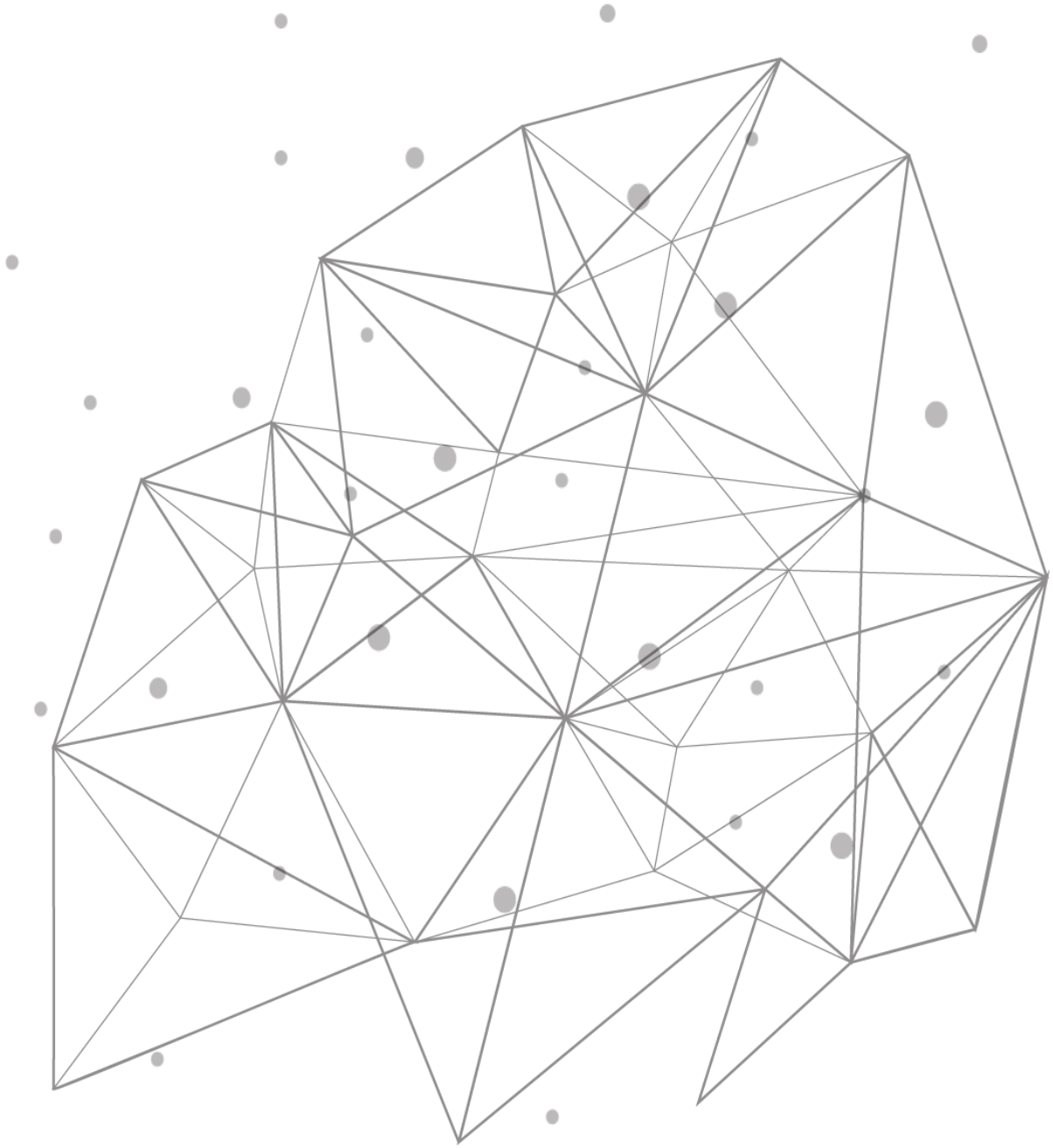
Simplify DNSSEC with TCPWave



Introduction

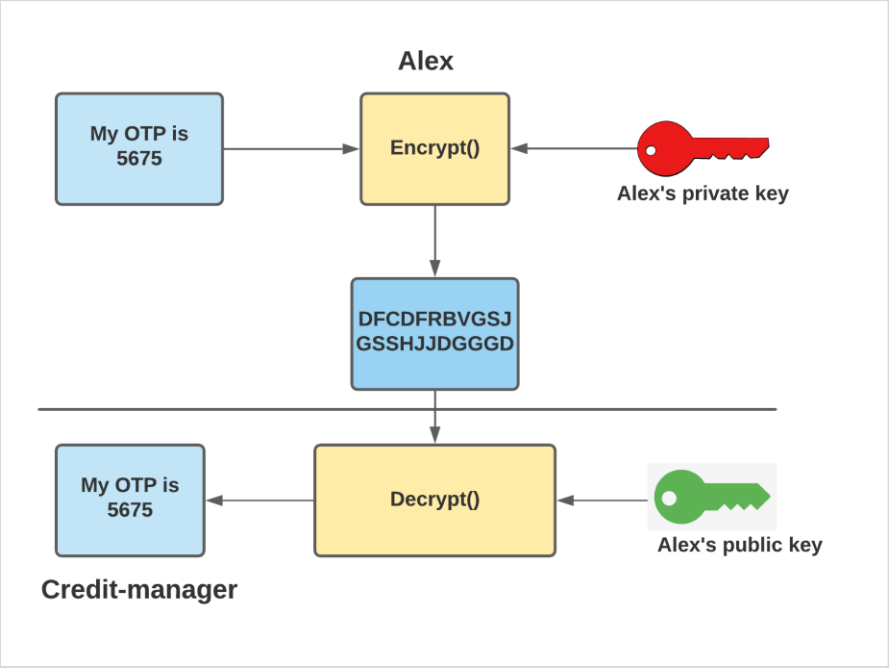
Paul Mockapetris did not have security in mind while inventing DNS in 1983; the security flaws discovered were a significant concern. In 1990 Dr. Steven M. Bellovin, Computer Scientist well known for his several RFC contributions, identified many DNS Security flaws, after which he published a paper on this by around 1995. By 1997, the IETF published an initial RFC 2065 to address those vulnerabilities. Following several revisions and updates, RFC 4033 and RFC 4035 became a standard followed by the Internet. Finally, by around July 2010, the DNS root zones were signed & published for the first time.

What is DNSSEC?

DNS Security Extension (DNSSEC) reinforces validation in DNS using digital signatures based on public-key cryptography. The DNS data gets cryptographically signed by the data owner to prove its authenticity. In addition to what existed in regular DNS without any security measures, DNSSEC introduced a public/private key pair to every DNS Zones.

DNSSEC uses asymmetric key cryptography, where the system uses two separate keys, one is public & the other is private. When DNSSEC is configured, the server uses the zone's **private key** to sign the DNS resource records in that zone & would also generate digital signatures for that data. The zone's public key would be published & a recursive resolver can pull that in along with regular DNS responses from this authoritative server. This public key eventually could be used to decrypt the data & validate authenticity. If the recursive resolver can validate the data's authenticity using the public key, the DNS response would be cascaded down to the actual requestor. If the data's signature isn't valid, the recursive resolver shall return a SERVFAIL response to the requestor.

Public Key Cryptography – In a nutshell



- Alex's Public/Private keys form a Key pair.
- The private key is never exposed & only the sender will have it.
- Alex's Public key is made available to the receiver by any ways.
- Alex will encrypt the data using his private key to sign data. If the data decrypts with the corresponding public key, it helps the recipient authenticate the sender.
- This system is leveraged in DNSSEC to validate resource records' authenticity.

DNSSEC – Common Nomenclature

- **Hash Function**: A one-way math function that processes information of indefinite length to create a different, fixed-length result that is always unique to the original input.
- **Digest**: Output of the hash function.
- **Fingerprint**: The hash/digest of a public key.
- **KSK – Key Signing Key**: Key used to sign the DNSKEY RRs in a zone. KSK has a public & private key to form the KSK Key pair.
- **ZSK – Zone Signing Key**: A key used to sign all data in a zone. ZSK has a public & private key to form the ZSK Key pair.
- **Trust**: To accept the truthfulness of a DNS entity with no need to validate further.
- **RRSet – Resource Record Set**: Set of records with the same type from the same zone.
- **RRSig – Resource Record Signature**: A record containing RRSet's digital signature.
- **DNSKEY**: The DNSKEY record is the zone's public keys. Clients will retrieve this record to verify the signatures that they receive.
- **DS Record – Delegation of Signing**: A record containing the digest of a child domain's Public KSK. DS records are generally used to establish trust between parent & child.
- **NSEC/NSEC3 Record**: In non-secure DNS, the nameserver returns no records when a name doesn't exist. No records mean nothing to sign & no signatures to be generated. The NSEC records are a mechanism to allow DNSSEC to authenticate the non-existence of a name by returning the next Secure record (NSEC) in the zone (which is signed).

How Does DNSSEC Work?

When we enable DNSSEC on a specific authoritative zone or all, the DNS server will sign all the resources records within that zone with a key called the Private-ZSK. We'd use the Public-ZSK for that zone to validate the signatures generated. Now the Public-ZSK would further be signed by another key called the Private-KSK. To validate the signature created by this second signing, we'd use another key called a Public-KSK. All public keys along with the signatures are published within a zone & someone can pull them by

using this simple DNS request:

```
dig @<server_name> <zone_name> dnskey
```

We've read so far about the changes to a DNSSEC enabled zone itself. Signing a zone wouldn't ensure that an upstream server would validate the data. To reach there, the parent domain should have a link to complete the chain of trust.

