istic Informatique Électronique

Architecture Réseaux Entreprises (ARES)

VPN et sécurité des communications

Brice - Ekane (brice.ekane@univ-rennes.fr)

ISTIC Rennes - France

2025-2026

git clone https://github.com/bekane/ares-2025.git

Plan du module

Objectifs pédagogiques

- Comprendre les enjeux de sécurité et performance des VPN
- Maîtriser le déploiement de WireGuard
- Justifier un choix de protocole en fonction des besoins

Problématique

Réseaux publics

Internet est un environnement hostile : interception, modification, usurpation.

Solution

Les VPN assurent confidentialité, intégrité, authentification.

Plan du module

Apperçu de la section 1

Définition

VPN = Réseau Privé Virtuel Permet de transporter du trafic privé sur un réseau public via un tunnel chiffré.

Cas d'usage

- Accès distant sécurisé
- Interconnexion de sites
- Contournement de restrictions géographiques

Menaces sur Internet

- Écoute (eavesdropping)
- Altération de données
- Injection / usurpation

Objectifs d'un VPN

- Confidentialité
- Intégrité
- Authentification

Encapsulation

Apperçu de la section 2

IPsec

- Standard industriel
- Avantages : intégré au noyau, interopérabilité
- Inconvénients : configuration complexe

OpenVPN

- Basé sur TLS
- Avantages : flexible, traverse NAT (Ex. tunnel sur 443)
- Inconvénients : performances inférieures

PPTP et L2TP

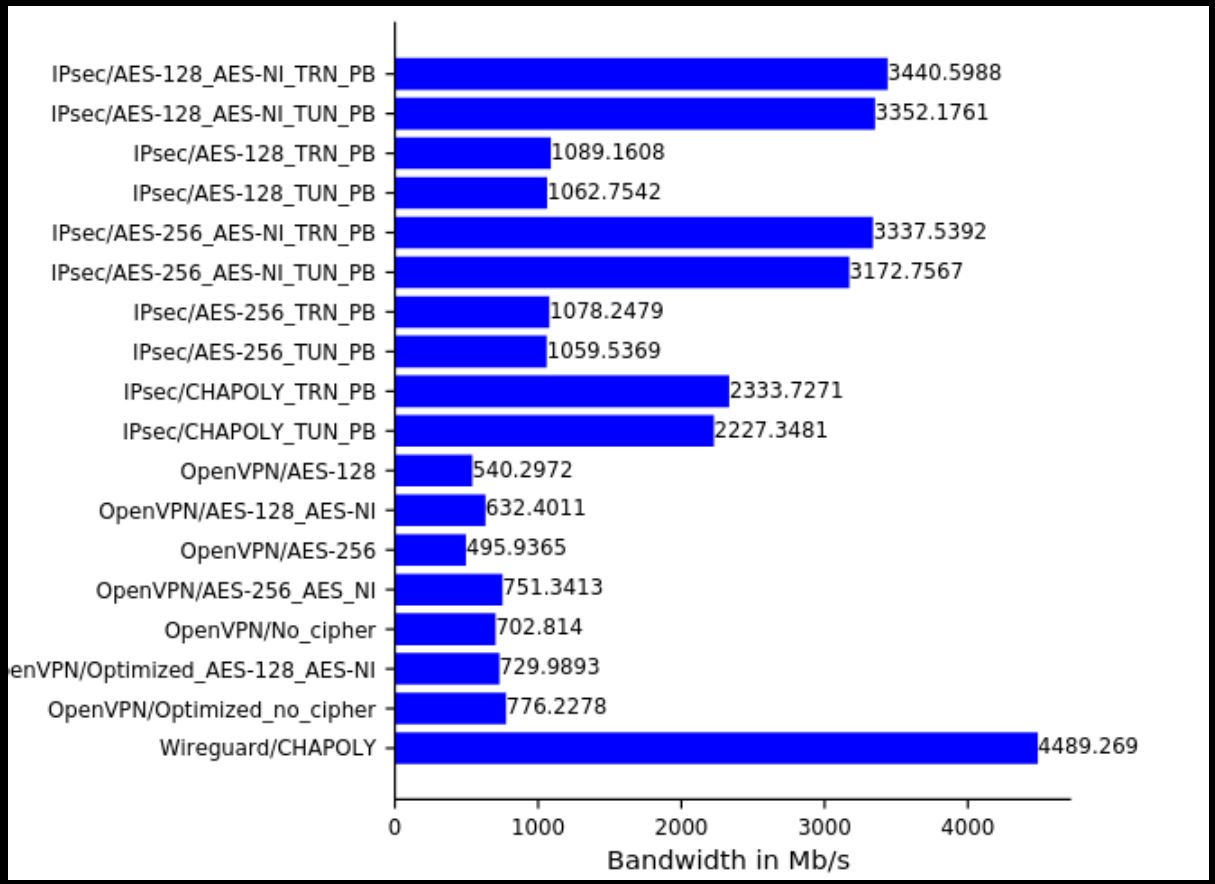
- Anciennes solutions
- 🔒 Faible (MPPE 128 bits)
- Failles connues -> obsolètes
  (https://nvd.nist.gov/vuln/search#/nvd/home?keyword=PPTP&resultType=records)

Critères de choix

- Sécurité
- Performances
- Simplicité de déploiement

Limites des VPN classiques

- Lenteur



Source : Osswald, Haeberle & Menth (2020).

