block size
    - Smaller block sizes are more efficient
      but provide weaker lattice reduction
    - Larger block sizes offer stronger reductions
      but require more computational resources



Attack

- How an attack can be made
  - The Key Recovery Attack Aims at finding(at least) one the short vectors of the secrets basis from knowledge of the public key
  - lattice reduction on a public basis
- Projection of such attack
  - Examines lattices formed by public basis
  - Finds vectors from secret basis by listing all possible vectors less than g min
  - Avoids listing all vectors by restricting search to a projection

Cont.

More Precisely:

$$\ell = \mathrm{covol}(\mathcal{L})^{\frac{1}{n}} \cdot \delta_{2B+2-n} \approx \mathrm{covol}(\mathcal{L})^{\frac{1}{n}} \cdot \left(\frac{B}{2\pi e}\right)^{1-\frac{n}{2B}}$$

Assuming that secret vectors behave as random vectors of norm $g_{\min}$, their projection on P is roughly:

$$\|\pi_P(\mathbf{b}_i)\| = \sqrt{\frac{B}{n}} \cdot g_{\min}$$

Thus, we will retrieve the projection among the sieved vectors if $\sqrt{\frac{B}{n}} \cdot g_{\min} \leqslant \sqrt{\frac{4}{3}}\ell$, that is if the following condition is fulfilled:

$$g_{\min} \leqslant \sqrt{\frac{4n}{3B}} \cdot \mathrm{covol}(\mathcal{L})^{\frac{1}{n}} \left(\frac{B}{2\pi e}\right)^{1-\frac{n}{2B}}. \tag{3}$$

D 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 attack has to find the position of non-zero elements in the secret.

D 4.3 Hybridizing the Attack for sparse secrets

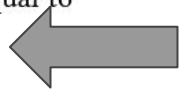
- A good guess is the only thing we need to search for secret keys outside these positions by intersecting the public lattice with the complement set of I.
- To hybridize the attack there are two main components.
  - Guessing Sparse Vector: You guess the positions of zeros in a sparse vector within the lattice. When you guess correctly, it reduces the search space.
  - Lattice Reduction: You then apply lattice reduction on the reduced lattice, which is faster because it's smaller. Lattice reduction transforms the basis of the lattice into a more structured form, making it easier to find shorter vectors.

- Hybridizing these strategies makes the attack more efficient.

D 4.3.1 Good Guess probability estimation

4.3.1 Good guess probability estimation.

Before further ado, we need an estimate of the probability of making such successful guesses of the zero coefficients. Set the sparsity level of the first secret vector to be $0 < \kappa < n$ and that $|\mathcal{I}| = g$. Then, over the randomness of the secret, the probability of getting a correct guess for a given secret vector v_i is equal to the probability of I being a subset of the set of zeros of v_i, i.e. is $\binom{\kappa}{g}/\binom{n}{g}$.



This expression is used to estimate the probability of successful guesses of the zero coefficients

D 4.3.1 Putting it all together

- Attack Strategy
  - While a secret vector remains undiscovered:
  - (a) Randomly guess the positions I ⊂ {1, ..., n} of g zeros.
  - (b) Compute the lattice L' = L ∩ ℤI.
  - (c) Reduce L' using the BKZ (Block Korkine-Zolotarev) Algorithim.
  - (d) List all vectors with a length smaller than 4/3 times the norm of the Gram-Schmidt vector ˜bi, which is the (−B)-th Gram-Schmidt vector of the basis obtained in step (c).
  - (e) For each such vector, lift it as a vector of L' using Babai's nearest plane algorithm and check if it is shorter than gmax, then return it."
- This method describes how to recover a secret vector from a lattice

D 4.4 Additional Security Properties

- There are extra security features are called BUFF properties
- This BUFF feature called “message bound signatures”.
- Message bound signatures is a feature that ties a digital signature to a specific message making it so the signature only applies to that message
  - Advantages
    - Signature Uniqueness
    - Message Integrity
    - Prevents Reuse
    - Useful for Verification