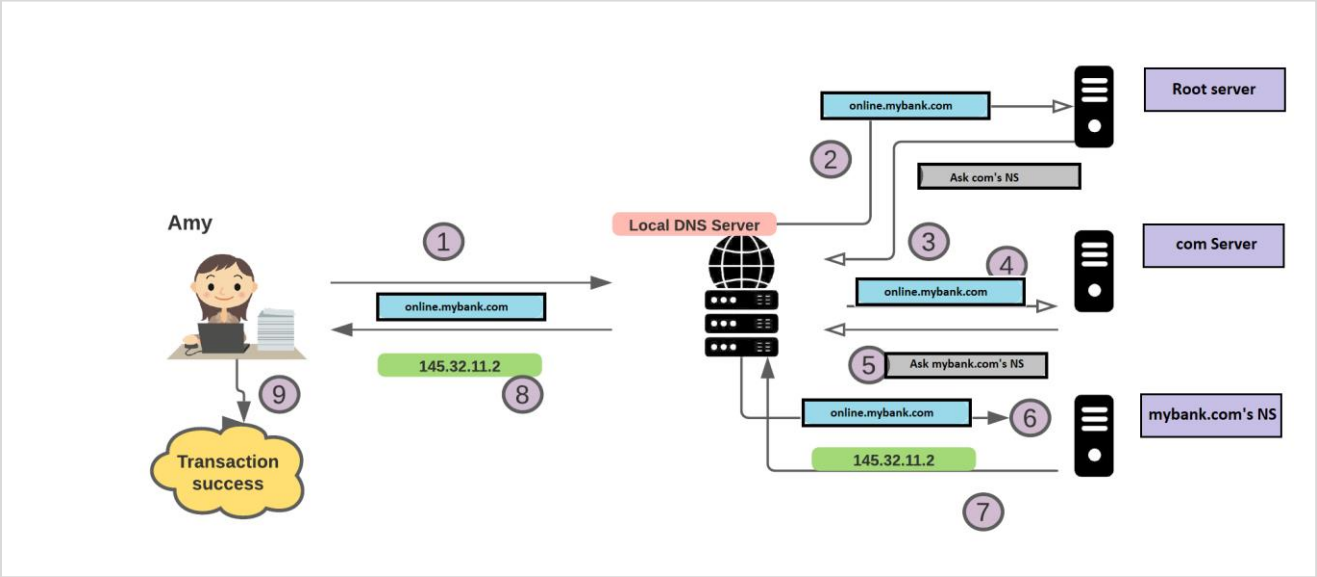
To achieve this, a hash of the child zone's Public-KSK is placed in the parent zone (called often the DS record). For example:

***tcpwave.com**'s DS record must exist within **com**'s authoritative servers to complete the chain of trust*.

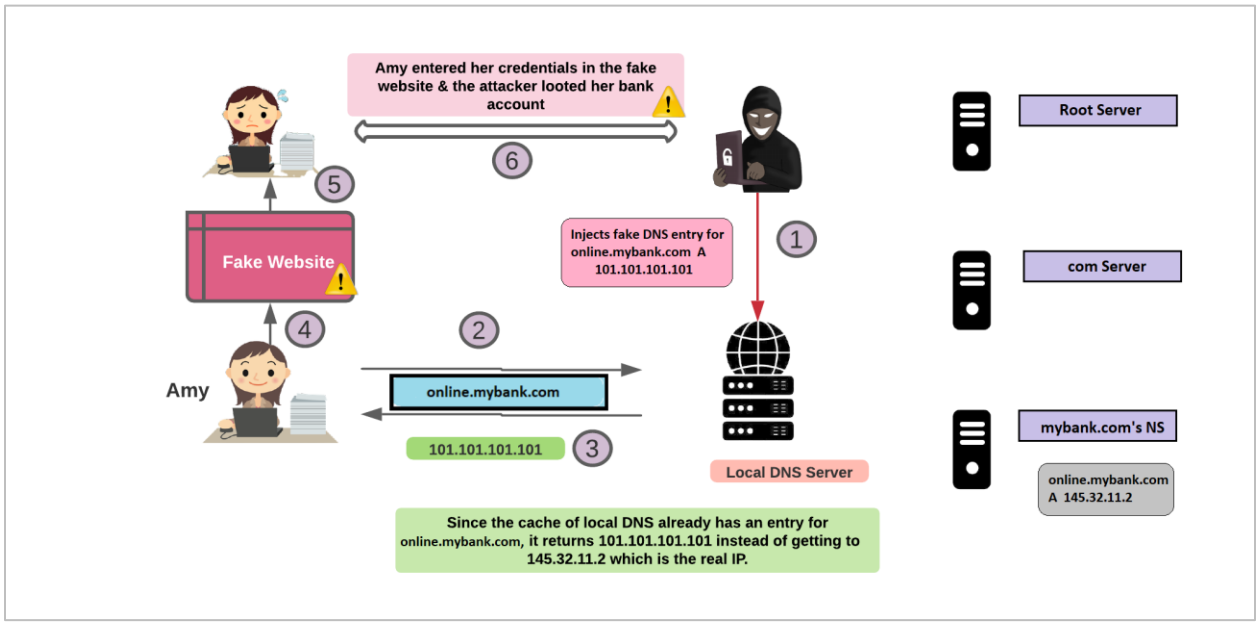
Consider the following example to understand DNSSEC validation:

