My section will cover the first half of Chapter 3 which is the heart of the paper.

A sigma protocol is an interactive proof of knowledge where a prover needs to prove that it knows some piece of information. Zero knowledge means it needs to prove this without revealing the information itself. As an example, consider Alice and Bob where Alice knows the password to a locked door in a cave with 2 paths. She randomly chooses a path. Bob, who doesn't know which path she took, randomly shouts out one of the paths. If Alice is able to reliably return to the entrance using the path Bob randomly shouted out when this experiment is repeated many times, it is extremely likely that she knows the password to open the door in the cave that allows her to appear at either side. Note that she never reveals the password.

Endomorphism rings will be important for the digital signature itself so let's review what they are. Given an elliptic curve, an endomorphism is an isogeny (a mapping) where the domain and the codomain consist of the same elliptic curve. The collection of all of the endomorphisms themselves along with addition and noncommutative multiplication form the endomorphism ring of an elliptic curve. So basically, you are adding and multiplying functions themselves. You can see here how these are defined. In particular, notice how function multiplication is defined as function composition which is a noncommutative operation. That is why we consider the ring instead of the field of endomorphisms.

The sigma protocol involves three interactive phases of commitment, challenge, and response. The public key is the elliptic curve E_A and the private key $End(E_A)$ which is the endomorphism ring of E_A. Basically, the prover randomly generates an E_1 and its endomorphism ring and sends E_1 to the verifier. The verifier randomly generates an isogeny from this E_1 to an E_2 and sends it to the prover. The prover uses this and $End(E_1)$ that it initially generated to compute the $End(E_2)$. It then uses this and the private key to respond with the isogeny from E_A to E_2. The verifier can easily use the public key, E_A, and the isogeny and see if it correctly maps to E_2.

The Fiat-Shamir Heuristic transforms this sigma protocol into a non-interactive proof of knowledge which is important if we want to separate the signer from the verifier which is needed for Digital Signature Schemes. This is accomplished by letting the prover compute its own challenge using a hash function whose output is deterministic but cannot be predicted.

Now let's discuss key generation. Here you can see the formula that will be used to find a secret ideal. O_0 is a maximal order meaning it is the largest possible proper subset of the quaternion algebra that also forms a ring with addition and multiplication. Gamma is a random element of O_0, a is a random positive scalar less than D_{secret}, i is the quaternion element such that $i^2 = -1$. D_{secret} acts as the norm of I_{secret} meaning it is the gcd of the elements of

I_{secret}. Then alpha which connects I_{secret} to an equivalent ideal with a power of 2 norm called J_{secret} is computed.

Here you can see the computation of the secret isogeny from the secret ideal, the maximal order, and the basis for $E_0[T]$. $E_0[T]$ is the T-torsion subgroup of E_0 meaning the set of all points on E_0 such that scalar multiplication with T yields infinity, the identity element. In SQI_{SIGN} we assume T is smooth meaning it has only small prime factors. For this case, an algorithm like the Pohlig-Hellman algorithm guarantees an efficient solution. The public key, E_A, is generated and φ_{secret} is modified to map to this normalized curve. Then $B_{A,\,T}$ (a basis for E_A's T-torsion subgroup) is found by applying this secret isogeny to $B_{0,\,T}$.

Here P is one of the basis points for the intersection of the kernel of the secret isogeny and the 2^f-Torsion subgroup of E_0. The other basis point, Q, is found using the CompleteBasis algorithm. The generating point Q is mapped from where it is on E_0 to a point on E_A. After a few more steps, the signing key and public key are returned and the key generation phase is complete.