WireGuard

A Fast, Modern and Secure VPN Tunnel

Developer: Jason A. Donenfeld

Documentation: https://www.wireguard.com/

Objective: Demonstration of the usage of wireguard with network namespaces

Aditya Chirania
181CO104

Contents of the Demonstration

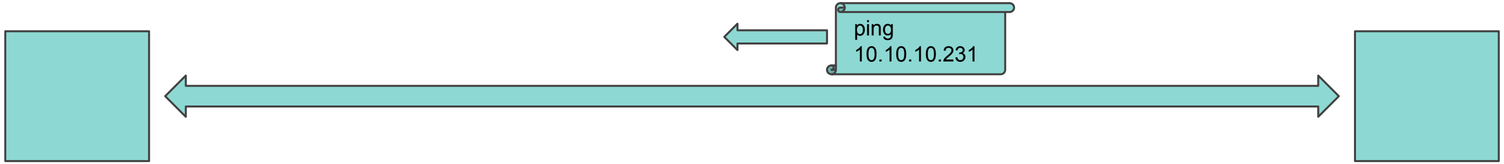
1. Basic concepts of Wireguard
2. Installation and preparation for demonstration
3. Creating a direct node-to-node secure VPN Tunnel.
4. Creating a relaying/bouncing VPN server.
5. Using Network Namespaces as an alternative to classical routing table hacks.
6. Creating a setup to access internet via VPN.

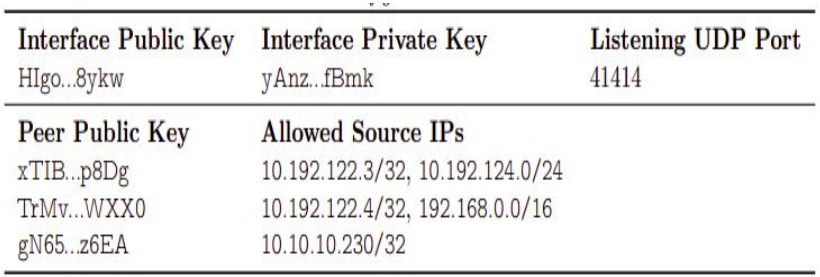
Note : All the above cases will be mere simulations with network namespaces to help viewers attempt the same parallely with ease. It is also important to mention that the point 5 is not a simulation but an alternative used in real world VPN setups.

Basic Concepts of Wireguard (Cryptokey Routing, endpoints, roaming)

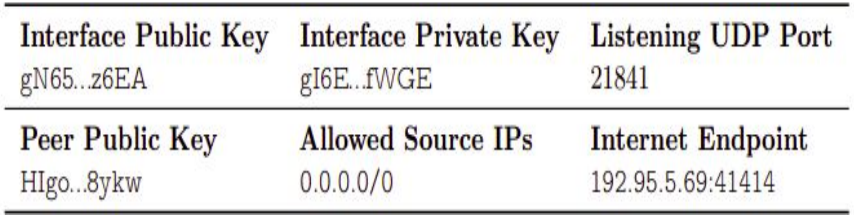
Node 1, VPN IP: 10.10.10.231/24, Public IP: 192.95.5.69

Node 2, VPN IP: 10.10.10.230/24, Public IP: 192.95.5.64





Interface Public Key	Interface Private Key	Listening UDP Port
HIgo...8ykw	yAnz...fBmk	41414
Peer Public Key	**Allowed Source IPs**	
xTIB...p8Dg	10.192.122.3/32, 10.192.124.0/24	
TrMv...WXX0	10.192.122.4/32, 192.168.0.0/16	
gN65...z6EA	10.10.10.230/32	



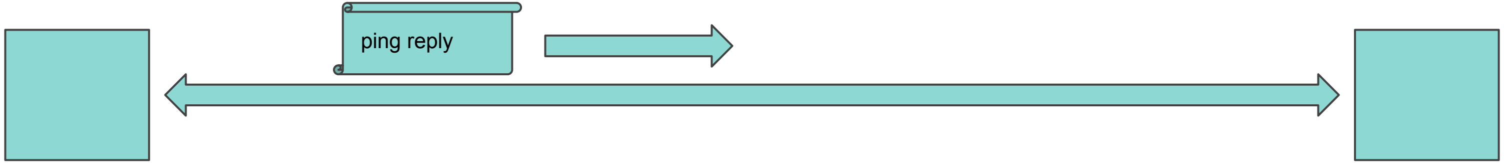
Interface Public Key	Interface Private Key	Listening UDP Port
gN65...z6EA	gI6E...fWGE	21841
Peer Public Key	**Allowed Source IPs**	**Internet Endpoint**
HIgo...8ykw	0.0.0.0/0	192.95.5.69:41414

