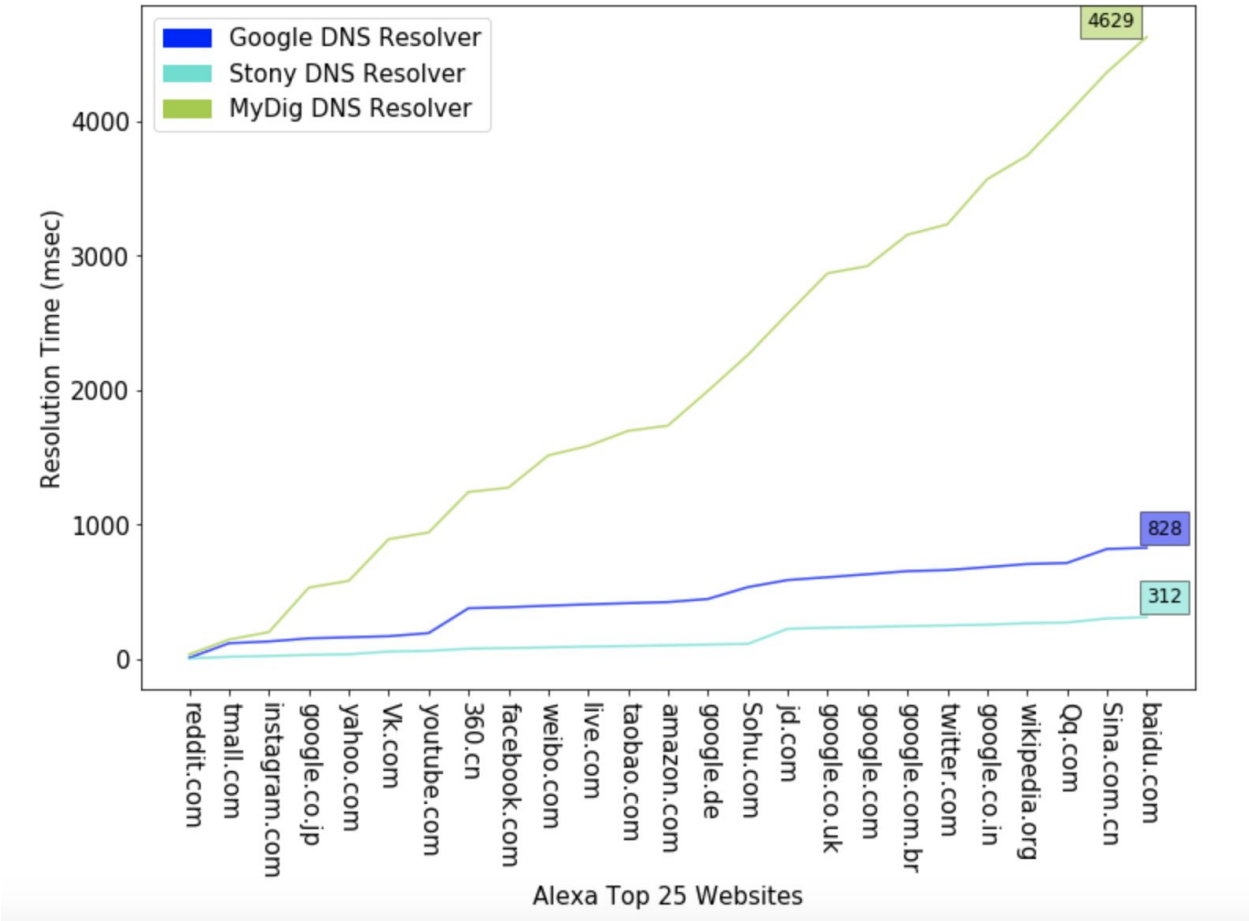
Assignment 1
Fundamentals of Computer Networks

Name: Jatin Garg
SBU ID: 111462753

Questions:3) Experiment on Alexa top 25 websites.

So we first noted the query resolving time of all these 25 websites, 10 times each and find the average time of a query on Google name Server, Stony Brook name server and my personal resolver. Here is the CDF of the results.