- Maintenance complexe

Maintenance des VPN

IPsec vs OpenVPN vs WireGuard

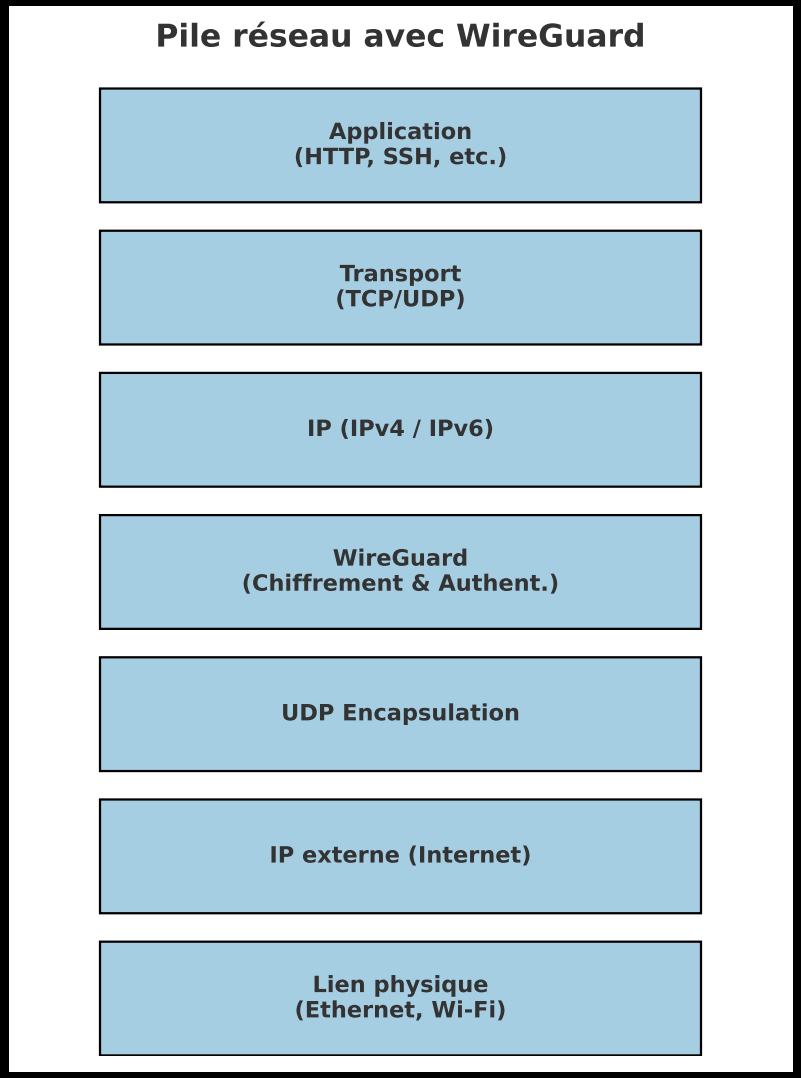
Aspect	IPsec	OpenVPN	WireGuard
Configuration	Très complexe (IKEv1/v2, AH/ESP, transport/tunnel, suites variées)	Plus simple mais riche (UDP/TCP, TUN/-TAP, TLS)	**Très simple** : pairs/-clefs, AllowedIPs, pas de nego ciphers
Clés/Certs	PKI lourde (génération, révocation, rotation)	Cycle TLS (certs, CRL/OCSP)	Clés statiques, **pas de PKI native** (rotation à orchestrer)
Interopérabilité	Différences constructeurs	Homogène, multi-plateforme	Bon clients, **faible support legacy**
Débogage	Logs cryptiques	Logs verbeux	État minimaliste (`wg show`, handshakes, compteurs)
Évolutivité	Politiques par sous-réseaux ingérables	Bon multi-utilisateurs (LB)	**Orchestration nécessaire** à grande échelle
Évolution protocole	Suites évolutives ⇒ reconfigurations	Dépend TLS/OpenSSL	Primitives modernes **fixées**
Performance	Excellente (noyau/offload)	Plus faible (user-space)	**Très élevée** (noyau Linux)
Fonctions réseau	L3, EAP/RADIUS	TUN/TAP, UDP/TCP	**UDP seul**, TUN (L3), NAT-T intégré

Apperçu de la section 3

Origine et philosophie

- Créé par Jason A. Donenfeld
- Code minimaliste

Positionnement dans la pile réseau



Objectifs de design

- Simplicité
- Sécurité moderne
- Performance

Comparatif des VPN : IPsec, OpenVPN, WireGuard

Synthèse des critères

Critère	IPsec	OpenVPN	WireGuard
Code source	Complexe, volumineux	Volumineux	**Minimal (<4k lignes)**
Performance	Bonne (kernel)	Moyenne (user space)	**Excellente (kernel)**
Simplicité config	Faible	Moyenne	**Élevée**
Interopérabilité	Large	Large	**En croissance**
Cryptographie	Variable	TLS (OpenSSL)	**Moderne (Noise, ChaCha20)**