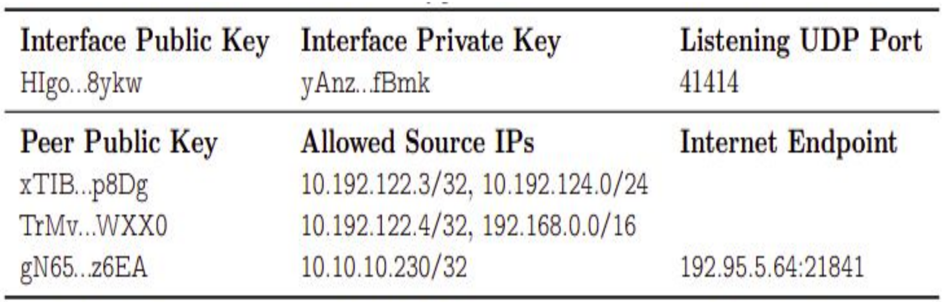
Cryptokey Table Image Credits: www.wireguard.com/papers/wireguard.pdf

Basic Concepts of Wireguard (Cryptokey Routing, Endpoints, & Roaming)

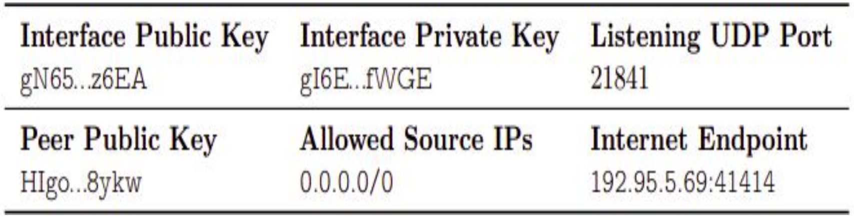
Node 1, VPN IP: 10.10.10.231/24, Public IP: 192.95.5.69

Node 2, VPN IP: 10.10.10.230/24, Public IP: 192.95.5.64





Interface Public Key	Interface Private Key	Listening UDP Port
HIgo...8ykw	yAnz...fBmk	41414
Peer Public Key	**Allowed Source IPs**	**Internet Endpoint**
xTIB...p8Dg	10.192.122.3/32, 10.192.124.0/24	
TrMv...WXX0	10.192.122.4/32, 192.168.0.0/16	
gN65...z6EA	10.10.10.230/32	192.95.5.64:21841



Interface Public Key	Interface Private Key	Listening UDP Port
gN65...z6EA	gI6E...fWGE	21841
Peer Public Key	**Allowed Source IPs**	**Internet Endpoint**
HIgo...8ykw	0.0.0.0/0	192.95.5.69:41414

Cryptokey Table Image Credits: www.wireguard.com/papers/wireguard.pdf

Installation and preparation for demonstration

Ubuntu Installation

```
$ sudo apt install wireguard
```

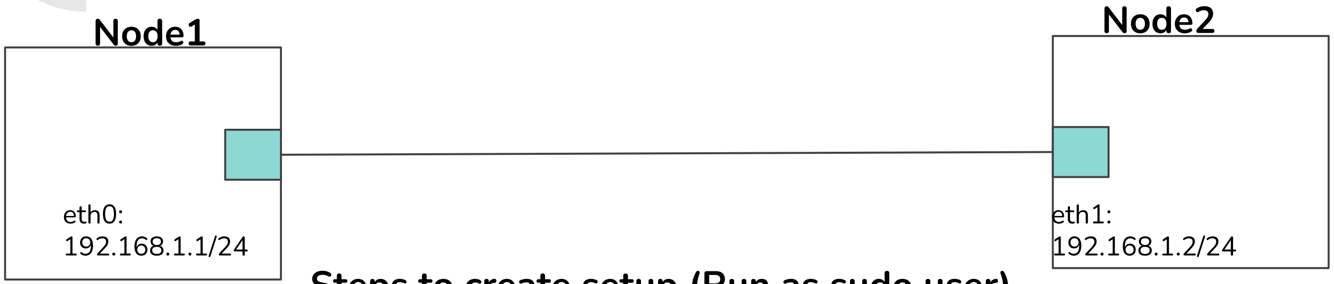
For any other linux distributions please refer: www.wireguard.com/install/

