1. Related functions:
    a. **issue_dnssec_request**:
        i. Set *want_dnssec=True* in the query.
        ii. Set a **larger payload size**, to prevent it from truncated data.
    b. **verify_zone**:
        i. Usage: compare the **Hash(sub zone's public key)** with the **parent zone's DS** record.
        ii. How: transform **sub zone's DNSKEY** record to a **DS record**, then compare if it is the same as the **parent zone's DS record**.
        iii. If the verification is successful, save the **zone_name**(eg. '.', 'com.', ...) **and DNSKEY pair** into **key_dict**.
    c. **verify_record**:
        i. Find **RRSIG** record in a DNS response.
        ii. Find the **cover** of i.'s RRSIG records(for example, given an RRSIG NS record, its cover is NS), then find records whose type is of this cover, let's say **RRSET**.
        iii. Then validate the **RRSIG** and **RRSET** above using the KSK of the zone itself, which has been stored in **key_dict**, through *dns.dnssec.validate*.
    d. **authenticate**:
        i. See explanation in 4.
    e. **sec_resolver**:
        i. Main implementation. This function will not do many conditional checks, neither set bunches of flags to indicate the multiple states, it just naturally throws all kinds of exception and let them be caught in the main function. In general, states are distinguished by types of exceptions.
2. For root servers:
    a. **Anchors** are fetched from https://data.iana.org/root-anchors/root-anchors.xml and hard-coded at the top of the python file as two 'DS' records.
    b. **Verify zone**: hash its DNSKEY using the anchor-defined algorithm(SHA256), then compare with the anchor.
    c. **Verify record**: verify its RRSIG of DNSKEY
3. For the following servers:
    a. Issue a dnssec query towards the queried domain, from which we can get non-secured referrals, DS, and their related RRSIG. Let's denote it **prev_resp**.
    b. Check if DS records exist in the **prev_resp**. otherwise, it does not support DNSSEC.
    c. Query its **sub zone's DNS key**.
    d. **Verify zone**: compare **Hash(sub zone's key)** with **DS** record in **prev_resp**.
    e. **Verify record**: verify **sub zone's RRSIG** using **sub zone's own key**.
    f. **Verify record**: verify **prev_resp's DS RRSIG** using **sub zone's public key**.
4. The general verification steps in 3 are integrated into a function **authenticate**. These authentication steps will be applied to every hop of the DNS query.

5. Example output:

```
QUESTION SECTION:
verisigninc.com         IN  A

ANSWER SECTION:
verisigninc.com. IN A 209.131.162.45
verisigninc.com. IN A 69.58.187.40
verisigninc.com. IN RRSIG A 8 2 60 20211024231946 20210924231946 20844 verisigninc.com. rd1Dnw82ZP/

Query time: 334 msec
WHEN: Sat Sep 25 10:10:39 2021

MSG SIZE rcvd: 444
```

```
/usr/local/bin/python3.8 /Users/evenyoung/Desktop/cse534/hw/hw1/dnssec_resolver.py dnssec-failed.org A
DNSSEC not supported
```