Adoption

- Intégré Linux 5.6+
- Ports BSD, Windows, macOS

Apperçu de la section 4

Protocole Noise IK

- Échange de clés Curve25519

Pour aller loins :`https://noiseprotocol.org/noise.pdf`

Noise IK – Handshake étape par étape

Initiator (I)	**Responder (R)**
Clés pré-partagées (WireGuard)	
s_i (statique), connaît s_r	s_r (statique), connaît s_i
Échange	
$\rightarrow$ envoie e_i (éphémère) envoie s_i (encrypté)	reçoit e_i déchiffre s_i
reçoit e_r	$\leftarrow$ envoie e_r (éphémère)
Calculs Diffie–Hellman	
I : $DH(e_r, s_i)$, $DH(e_r, e_i)$	R : $DH(e_i, s_r)$, $DH(s_i, s_r)$
Clés de session dérivées (KDF)	
HKDF-SHA256($DH\_1 \parallel DH\_2 \parallel \dots$)	
$\rightarrow K_{I\rightarrow R}$: clé sym. pour I $\rightarrow$ R (ChaCha20-Poly1305)	
$\rightarrow K_{R\rightarrow I}$: clé sym. pour R $\rightarrow$ I (ChaCha20-Poly1305)	

Propriétés : authentification mutuelle, forward secrecy, 1.5 RTT

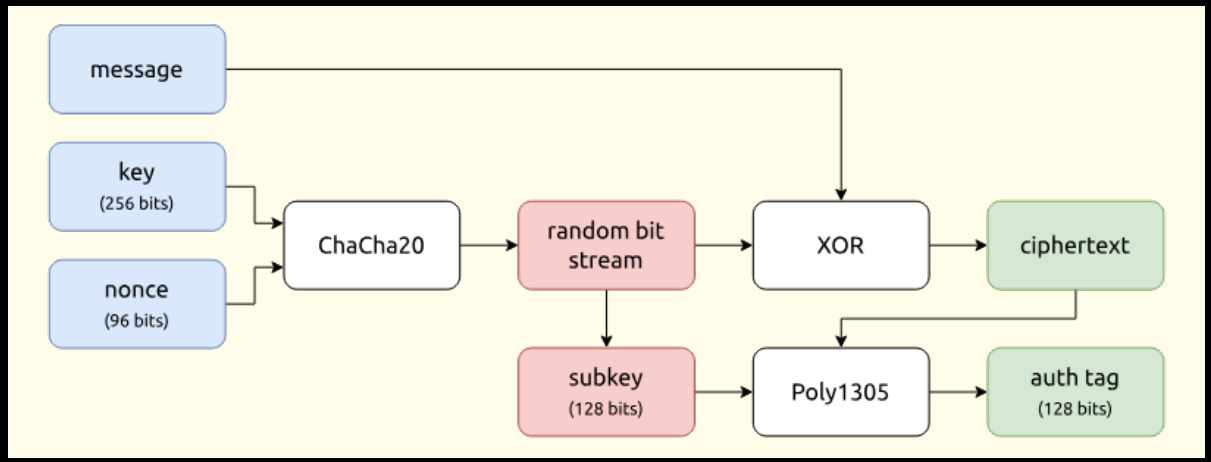
Interface wg0

- Capture des paquets vers AllowedIPs
- Chiffrement + encapsulation UDP

Structure d'un paquet WireGuard encapsulé

ChaCha20-Poly1305

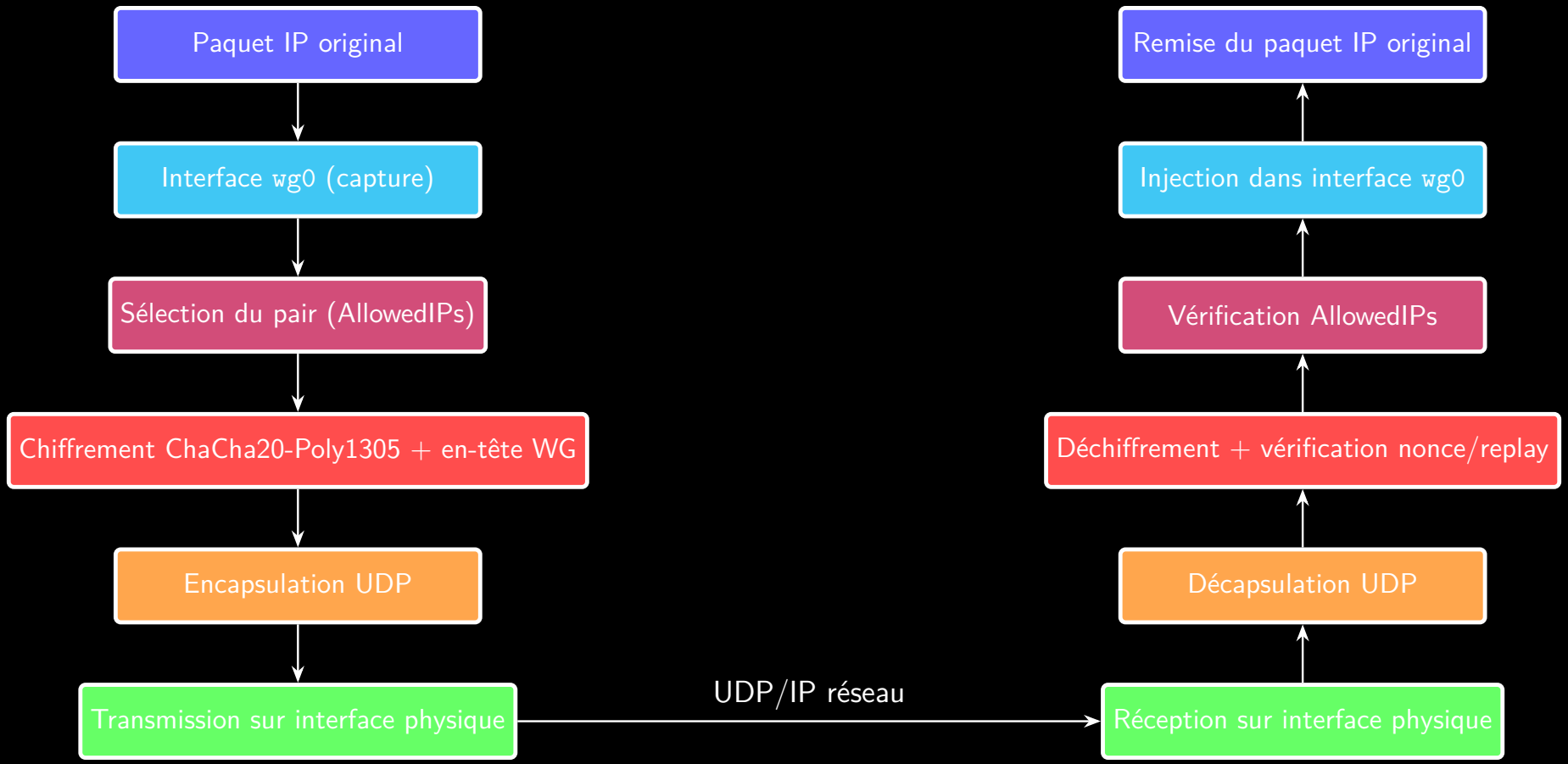
- Chiffrement rapide et sûr
- Authentification intégrée