To follow parallely along with this demo, please clone this repository by:

```
$ git clone https://github.com/adityachirania/Wireguard-Demonstration.git
```

Creating a direct node to node VPN tunnel with wireguard

Initial Setup

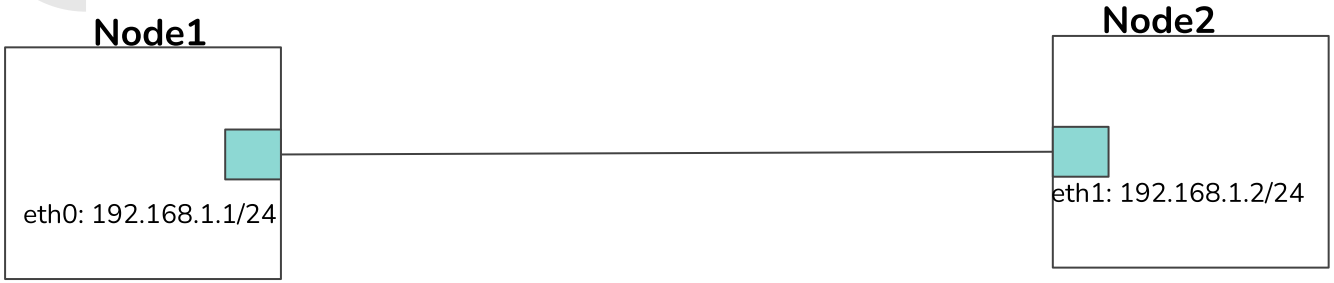


Steps to create setup (Run as sudo user)

```
$ cd wireguard-demo/Experiment1
$ bash setup1.sh
```

Creating a direct node to node VPN tunnel with wireguard

Add wireguard interfaces

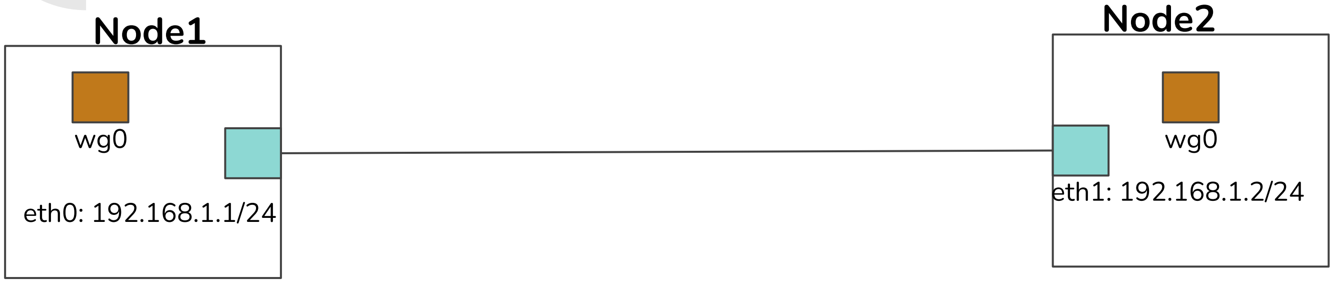


```
$ ip netns exec Node1 bash
$ wg genkey > private_node1
```

```
$ ip netns exec Node2 bash
$ wg genkey > private_node2
```

Creating a direct node to node VPN tunnel with wireguard

Add wireguard interfaces

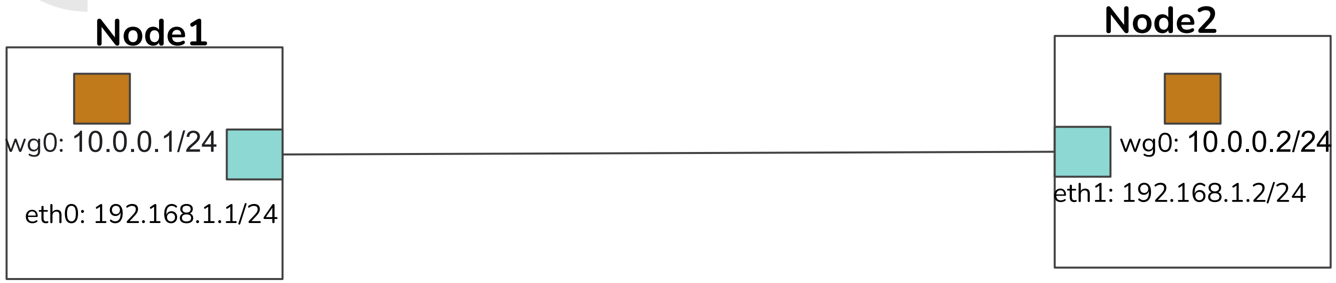


```
$ ip netns exec Node1 bash
$ wg genkey > private_node1
$ ip link add wg0 type wireguard
```

```
$ ip netns exec Node2 bash
$ wg genkey > private_node2
$ ip link add wg0 type wireguard
```