Below is the code excerpt to produce this graph.

```
googleServerListCum = np.cumsum(googleServerList)
stonyServerListCum = np.cumsum(stonyServerList)
mydigListCum = np.cumsum(mydigList)
blue = mpatches.Patch(color = 'Blue',label="Google DNS Resolver")
yellow = mpatches.Patch(color = 'turquoise',label="Stony DNS Resolver")
green = mpatches.Patch(color = 'yellowgreen',label="MyDig DNS Resolver")
plt.legend(handles=[blue,yellow,green])
plt.plot(uniqueCount,googleServerListCum,c='blue',alpha=0.75)
plt.plot(uniqueCount,stonyServerListCum, c = 'turquoise',alpha=0.75)
plt.plot(uniqueCount,mydigListCum, c= 'yellowgreen',alpha=0.75)
my_xticks = serverNameList
plt.xticks(uniqueUser, my_xticks,rotation=270)
plt.xlabel('Alexa Top 25 Websites')
plt.ylabel('Resolution Time (msec)')
plt.rcParams.update({'font.size': 15})
plt.text(24,930,'828',fontsize=12,bbox=dict(facecolor='Blue', alpha=0.5))
plt.text(24,420,'312',fontsize=12,bbox=dict(facecolor='turquoise', alpha=0.5))
plt.text(22.5,4700,'4629',fontsize=12,bbox=dict(facecolor='yellowgreen', alpha=0.5))
fig = plt.gcf()
fig.set_size_inches(12,8)
```

Results Explanation:

1.) The Stony Brook server has the least resolution time because it is the nearest server of all 3. Also no VPN service was required to query this server, so it was fastest of all 3.

2.) The Google server was the second fastest with 828 msec across all the queries. Also this server was no closer than stony brook server, so it took a little bit more time to resolve all the dns queries.

3.) Mydig resolver had the highest resolving time of all 3 with around 4.8k msecs. So one reason is that there is no cache used in Mydig resolver so it has to perform all queries from scratch whereas other resovers has cache implementation. So the successive queries resolutions was quite fast.

4.) Also Mydig resolver has to use VPN service to get the response from root servers. So as I noticed that due to the VPN the normal internet speed was getting slower.

Question:1)c.) In some cases, you may need additional resolution.For example, google.co.jp will often not resolve to an IP address in one pass. Extend your tool to address this corner case. Also write an explanation for why the resolution did not complete in one pass in this corner case.

Sol.) So for these corner cases, instead of getting the IP address for the authoritative name servers, we just get the name of the name servers in the Authority Section. The current server does not have the IP address of these name servers. So we first have to resolve the IP address of those name servers. After resolving the IP address of these name servers we proceed for the normal resolution of the original request. That is why an extra resolution is required.
Example is shown below.

```
;QUESTION
google.co.jp. IN A
;ANSWER
;AUTHORITY
google.co.jp. 86400 IN NS ns3.google.net.
google.co.jp. 86400 IN NS ns1.google.net.
google.co.jp. 86400 IN NS ns4.google.net.
google.co.jp. 86400 IN NS ns2.google.net.
;ADDITIONAL
```

These are the results got when I queried for google.co.jp. So first I had to resolve IP of these ns3,ns1 etc name servers.

Question.2) Please explain in detail how you implemented DNSSEC.
Sol.) I used dnspython to implement the resolver that also verified DNSsec at each level of resolution. **The steps are:**

1. Special Root Verification: I stored the KSK of the root servers so as to validate the DNSKEY rrsets that I received from the root. That is the beginning of the Chain of Trust.
2. So I queried the DNSKEY rrsets from the root level server. I got:
    a. **Key Signing Key:** This come with a tag of 257. This is validated with already stored KSK's stored in case of root server. Otherwise it is validated with the DS record provided by parent zone. So if this validation is successful, we moved to next level.
    b. **ZSK:** Zone signing key is received in the same rrset with KSK.
    c. **RRSIG**: This is the signature of rrsets. So we validated this with DNSKEY received by querying the server.
3. Then at same level, I queried the server for its DS records. We also get its RRSIG record. So I stored the DNSKEY that I talked about in second point that contained the ZSK. I used the ZSK to validate its RRSIG records. If this is validated it means we can trust the corresponding rrset record.
4. Now the DS record received in previous point is stored and then used to validate the child DNSKEY(KSK) so as to transfer from one zone to next zone in the hierarchy.
5. The above steps are done till the IP is resolved or on error conditions are met.

Error Conditions:

1. So 3 validations are done at each step. If any one of validation fails then that means the record are tampered and hence we can not trust the records. So I stopped the resolution.
2. If the server does not provide DS records or DNSKEY records then it means the domain does not use DNSSEC and hence "**DNSSEC is not implemented**" is outputted.
3. If nothing goes wrong, I provided the resolved IP address.