Private Key

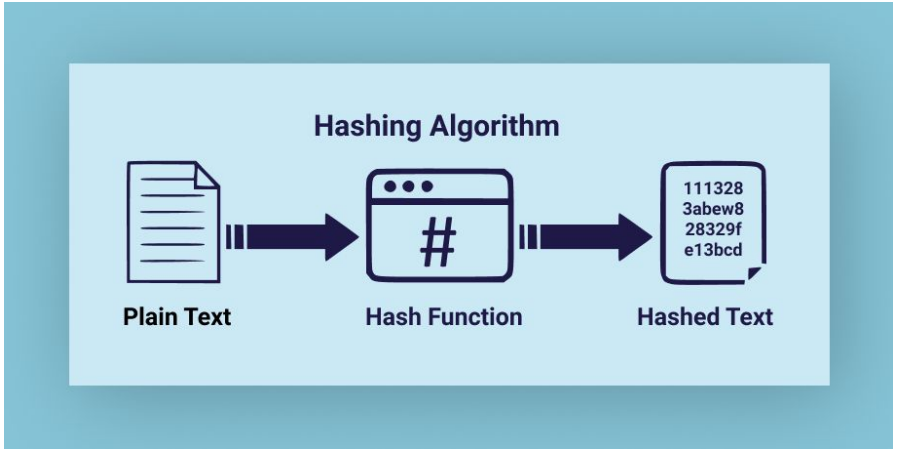
- The private key is a matrix of size n × n, each row is a vector of B
- Must follow these 3 requirements:
  - Vectors in B must have norms within the bounds defined by g-min and g-max, ensuring they are neither too small & too large.
  - det(B) = Δ, vector norm value
    - Used as a security parameter
  - The private key matrix B is co-cyclic
    - Leads to faster matrix-vector multiplications

Public Key

- Starts with value called “vector check”
- The formula performs modulo operations on various prime numbers from P
- “i” , last vectors, iterates through the values from 1 to n - 1
- For each value of "i," the modulo operation is performed, resulting in finding the public key.

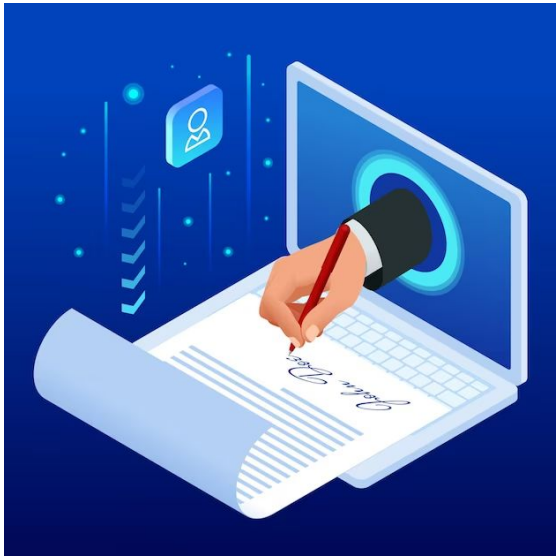
$$\mathrm{pk} = (\mathrm{v}_{\mathrm{check},i} \bmod p)_{1 \leqslant i \leqslant n-1, p \in P_\Delta}$$

Hash-and-Sign Signature Scheme



- Messages are hashed before signing and verifying signatures
- In Squirrels, messages get hashed to become a vector
- The hash function that is used is called “HashToPoint”
- The function converts the message into a vector, turning parts of the hashed message into numbers that fall in a certain range

Signature Generation



- The message "m" is hashed, along with a salt value "r," resulting in a point "h."
  - “Salt Value” is a random value that is used to enhance the security and uniqueness of the signature
- Squirrels then uses Klein's trapdoor sampler to identify a lattice point that is in close proximity to "h."

Signature Verification

- First, we recalculate the hash point, "h", using the original message, "m", and salt, "r," *HashToPoint does this*
- Next, we check to see that the signature "s" is within the expected range.
- The final step involves using the public key to verify that the sum of the signature "s" and the hash point, ‘h’, (s + h), is part of the lattice structure.
  - This step confirms the signature's accuracy and its association with the original message.