Creating a direct node to node VPN tunnel with wireguard

Add wireguard interfaces

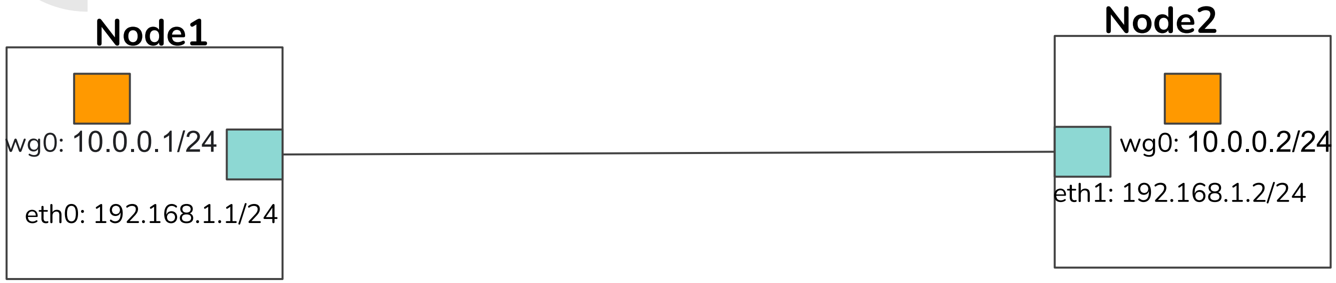


```
$ ip netns exec Node1 bash
$ wg genkey > private_node1
$ ip link add wg0 type wireguard
$ ip address add 10.0.0.1/24 dev wg0
$ wg set wg0 private-key
./private_node1
```

```
$ ip netns exec Node2 bash
$ wg genkey > private_node2
$ ip link add wg0 type wireguard
$ ip address add 10.0.0.2/24 dev wg0
$ wg set wg0 private-key
./private_node2
```

Creating a direct node to node VPN tunnel with wireguard

Add wireguard interfaces



Node1

wg0: 10.0.0.1/24

eth0: 192.168.1.1/24

Node2

wg0: 10.0.0.2/24

eth1: 192.168.1.2/24

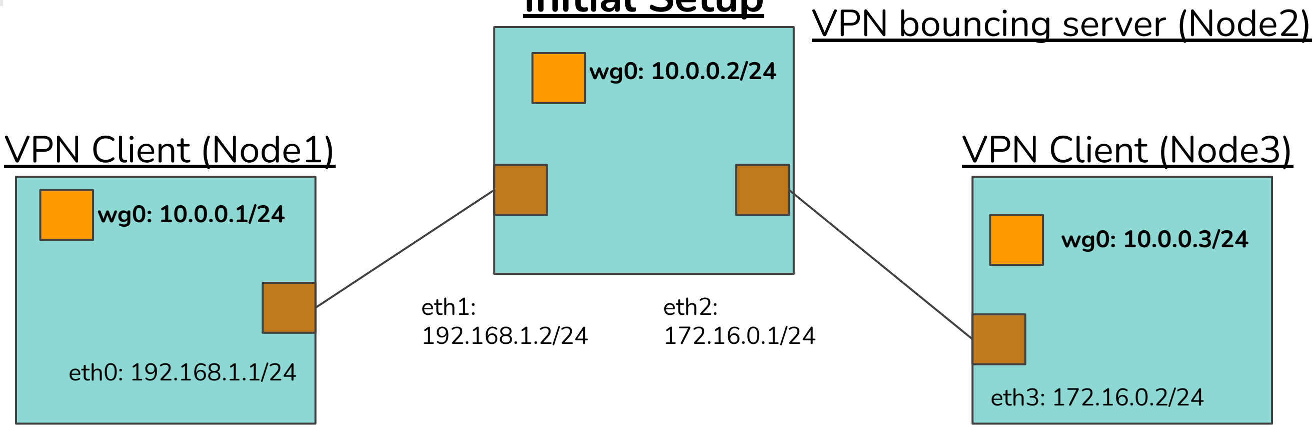
```
$ ip link set wg0 up
$ wg
[Interface]
public: xyZ...pq=
private: (hidden)
listening port: 41231
$ wg set wg0 peer yPq...ab=
allowed-ips 10.0.0.2/32 endpoint
192.168.1.2:21356
$ ping 10.0.0.2
```