1) Amy decided to log in to her bank account & make an online transaction.
2) Amy tries to access https://online.mybank.com from a browser & this is what she'll go through:

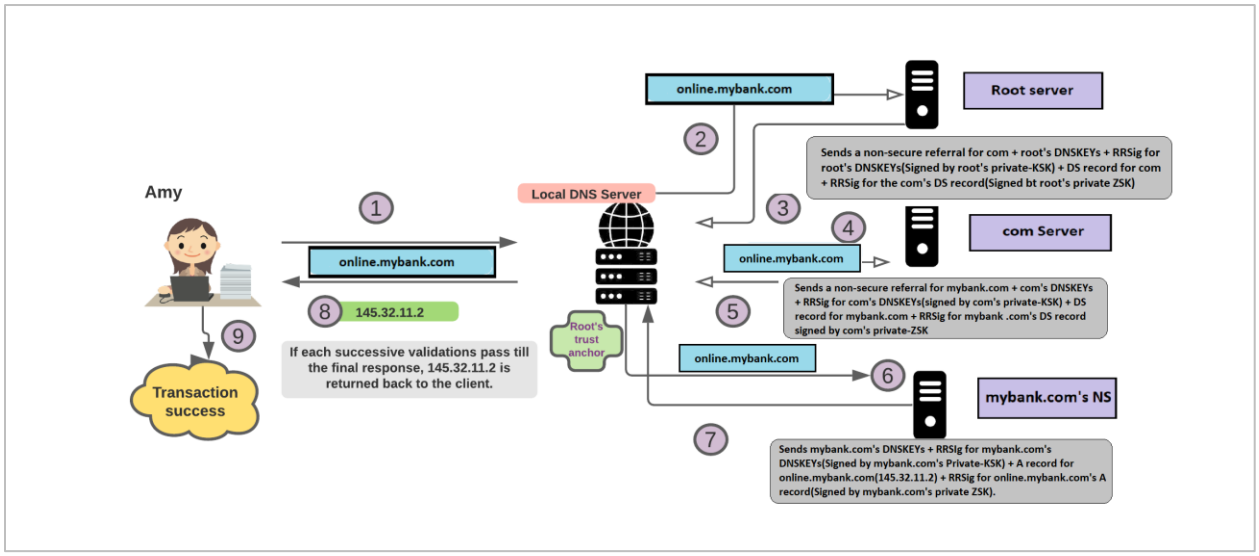
Normal resolution when DNSSEC is disabled:



Now see how Amy's system gets compromised without DNSSEC protection:



How DNSSEC will protect Amy:



From the diagram above, at steps #3, #5 & #7, the DNS server would perform a set of validations as explained below:

At Step #3:

- The recursive local DNS server validates the root zone's DNSKEY set by decrypting its RRSig using the root zone's PublicKSK(That it has as the trust anchor). The trust anchor can be shipped with the operating system or added manually by a DNS Administrator.
- The recursive DNS Server will verify the root zone's DS record for the com zone by decrypting its RRSig using the root zone's PublicZSK(Gathered from the verified DNSKEY set from above).
- It compares the public KSK with what it see's for the same record within the DNSKEY set.
- If all of the above checks are cleared to be positive, then the DNS server moves to Step #4 in the diagram above.

At Step #5:

- The recursive local DNS server will validate the com zone's DNSKEY set by decrypting its RRSig using the com zone's PublicKSK (That it received & validated from Step #3).
- The recursive DNS server will verify the com zone's DS record for the mybank.com zone by decrypting its RRSig using the com zone's PublicZSK (Gathered from the verified DNSKEY set from above).
- It verifies the com zone by comparing the hash or digest of the com zone's PublicKSK with the previously obtained DS record from the root zone for the com zone.
- If all of the above checks are cleared to be positive, then the DNS server moves to Step #6 in the diagram above.

At Step #7:

- The recursive local DNS server will validate the mybank.com zone's DNSKEY set by decrypting its RRSig using the mybank.com zone's PublicKSK(That it received & validated from Step #5).
- The recursive DNS server verifies online.mybank.com's A RR set by decrypting its RRSig using its PublicZSK(Verified above).
- It verifies the mybank.com zone by comparing the hash or digest of the mybank.com zone's PublicKSK with the previously obtained DS record from the com zone for the mybank.com zone.
- If all of the above checks are cleared to be positive, then the DNS Server moves to Step #8 in the diagram above & Amy is finally able to access her bank account securely.