Pour aller-loins:

- https://www.rfc-editor.org/rfc/rfc8439
- https://andrea.corbellini.name/2023/03/09/authenticated-encryption/

Flux d'un paquet à travers WireGuard



Apperçu de la section 5

Exemple serveur

Installer wireguard :

```
sudo apt install wireguard-tools
```

```
[Interface]
PrivateKey = 4CRi/+PmdSnFrkS1p1GY+mJikMG8wYkobZfcCmnfj1k=
ListenPort = 51822
Address = 10.0.0.2/32
[Peer]
PublicKey = BB+W6g8D4P1sVC55i+rKRoGAFgddxqIQfP+LkOH8iFI=
AllowedIPs = 10.0.0.1/32
Endpoint = 192.168.121.53:51821
```

Exemple client

```
[Interface]
PrivateKey = oEBCwSUJ9OVc2XxqlqJQXxrRq9Ux4hacZTlvgog1sWE=
Address = 10.0.0.1/32
ListenPort = 51821

[Peer]
PublicKey = Ox8RS3rCq0ugpaeqLHVwsHxHXygTqATT3QxVFc+DgVk=
Endpoint = 192.168.121.202:51822
AllowedIPs = 10.0.0.2/32
```

AllowedIPs

AllowedIPs = routes + ACLs

- routage : "quels paquets envoyer dans le tunnel vers ce peer"
- filtrage : "quelles adresses IP on accepte de ce peer"

Filtrage et routage conditionnel des flux chiffrés

Routage des flux chiffrés

- **AllowedIPs** : définit les préfixes autorisés pour un pair et agit comme table de routage privée.
- **Full tunnel** : `0.0.0.0/0, ::/0` $\Rightarrow$ tout le trafic passe dans le VPN.
- **Split tunnel** : seuls certains sous-réseaux passent dans le tunnel.

Exemple WireGuard

```
[Peer]
PublicKey = abc...
AllowedIPs = 10.0.0.0/24, 192.168.1.10/32
```

$\rightarrow$ Trafic pour 10.0.0.0/24 et 192.168.1.10 via le tunnel.

Filtrage et routage conditionnel des flux chiffrés

Filtrage côté VPN

- Utiliser `iptables`/`nftables` pour limiter l'accès via `wg0`.
- Empêcher les injections de routes non autorisées par les pairs.

Exemple iptables

```
iptables -A INPUT -i wg0 -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -i wg0 -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -i wg0 -j DROP
```

→ Seuls SSH et HTTP sont autorisés via le VPN.

Filtrage et routage conditionnel des flux chiffrés

Routage conditionnel avancé

- **Policy routing** : appliquer des règles selon source, port, protocole.
- Associer des paquets marqués à une table de routage spécifique.