Endpoint is not mandatory to specify on both ends.
Note the output of "wg" before and after you ping in both namespaces

```
$ ip link set wg0 up
$ wg
[Interface]
public: yPq...ab=
private: (hidden)
port: 21356
$ wg set wg0 peer xyZ...pq=
allowed-ips 10.0.0.1/32
```

Creating a relaying/bouncing VPN server.

Initial Setup



Steps to create setup (Run as sudo user)

```
$ cd wireguard-demo/Experiment2
$ bash setup2.sh
```

This setup has the wireguard interfaces already up and assigned their addresses because we covered how to do so in the previous demo. We shall be focusing on viewing the hops of the packet in this demo.

Creating a relaying/bouncing VPN server

Peers of clients

Node 1’s peers

```
[peers]
1:public-key: <public-key-Node2>
   allowed-ips: 10.0.0.0/24
   endpoint: 192.168.1.2 : <port-node2>
```

Node 3

```
[peers]
1:public-key: <public-key-Node2>
   allowed-ips: 10.0.0.0/24
   endpoint: 172.16.0.1 : <port-node2>
```

Configuration of the VPN Server (Node 2)

```
[peers]
1:public-key: <public-key-Node1>
   allowed-ips: 10.0.0.1/32
   endpoint: 192.168.1.1 : <port-node1>
2:public-key: <public-key-Node3>
   allowed-ips: 10.0.0.3/32
   endpoint: 172.16.0.2 : <port-node3>
```

Creating a relaying/bouncing VPN server

- Finally just run “ping 10.0.0.3” in Node1 and notice that the ping works.
- Also Notice the hops across the relay server by running “traceroute -4 10.0.0.3” on Node1 to see the intermediate hop on the relay server.
- A relay server can be used to provide continuous access to the VPN network without failure.

Creating a relaying/bouncing VPN server

What happened in the entire process ? Ping 10.0.0.3 from Node 1

UDP Packet created at wg0

10.0.0.1	10.0.0.3	PING MESSAGE

Creating a relaying/bouncing VPN server

What happened in the entire process ? Ping 10.0.0.3 from Node 1

UDP Packet encrypted

XXXXXXXXXXXXXX

Creating a relaying/bouncing VPN server

What happened in the entire process ? Ping 10.0.0.3 from Node 1

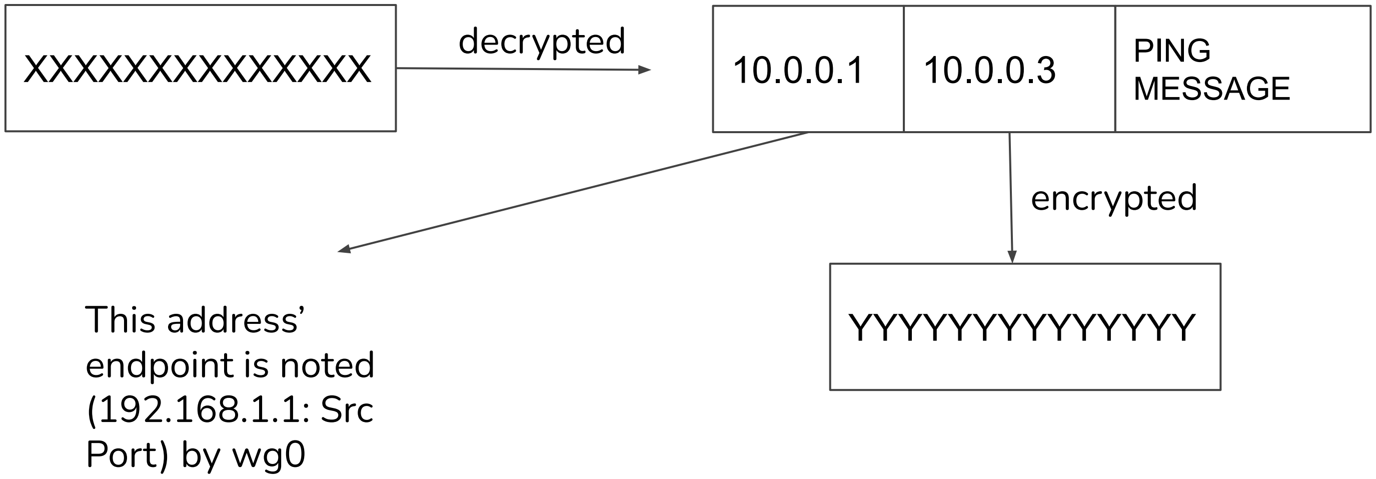
Encrypted packet to be sent at endpoint 192.168.1.2: Dst Port from eth0 interface

