DNSSec Implementation

The domain name system (DNS) is a database where all the IP addresses are mapped to their domain names so one can use the domain name to access information. However, since it does not provide any layer of security, it is vulnerable to attacks and other security issues. Keeping these issues in mind, another protocol called DNSSec was developed, which added a layer of security to the existing protocol. The implementation of DNSSec in the file dnssec_resolver_part_b.py is explained now:

A total of 9 functions have been used to code the implementation. A detailed description of each function is given below:

1. getrootserverlist():- In this function, we return the list of all the root servers.
2. TwoStepValidation(): This function takes the input of domain, hash, dslist, RRSig, and RRSet. Here we perform two steps. The first step is to check whether the hash is in the dslist or not. If it is not, the function returns False, and a 'DNS verification failed' message. If it is found, another step is performed where we check for validating the public key. This function returns a Boolean argument.
3. get_nextlevelservers(): This function takes as an input the domain and the server to get the next level domain. After sending a UDP query to the domain and the server, the response is further used to extract data. Here different conditions and edge cases are used to handle the query. It outputs the response, the child DS, and the hash function.
4. sendqueryudp(): It takes as an input the domain, DNS Type, the rootserver, and a DNSSEC flag which indicates a DNSSEC request and returns a response as an output
5. Get_RRSig_RRSet_Key(): This function again takes the domain and server as input which is used to fire a query and later uses the response received to extract the RR Sig, RR Set, and KSK
6. populate_nextlevel_servers(): This function is used to call get_nextlevelservers() using which we can use to make a list of next level servers.
7. resolve(): This is the main function of the code where the actual resolving takes place. It takes as an input the domain and outputs the current level servers. The resolving is done in 2 steps. In the first step, we handle the root, and in the next step, we iterate over the higher levels of the domain.
8. rootserverlooper(): In this function, we input the domain and iterate over the output of resolve to send a UDP query and return the successful result
9. mydig(): This function prints the resolved final IP

First, the user enters a domain that goes to the resolve function via mydig and rootserverlooper. In the resolve first, the root is handled by extracting the RRSign,

RRSet, and KSK. The KSK is then hashed using the hashing algorithm of the root('sha256'), which is further used in the TwoStepValidation where the hash verification and validation of the public key take place, which, if validated further moves on to resolving the higher level zones/servers. Now in the higher zones, similar to the root, we first extract the RR_Set, RR_Sig, and the KSK, which is then validated. The validation Is done by checking if the hash(hash of the next server done using the private key of the server) is present in the dslist or not. Once the validation is done, we move on to the next server, and this process repeats.

This way, an extra layer of security is added because the hash done by the server can be verified using the public key available to the user using, which the user can validate that it has received the response from the correct server.