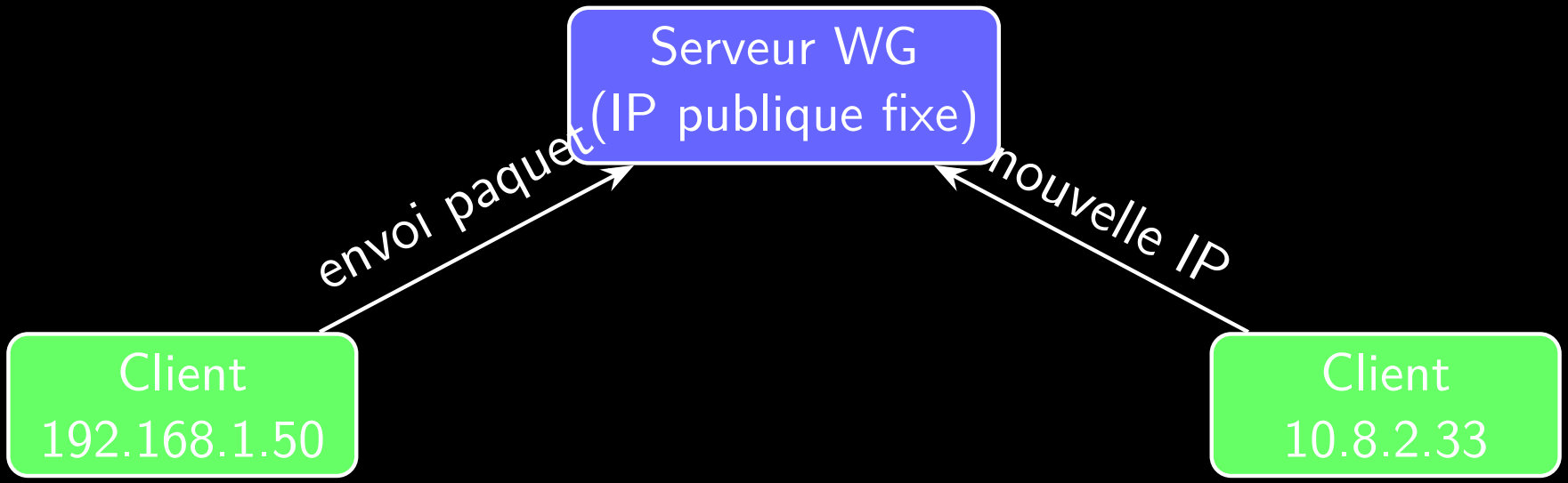
Exemple HTTPS uniquement via VPN

```
iptables -t mangle -A PREROUTING -p tcp --dport 443 \
         -j MARK --set-mark 1

ip rule add fwmark 1 lookup 100
ip route add default dev wg0 table 100
```

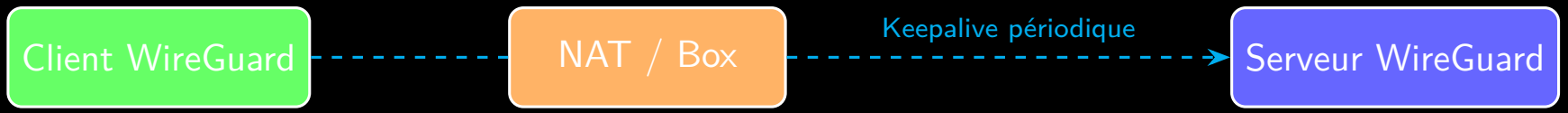
→ Seul le trafic HTTPS passe par le tunnel.

Endpoints dynamiques



- Le client peut changer d'IP (Wi-Fi $\rightarrow$ 4G).
- Dès qu'il envoie un paquet valide, **le serveur met à jour son endpoint**.
- Pas besoin de reconfigurer : l'identité repose sur la clé publique, pas sur l'IP.

Keepalive



- Envoie de petits paquets vides
- Empêche le NAT d'effacer la session
- Très léger, seulement si nécessaire