192.168.1.1	Src Port	192.168.1.2	Dst Port.	XXXXXXXXXXXXXXX

Creating a relaying/bouncing VPN server

What happened in the entire process ? Ping 10.0.0.3 from Node 1

Packet reaches eth1 at Node2 and the data payload is passed on to the wireguard interface listening on port : Port-No.



XXXXXXXXXXXXXX

decrypted

10.0.0.1

10.0.0.3

PING MESSAGE

encrypted

YYYYYYYYYYYYYY

This address' endpoint is noted (192.168.1.1: Src Port) by wg0

Creating a relaying/bouncing VPN server

What happened in the entire process ? Ping 10.0.0.3 from Node 1

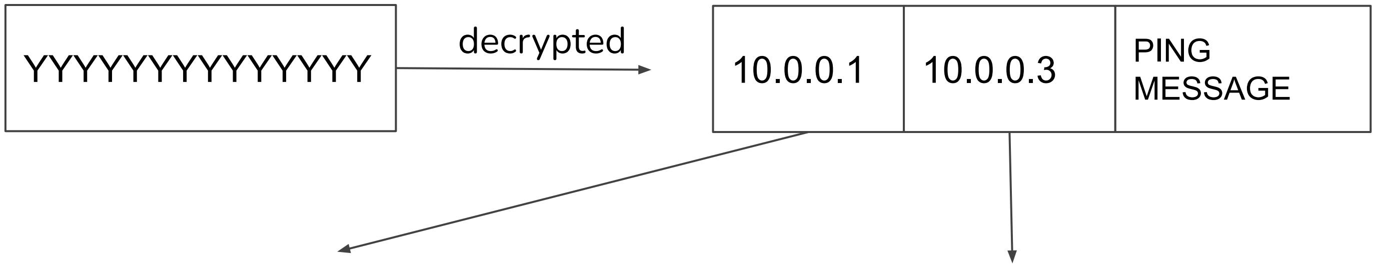
Encrypted packet to be sent at endpoint
172.16.0.2: Dst Port from eth2 interface

172.16.0.1	Src Port	172.16.0.2	Dst Port	YYYYYYYYYYYYY

Creating a relaying/bouncing VPN server

What happened in the entire process ? Ping 10.0.0.3 from Node 1

Packet reaches eth3 at Node3 and the data payload is passed on to the wireguard interface listening on port : Dst Port.