Zone - Validation

Assume that **tcpwave.com** is an unsigned zone managed by a raw Linux machine. Assuming that the reader understands regular bind zone configurations & zone databases, here's how its zonedb would appear without DNSSEC:

```
dnsadmin@dnsadmin-virtual-machine:~$ cat /etc/bind/db.tcpwave.com
$TTL 86400

@ IN SOA supportstaffs tech.supportstaffs (
  2018050600
  3600
  900
  604800
  86400
)

@      IN NS server
server IN A  11.11.22.22
www IN A  88.88.88.8
```

When tcpwave.com is DNSSEC signed, two key pairs are generated. They are the public, private keys of ZSK & KSK, respectively. If data is signed with the private key, it can only be decrypted using its corresponding public key. Now the process of signing means that the private key and the data to be signed would through a sequence of steps or algorithms which will end up in new data. Now the new data & the corresponding public key could go through the algorithm & that'll result in the original data. The new data is called the hash of the old data. The DNSSEC signing algorithm is just a sequence of instructions more like a mathematical formula. To understand what a signing algorithm process is, consider this simple example:

First, generate a key pair to sign the DNS records for the tcpwave.com domain.

```
dnsadmin@tcpwave$:sudo openssl genrsa -out privatezsk.pem 4096
Generating RSA private key, 4096 bit long modulus (2 primes)
.....................................................................................
.........................+++
.........................+++
e is 65537 (0x010001)
```