Commandes

- `wg`
- `wg-quick`

Apperçu de la section 6

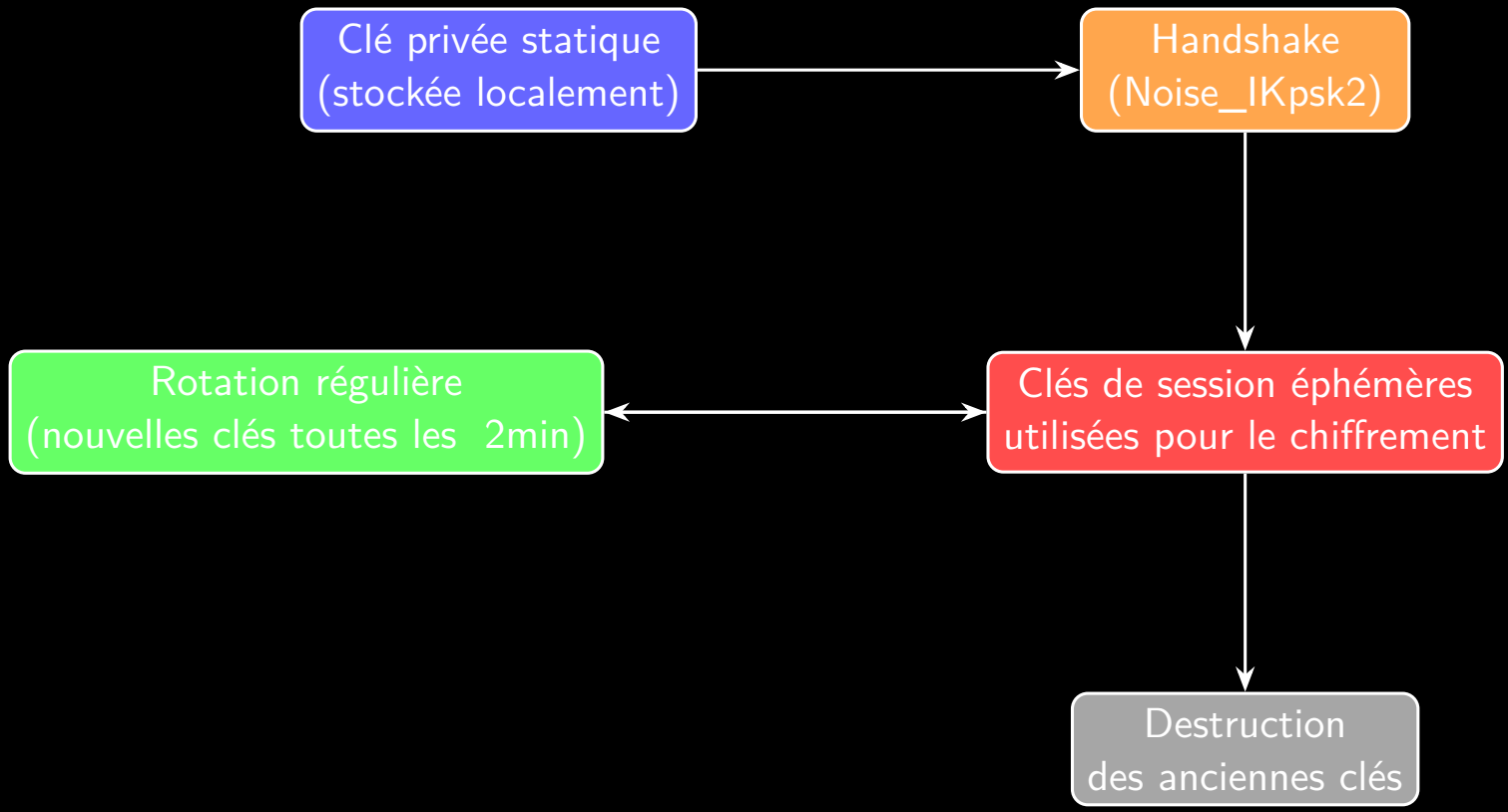
Surface d'attaque

4k lignes de code -> audit facile

Pas de réponse aux paquets invalides

- Difficile à détecter pour un attaquant

Gestion des clés



- **Rotation régulière** : nouvelles clés dérivées automatiquement.
- **Stockage sécurisé** : clé privée jamais transmise, clés de session effacées.

Pare-feu et WireGuard

- WireGuard crée une interface réseau normale (‘wg0‘).
- **Nftables** filtre le trafic entrant/sortant sur ‘wg0‘.
- Intégration simple : mêmes règles que pour pour une interface physique.

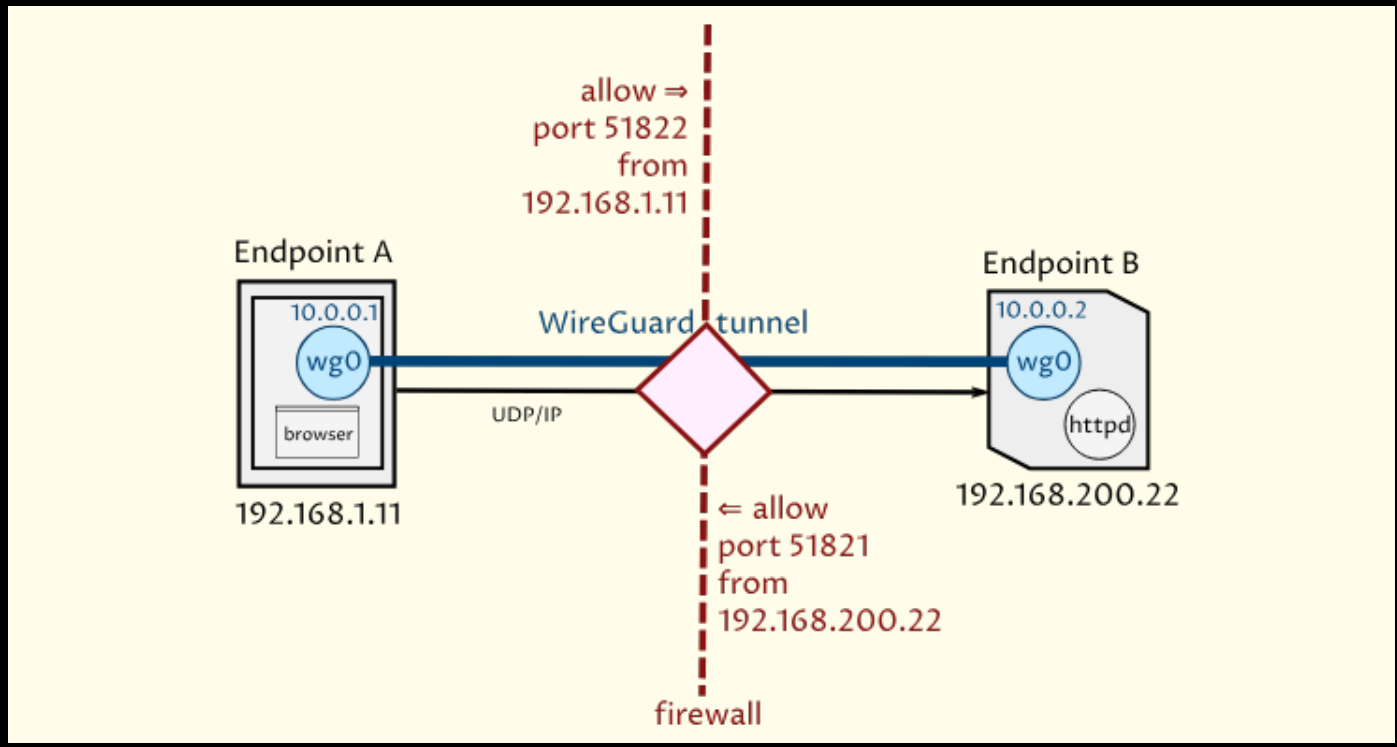
Audit et logs

Outil	Usage
`wg show`	État du tunnel, derniers handshakes, volume échangé
`tcpdump / tshark`	Capture et analyse du trafic UDP WireGuard
`nftables`	Compter, logguer et filtrer les paquets sur `wg0`
`journalctl / dmesg`	Logs système et éventuelles erreurs du module

- **Monitoring trafic** : suivre activité et volume sur `wg0`.
- **Détection anomalies** : repérer scans, absence de handshake, trafic inhabituel.