HashToPoint Function

Algorithm 8 HashToPoint(m, q, n)

Require: A message m, size of hashing space $q \leqslant 2^{16}$ (assumed to be a power of two), dimension n

Ensure: A vector $\mathbf{v}$ in $[0, q-1]^{n-1} \times \{0\}$.

```
ctx ← SHAKE-256-INIT()
SHAKE-256-Inject(ctx, m)
for i = 1, ..., n − 1 do
    t ← SHAKE-256-Extract(ctx, 16)
            ▷ Sample a 2-byte number, interpreted with big-endian convention
    v_i ← t mod q                ▷ Uniform in [0, q − 1] as q is a power of two
end for
v_n ← 0
return v
```

Klein Trapdoor Sampler

Algorithm 9 KleinSampling(sk, $\mathbf{t}, \sigma$)

Require: A private key sk, a target vector $\mathbf{t} \in \mathcal{Z}^n$

Ensure: A lattice vector close to $\mathbf{t}$, with distribution $D_{\mathcal{L},\sigma,\mathbf{t}}$

$\mathbf{t}_n \leftarrow \mathbf{t}$

$\mathbf{v}_n \leftarrow 0$

for $i = n, ..., 1$ **do**

$d_i \leftarrow \langle \mathbf{t}_i, \tilde{\mathbf{b}}_i \rangle / ||\tilde{\mathbf{b}}_i||^2$

$\sigma_i \leftarrow \sigma / ||\tilde{\mathbf{b}}_i||$

$z_i \leftarrow \text{SamplerZ}(d_i, \sigma_i)$ ▷ Gaussian sampling in $\mathbb{Z}$ with mean d_i, std σ_i

$\mathbf{t}_{i-1} \leftarrow \mathbf{t}_i - z_i \cdot \mathbf{b}_i$

$\mathbf{v}_{i-1} \leftarrow \mathbf{v}_i + z_i \cdot \mathbf{b}_i$

end for

return v_0

Sign Algorithm

Algorithm 14 Sign(m, sk, $\lfloor\beta^2\rfloor$)

Require: A message m, a secret key sk, a bound $\lfloor\beta^2\rfloor$
Ensure: A signature (r, s) of m
  $r \leftarrow \{0,1\}^{320}$ uniformly
  $\mathbf{h} \leftarrow$ HashToPoint$(m||r, q, n)$
  while true **do**
    $\mathbf{c} \leftarrow$ KleinSampling$(\mathbf{sk}, \mathbf{h}, \sigma)$
    $\mathbf{sig} \leftarrow \mathbf{c} - \mathbf{h}$
    if $||\mathbf{sig}||^2 > \lceil\beta^2\rceil$ **then**
      continue ▷ If signature produced is not short enough
    end if
    $s \leftarrow$ Compress$(\mathbf{sig}, \text{sig}_{\text{size}} - 41, \text{sig}_{\text{rate}})$ ▷ 40 bytes for salt r, 1 byte for header
    if $s \neq \perp$ **then**
      return (r, s)
    end if
  end while

Verify Algorithm

Algorithm 15 Verify$(m, (r, s), \text{pk}, \lfloor \beta^2 \rfloor)$

Require: A message m, a signature (r, s), a public key $\text{pk} = (\text{v}_{\text{check},i} \bmod p)_{1 \leqslant i \leqslant n-1, p \in P_\Delta}$, and a bound $\lfloor \beta^2 \rfloor$

Ensure: Accept or reject

$\mathbf{h} \leftarrow \text{HashToPoint}(m \| r, q, n)$

$\mathbf{sig} \leftarrow \text{Decompress}(s, \text{sig}_{\text{size}} - 41, \text{sig}_{\text{rate}})$ ▷ 40 bytes for salt r, 1 byte for header

if $\mathbf{sig} = \perp$ **then**

  return "reject"

end if

$\mathbf{c} \leftarrow \mathbf{s} + \mathbf{h}$

result $\leftarrow$ "accept"