Generating public key from the private key **privatezsk.pem** :

```
dnsadmin@tcpwave$:sudo openssl rsa -in privatezsk.pem -pubout -out
publiczsk.pem
writing RSA key

dnsadmin@tcpwave$:ls
privatezsk.pem  publiczsk.pem

dnsadmin@tcpwave$:sudo cat *
-----BEGIN RSA PRIVATE KEY-----
MIIJKQIBAAKCAgEAwaVDa4NxzhAMxBFFiUJEf/ZEqsZtC1zj8tDGrm/gMwW3Hf4r
nG3t0SdBldVmEz1CHvsq/20V4SY11WcHqQtvx3pdkEBHBXSCYxPeep7YFYUX4sYv
```

```
Ys3OrLvSBrsyrzEmVA7VC6d+vU4oLKGw+0leI4IlDct3TZMjntQWRsQGkilR5iyt
4werYyyHPiengaJzA3bsctBaYhyUEg4ZbIBgpCeR56y/WM+cFgiP14EhWpshKhI4
+9UCogIox8NPwpNUZDN1TV56ejjOdAfoMsMUL6azKNXTDFi0Hnhr0I526zrNCWrt
dkXNPQZdmGJHiFxUJUIAJTAu5awd1AQTtljJ3P34761aoYyCbawNRDciUPEdDTMB
cd774UwyvxmsoANzyNn/DxfgJ8Jn5JwdVLEZBXvAjocuDvjl7rZ9c9TUwB6E06vz
l7dILaEbiFqKKcX92I2V4/RjiTp6Ovk90X7+kGIsk5aW6vx6E3M2K3/UZyo7QW2k
QRBA+UgeIQeffSZ2AhHSoyhJa6fBNagOl7INThBeullVus0VaWmDEuWTo99YxIOe
CEb6IwOe2PIC24mMKpSVua2AlfX46OPAaZ5ftmmwUlY5FTNk8PFXUOf1drOzc0Qf
gA7Hq4t/AWEZD2ey0ZaCJV8IgEqKRdizNqvPMPZq8mCUZ15YJglxKg65GS0CAwEA
AQKCAgAFiAL0/zww9qRmtVeXJB1wykgL2L8ko/Ke7Pe/49IzouYoJSp5vWodn1Fw
Rjg0NHu/Bs2jnioR1FWVUuwVGA6JVCV4CV39+eX/joG8Xu0tMDAOejrAR8Cvjm3h
Y1d/SZgero3HV3F3VtQu8i9r/hyCRyVnz3gI7zsFPmmke9T0N9nvCTeuVHnAy3TR
gbB3M3y1jKAUZ38tbkOMgHqZbtFT3w174KNWlhePSY9SSI37WJMJ8JtQFjWNCyK/
EgUlObAfbJjdLOWyuAyto8wuJwRcmBZmBDWTFLWQaUArJOagsX1J15QQ3M2QAK6L
u8LvUjPVHJ2Ecmvkg5YWiEdH3iFl8eJzdK3vWQvG6gdNzN/eMMkrJwcT/Te1egHh
vTAzHXO3gQrpl8vxFPztMm6EBmisGVH8rINrrrmFVsfm29kF8yKtRVhTrVncOT1c
GO5RRhgdTIWklTtbZ1YiRn82fsn1VutJY87xOq+qC/ekMeqDkcphaeaH38zY7uw3
47dw3Vw9gRNzjwQMqsbv/tTLTdzkmgaA97WTDNiH2m/OrrO8JNKR/D/YQDCdMQYr
t30FfNqyXP7GOdLgspb5EEPCjXPVGEmfGeZiPZCk8jdHxBli13fHLs6CBQuO4GRb
uW98Ff0/+BhJDazV5QI0DAmIevc26DdA3sqf3MbejL0YRdTUAQKCAQEA4qOGckRE
6Uc5G0/AYlkYyL3aGMomC15nFjJU6j47nJg7E0/CD4FlBum47gKZiHbQVS0aXCeC
GnSmUy+u4/bJYUUVebtPPlMKF7Rfr0YfUpwvZpMlsTCa4uP3gztvtrYlRRZmxuJT
G3MuTZKJUvh35FOq6i1nAqDmgHMQAEP5OUb9y86w2nSHM4YcRPfYVD7zC/aosbvV
PAfnLE86h4fxgz0Vt+lA8JgOetCnFbv6Qenmw2kNPx7UOxxGWfHuD3pdL/3LtGlV
xaMS8vKuXzJgMg92kExx4cqBJmh90paaep7zS7jHgd9CVgjoRhFWQNIuCj4hvPaD
A+JE/6+VURV78QKCAQEA2ruEsz0yHD1mGKRgQFGNZtGIaK4hqqF8Kg4f61ZIR0p3
PrI4+EwKz3Ueo8Ajgstv9INRXVZjAL1O6Sa1HKTUQYoUntsm4CHHNyKZliYza4iU