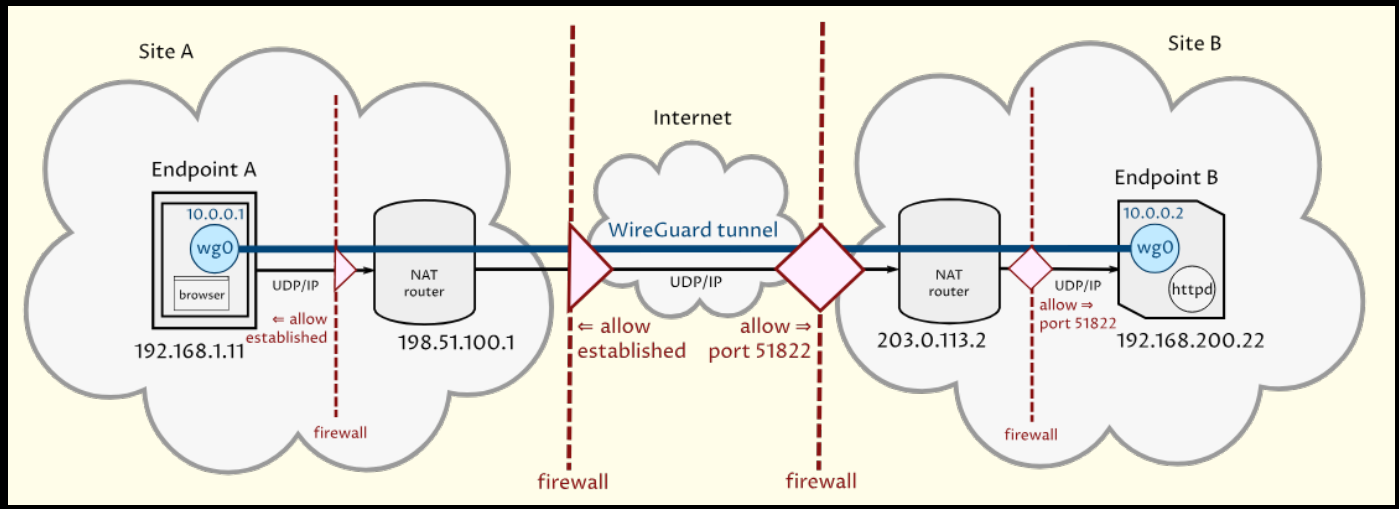
Apperçu de la section 7

Point to Point



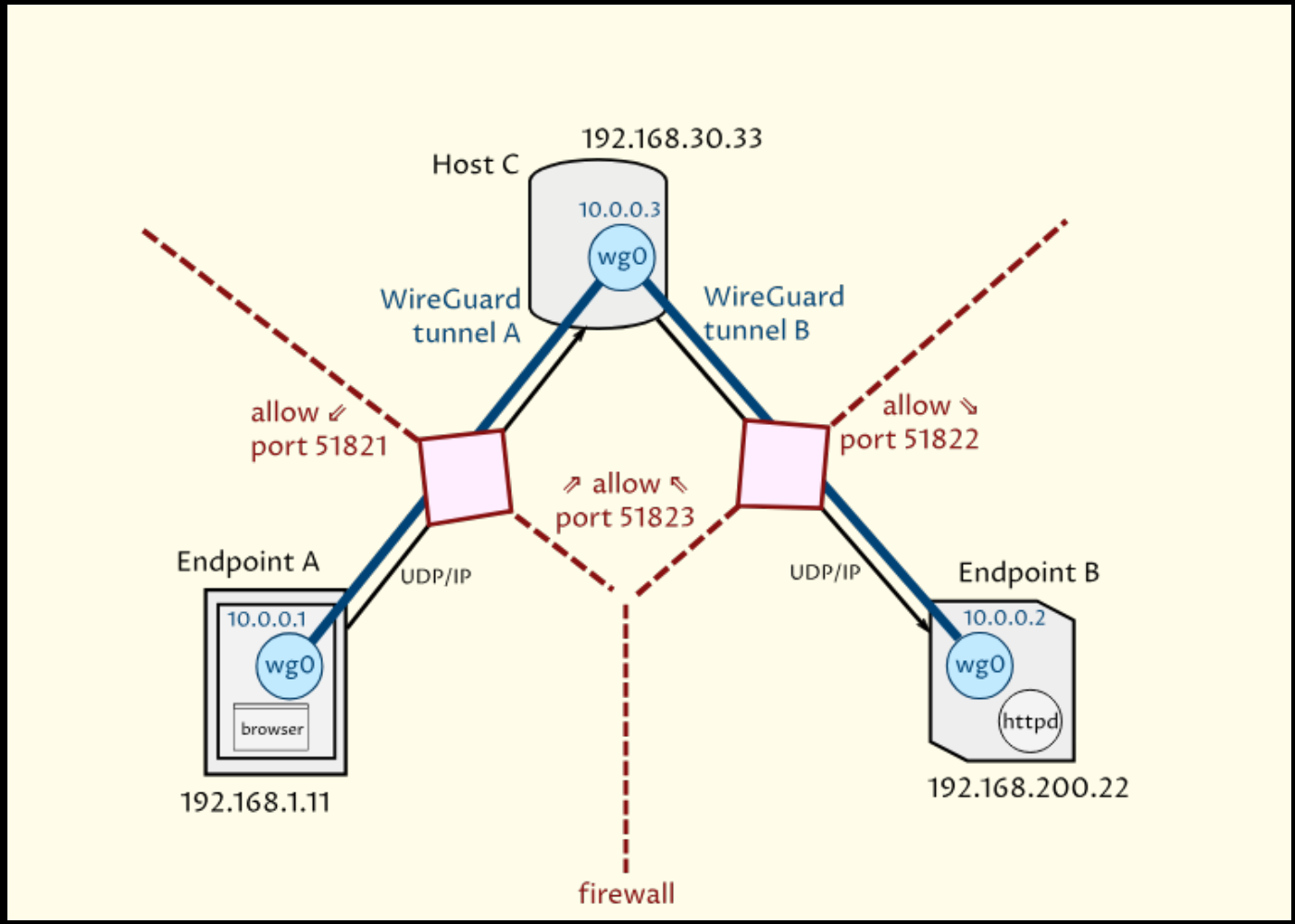
source : https://www.procustodibus.com/blog/2020/10/wireguard-topologies/