YYYYYYYYYYYYYY

decrypted

10.0.0.1

10.0.0.3

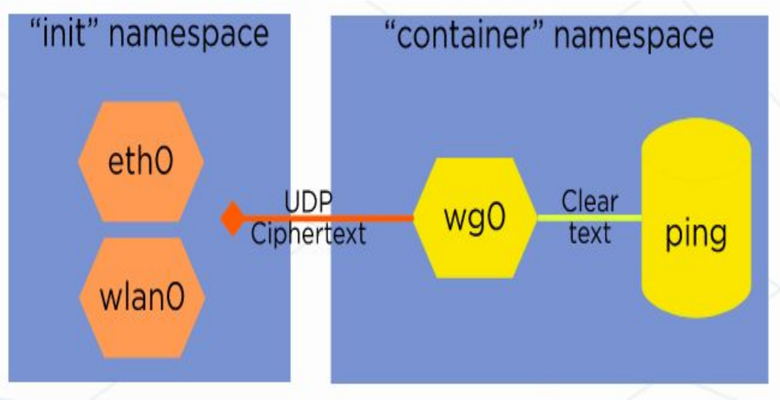
PING MESSAGE

This address' endpoint is noted (172.16.0.1: Src Port) by wg0

Now a ping reply is sent back in a similar fashion

Using network namespaces as an alternative to classical routing table hacks

Useful to give containers a single sole interface



Useful for routing all internet traffic via wireguard

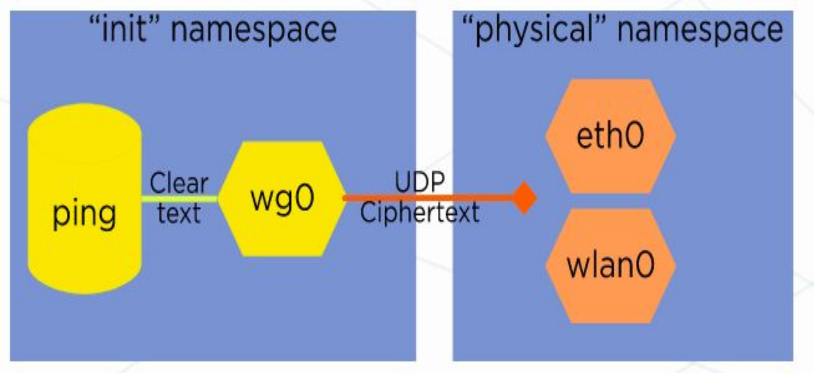
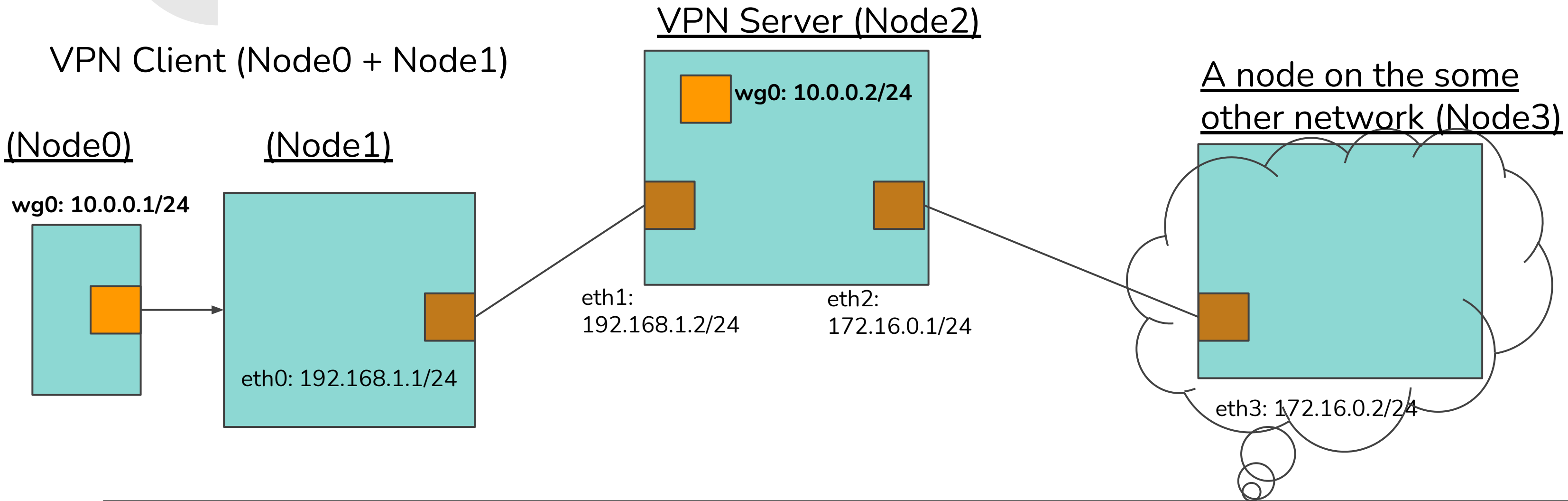


Image credits: www.wireguard.com

Creating a setup to access internet via VPN.



VPN Client (Node0 + Node1)

(Node0)

wg0: 10.0.0.1/24

(Node1)

eth0: 192.168.1.1/24

VPN Server (Node2)

wg0: 10.0.0.2/24

eth1: 192.168.1.2/24

eth2: 172.16.0.1/24

A node on the some other network (Node3)

eth3: 172.16.0.2/24

```
$ cd wireguard-demo/Experiment3
$ bash setup3.sh
```

Creating a setup to access internet via VPN.