03aO054Oy1evIqkVeXOdUYOI1SFOW29GBJZ7q/EQmVSNLQtlOL2C8rlzf8p6hI3N
8I0oquBsRcoJSC1uC7pTgcs3GHv7HbD3/LOxL8QySJd4QZwYSjsNvwvnyZPf7eYz
cu731imd+UQRI2dCgOUAC+yXp6Ajic6m5tc8VBd42eXQCpFNJJFqZtVnaffbA96K
anpwvbkqArbT95zh/aDbnahKV6bUBmUDjaJDYiNc/QKCAQAC7BLWtFSjIVz2IjAb
1IRCSlhei3S0HtJ7cciwgH7qRPg2Bog9BPEVhrvwra3nGYkUIBG/8G54AtsGJY4+
uDbefzKIsEaUSiPOLHAzZVcQDJ172BfbUgIdPM07LvAAzhZq7qACnnZt2y1UqLgG
jteU5RNGE7HJUePG8jTZnAbBm024bADRwEDiBC+nH/71Dx65H8QLDp3wOTXc12Ss
zK7rf0UhauuAbqV5JemRmSvf37EvAdK+5pnb7UBm9nyYTH6ZGM+gkt8dPj4HTeO7
rH2bB6ICuEmbnK3PnfdJWNHDFPuUFVzVKtjDHThJdv3CtjMLxMNQmWkVCrzTAyOM
uRcxAoIBAQDQoTeLdv1K7k1ph6++6YqD3avMnPrDs8+olVxgriW2yhI/HOM2AeH1
yvJ0ohm88apRgMekzXcVB3wy6papQZarBLdyZfoIom9P6ZKxIrK9CMNbh6PnuMTH
s1T7LGetNleBnoJvN5tznwkb6yT+9aotdfZKVqovmqNCPR0rcMe3k0IxBZMamYDs
eC6uR5bFXO21yU0971OwGFIRGJwSwYJstv0fJ3ILq5QK+6IUJRXb5kyAfeqqvXtO
fFwUdrN6GfMXbI2idx4unYuYdwe19cVxoL/0tG7uSD1C5J4E90JRyHwh37OVYv3t
JNG9RkhQEJQpJpHyVM+7YbIn7/NxseVJAoIBAQDhRGxLuVO0VJA2Q3BQcn9Fp1jH
XykNNzvJaFPwAcf2/hc914XF2IRKCP3hIBiNNg5dZdvW5Zfj3lMGKbJY7OrNGapE
cvWnHUVmR+r3EYjf0Ae9YbZJDVW1oynJkVHuJ3Y7FBSvkSPcf2z4+GXwDjcgldHP
Z3rqpwLNLxA6tublmn04ATs6IcXE5MUWK3Lk1TjAnGTlOCmH8K6PfjaF7Tv2zx+2
4Vq1KprK/2BX1aifeBWHB4tFdAdGsqPrFyldwKeJjyFYelegPIqvq8vzXGL0iDRo
798A4zT6fA90Mszjxnrh0IXJh1DtvgNuq5rcqQpj0YaMu/9rGMHeQ8i6BB+G
-----END RSA PRIVATE KEY-----
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAwaVDa4NxzhAMxBFFiUJE
f/ZEqsZtC1zj8tDGrm/gMwW3Hf4rnG3t0SdBldVmEz1CHvsq/20V4SY11WcHqQtv
x3pdkEBHBXSCYxPeep7YFYUX4sYvYs3OrLvSBrsyrzEmVA7VC6d+vU4oLKGw+0le
I4IlDct3TZMjntQWRsQGkilR5iyt4werYyyHPiengaJzA3bsctBaYhyUEg4ZbIBg
pCeR56y/WM+cFgiP14EhWpshKhI4+9UCogIox8NPwpNUZDN1TV56ejjOdAfoMsMU
L6azKNXTDFi0Hnhr0I526zrNCWrtdkXNPQZdmGJHiFxUJUIAJTAu5awd1AQTtljJ
3P34761aoYyCbawNRDciUPEdDTMBcd774UwyvxmsoANzyNn/DxfgJ8Jn5JwdVLEZ
BXvAjocuDvjl7rZ9c9TUwB6E06vzl7dILaEbiFqKKcX92I2V4/RjiTp6Ovk90X7+
```

```
kGIsk5aW6vx6E3M2K3/UZyo7QW2kQRBA+UgeIQeffSZ2AhHSoyhJa6fBNagOl7IN
ThBeullVus0VaWmDEuWTo99YxIOeCEb6IwOe2PIC24mMKpSVua2AlfX46OPAaZ5f
tmmwUlY5FTNk8PFXUOf1drOzc0QfgA7Hq4t/AWEZD2ey0ZaCJV8IgEqKRdizNqvP
MPZq8mCUZ15YJglxKg65GS0CAwEAAQ==
-----END PUBLIC KEY-----
```