Enabling the NAT with masquerading at Node2

```
$ iptables -t nat -A POSTROUTING -s 10.0.0.0/24 -o eth2 -j MASQUERADE
```

It is necessary to replace the source IP as the gateway to the other network otherwise, it would not be possible to route back to the VPN server. Also a VPN server always replace the sender’s IP with its own IP address to make the VPN client untraceable.

Creating a setup to access internet via VPN.

Now we will run tcpdump on the ethernet interface of the Node3 and we will ping from node0 to IP of Node3 and appreciate the working of the VPN set up.

```
Node0:
$ ping 172.16.0.2
```

```
Node3:
$ tcpdump
```

Creating a setup to access internet via VPN.

What happened in the entire process ? Ping 172.16.0.2 from Node 0

UDP Packet created at wg0

10.0.0.1	172.16.0.2	PING MESSAGE

Creating a setup to access internet via VPN.

What happened in the entire process ? Ping 172.16.0.2 from Node 0

UDP Packet encrypted

XXXXXXXXXXXXXXX

Creating a setup to access internet via VPN.

What happened in the entire process ? Ping 172.16.0.2 from Node 0

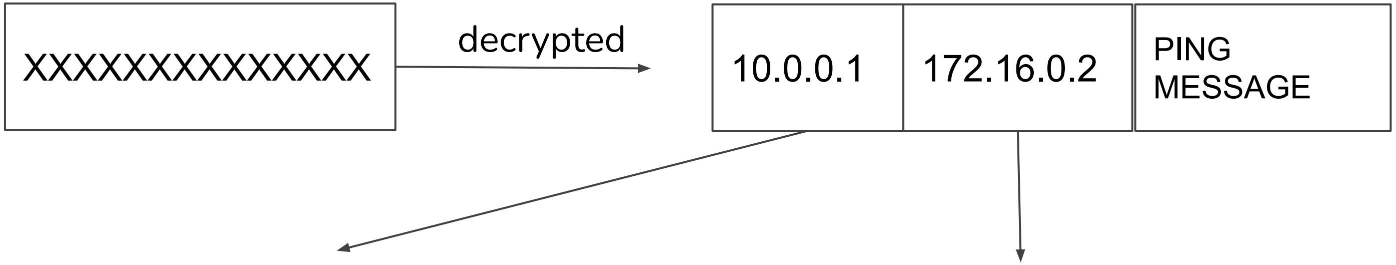
Encrypted packet to be sent at endpoint 192.168.1.2: Dst Port from eth0 interface in Node1

192.168.1.1	Src Port	192.168.1.2	Dst Port.	XXXXXXXXXXXXXX

Creating a setup to access internet via VPN.

What happened in the entire process ? Ping 172.16.0.2 from Node 0

Packet reaches eth1 at Node2 and the data payload is passed on to the wireguard interface listening on port : Port-No.



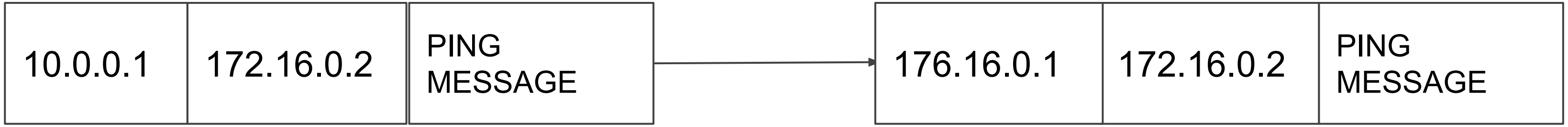
This address' endpoint is noted (192.168.1.1: Src Port) by wg0

Now since 172.16.0.2 belongs to the subnet 172.16.0.0/24 it will be sent via interface eth2

Creating a setup to access internet via VPN.

What happened in the entire process ? Ping 172.16.0.2 from Node 0

Now since 172.16.0.2 belongs to the subnet 172.16.0.0/24 it will be sent via interface eth2 to 172.16.0.2 but the NAT will first replace the source IP



10.0.0.1 | 172.16.0.2 | PING MESSAGE

176.16.0.1 | 172.16.0.2 | PING MESSAGE

It then reaches 172.16.0.2 at Node3 and the response is sent back in a similar fashion

Bibliography

- https://www.wireguard.com/
- https://dev.to/tangramvision/what-they-don-t-tell-you-about-setting-up-a-wireguard-vpn-1h2g
- https://github.com/pirate/wireguard-docs