Using privatezsk.pem, let's now sign the zonedb for tcpwave.com & verify that publiczsk.pem is able to decrypt it(***Note that this is only for demonstration & in real-world, each resource records would be signed individually***).

```
dnsadmin@tcpwave$:cd /etc/bind/

dnsadmin@tcpwave$:sudo openssl dgst -sha256 -sign /opt/dnsadmin/privatezsk.pem
-out db.tcpwave.com.signature db.tcpwave.com

dnsadmin@tcpwave$:ls -l *.signature
-rw-r--r-- 1 root bind 512 Jan  1 03:42 db.tcpwave.com.signature
```

Now, we have db.tcpwave.com.signature to be the signature record for db.tcpwave.com. We can now use the public key to decrypt & verify the data's integrity by doing:

```
dnsadmin@tcpwave$:openssl dgst -sha256 -verify /opt/dnsadmin/publiczsk.pem -
signature db.tcpwave.com.signature db.tcpwave.com
Verified OK
```

Now let's tamper the data & modify the record to simulate a scenario where the data has been compromised:

```
dnsadmin@tcpwave$:sed "s/88.88.88.8/99.99.99.9/g" db.tcpwave.com
$TTL 86400

@ IN SOA supportstaffs tech.supportstaffs (
  2018050600
  3600
  900
  604800
  86400
)

@      IN NS server
server IN A  11.11.22.22
www IN A  99.99.99.9
```

Now when the integrity of the data is checked, it'll end up in validation failure. In a real-world scenario, when the data's signature fails to prove its integrity, that'll end up in SERVFAIL responses back to a client – which technically means that the record is spoofed:

```
dnsadmin@tcpwave$:openssl dgst -sha256 -verify /opt/dnsadmin/publiczsk.pem -
signature db.tcpwave.com.signature db.tcpwave.com
Verification Failure
```

To understand more specifics about the implementation using the logic above, please read through RFC-4034 which will help understand the actual representation of RRSIGs, encoding using base64, etc. As you

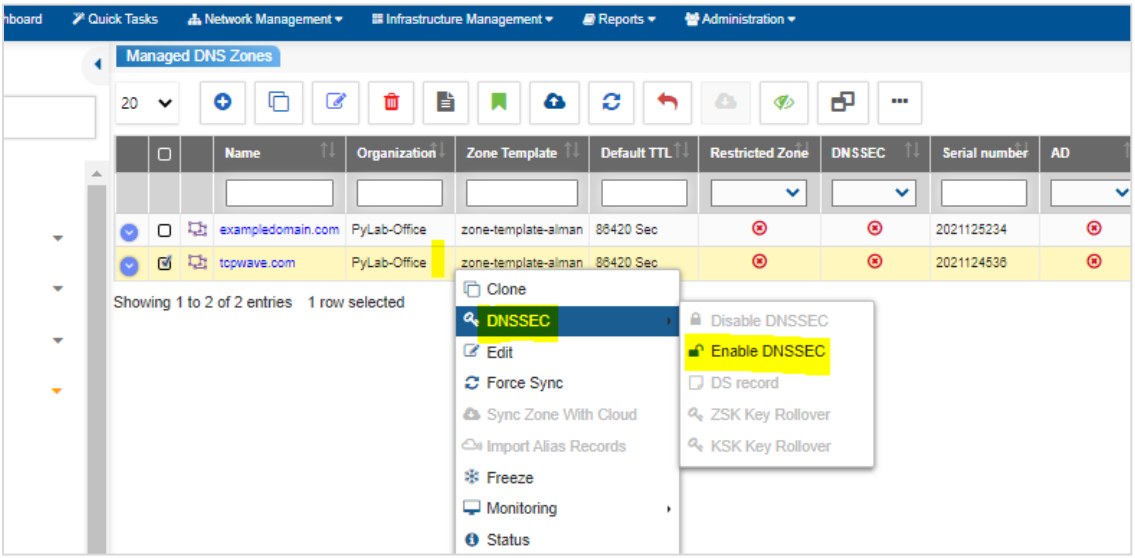
may have realized by now, to validate the data in a chain, the public key(KSK) must be passed on to the parent domain so that it'll be able to trust what's received from downstream(this is called the DS record).

The ZSK's public key is signed by the private KSK & the signature is encoded in base64 to create the RRSIG for the Zone Signing Key. The KSK from the DNSKEY set is used to verify the authenticity of the Zone Signing Key & that verified ZSK will eventually be used to verify the integrity of the zone's resource records.

Private ZSK Public ZSK	Basic DNSSEC Records formation in a nutshell		Private KSK Public KSK
www.tcpwave.com +	Private ZSK >	Signed & Encoded	www.tcpwave.com's RRSIG
mail.tcpwave.com +	Private ZSK >	Signed & Encoded	mail.tcpwave.com's RRSIG
Public ZSK >	Encoded >	DNSKEY(ZSK)	
Public ZSK +	Private KSK >	Signed & Encoded	ZSK's RRSIG
Public KSK >	Encoded >	DNSKEY(KSK)	
Public KSK >	Hashed >	DS Record	

TCPWave Makes It Easy!

All you need to do in TCPWave to enable DNSSEC validation for a zone is:



TCPWave IPAM takes care of the key rollover automatically. That single click will take care of all remaining things in the backend such as key pair generation, RRSIG creation for all zone RRs, etc. By default, KSKs are rolled over every year & the ZSKs are rolled over every three months. TCPWAve IPAMs have embedded support for the DNSViz utility to make administration much better. An administrator can immediately lookup the chain of trust for a domain managed within the TCPWave infrastructure.

DNSSEC Key pair automatically generated after clicking the “Enable DNSSEC” option above:

```
192.168.29.55>pwd
/opt/tcpwave/chroot/var/named/keys
192.168.29.55>ls -l
total 16
-rw-rw---- 1 twcadmin twcadmin  386 Dec 31 23:34 Ktcpwave.com.+008+16590.key
-rw-rw---- 1 twcadmin twcadmin 1847 Dec 31 23:34 Ktcpwave.com.+008+16590.private
-rw-rw---- 1 twcadmin twcadmin  386 Dec 31 23:34 Ktcpwave.com.+008+18170.key
-rw-rw---- 1 twcadmin twcadmin 1847 Dec 31 23:34 Ktcpwave.com.+008+18170.private
192.168.29.55>
```

DNSKEYs visible to the outside world for validation (Check out the Key ID & compare it with the snip above)

```
; <<>> DiG 9.11.35 <<>> @192.168.29.55 tcpwave.com dnskey +multi
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23951
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 25f405826b3c91349d22509261cf93eaa4370546028bf348 (good)
;; QUESTION SECTION:
;tcpwave.com.           IN DNSKEY

;; ANSWER SECTION:
tcpwave.com.            86400 IN DNSKEY 257 3 8 (
                               AwEAAYW8TQBy+fDfsJBFsrFh4EB7VQR3pnps8+uhWK76
                               WM9Dc891kleSPplDoRtmZXjcTTa8eGrPVprUgSGXkZLz
                               mwdqqUtfySN2beylGQdeMWwbZVpWSz2+UGNl8qOX8ih2
                               OozvJz5tJ07FjFOh6/chb2WBlfkqYGEV9s2fVZ34inGj
Pub-KSK                        mErEVBCeUpLhvm1EpSQ25P4vGeWqBBMo0/U2GV9IXjPX
                               LAZBYx/CpdQDbB7LxY1u7V47495P/uwjONCtaRkvyB4K
                               9BTTZOab9EwwBNDzRswtc7wuCS7SpUkb3d0qPbIVl1YR
                               5ibRywBM/2bj7TfPIYr6Ic/rfm2iEVQnCBu+bYc=
                               ) ; KSK; alg = RSASHA256 ; key id = 18170
tcpwave.com.            86400 IN DNSKEY 256 3 8 (
                               AwEAAZDEYNFfJUbOQobp8iTtOpo601jIIUGBixR9h6QF
                               yTBbfb0kzzU6E85/k5KgQOJe9Wirk7H3eMAZX3lGdT98
                               AcAcm2PMF1RcPLgSEaSWwArG2G5TOUCiSq0WaHTj1XOs
Pub-ZSK                        /XMJd9ZIhkCvB2LK2jPe8u1gLhGDHvUquk8Sl9A+lN3m
                               +/LUhL9aiIdtvJGVVoZvdzfi8Mk41Z8MAUJ3/v/9hKGU
                               dql2Wu8RdHvbbojmSeA2rhj+V/zeCD+Z4UagHwsq022C
                               4M/LrhYZ5sHh0iL2hn7YrvExHUSt3GDcER24wOmEApon
                               gTYeuTKm5OWE6mBlfEZsLg4Ce1isfKgcYQ2HuoM=
                               ) ; ZSK; alg = RSASHA256 ; key id = 16590

;; Query time: 0 msec
;; SERVER: 192.168.29.55#53(192.168.29.55)
;; WHEN: Fri Dec 31 23:36:10 GMT 2021
;; MSG SIZE  rcvd: 620
```

TCPWave has support for reports around DNSSEC & many other security features. For a quick demo, contact the **TCPWave Sales Team**.