Point to Point avec firewall



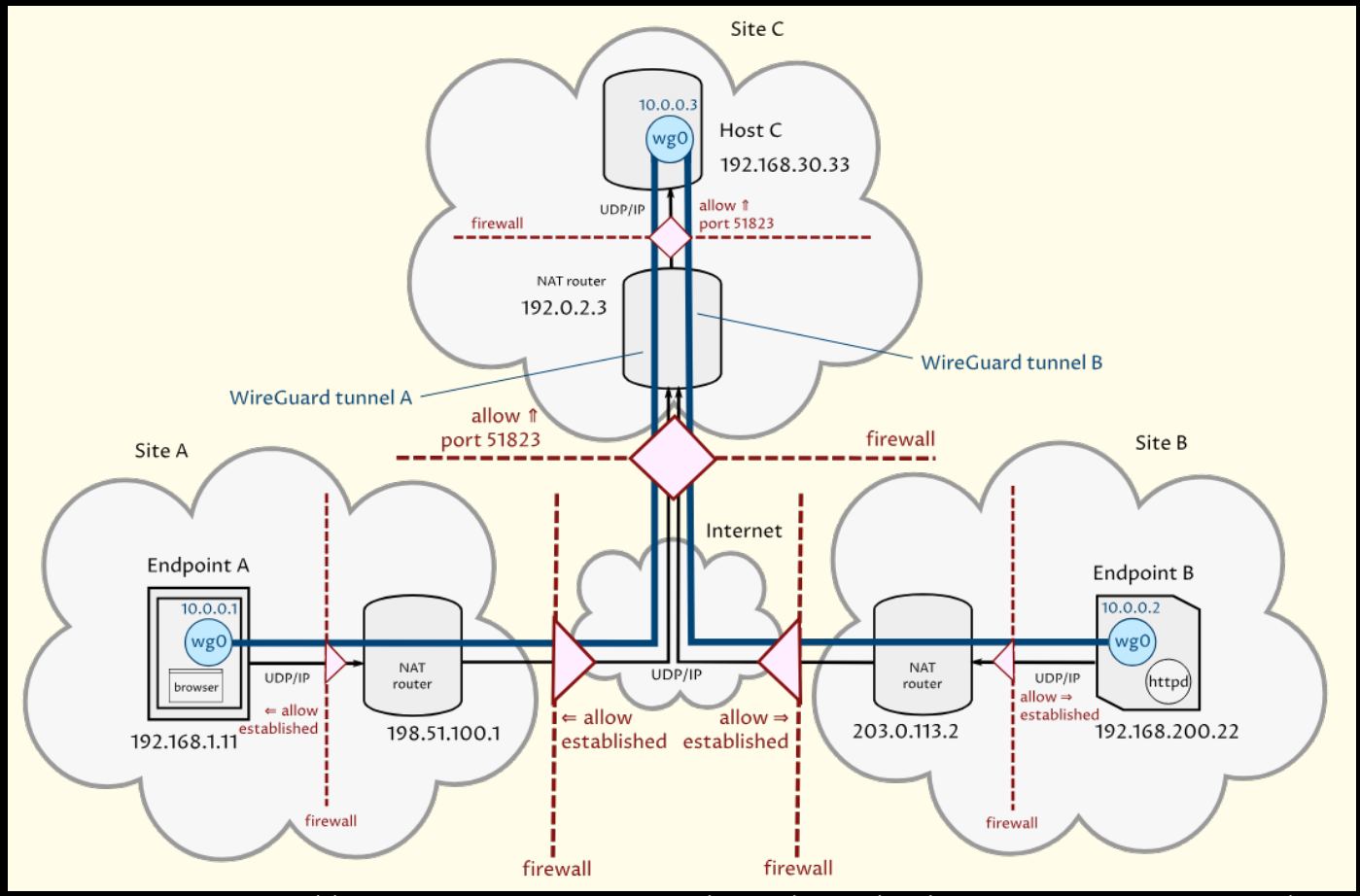
source : https://www.procustodibus.com/blog/2020/10/wireguard-topologies/

Hub and Spoke



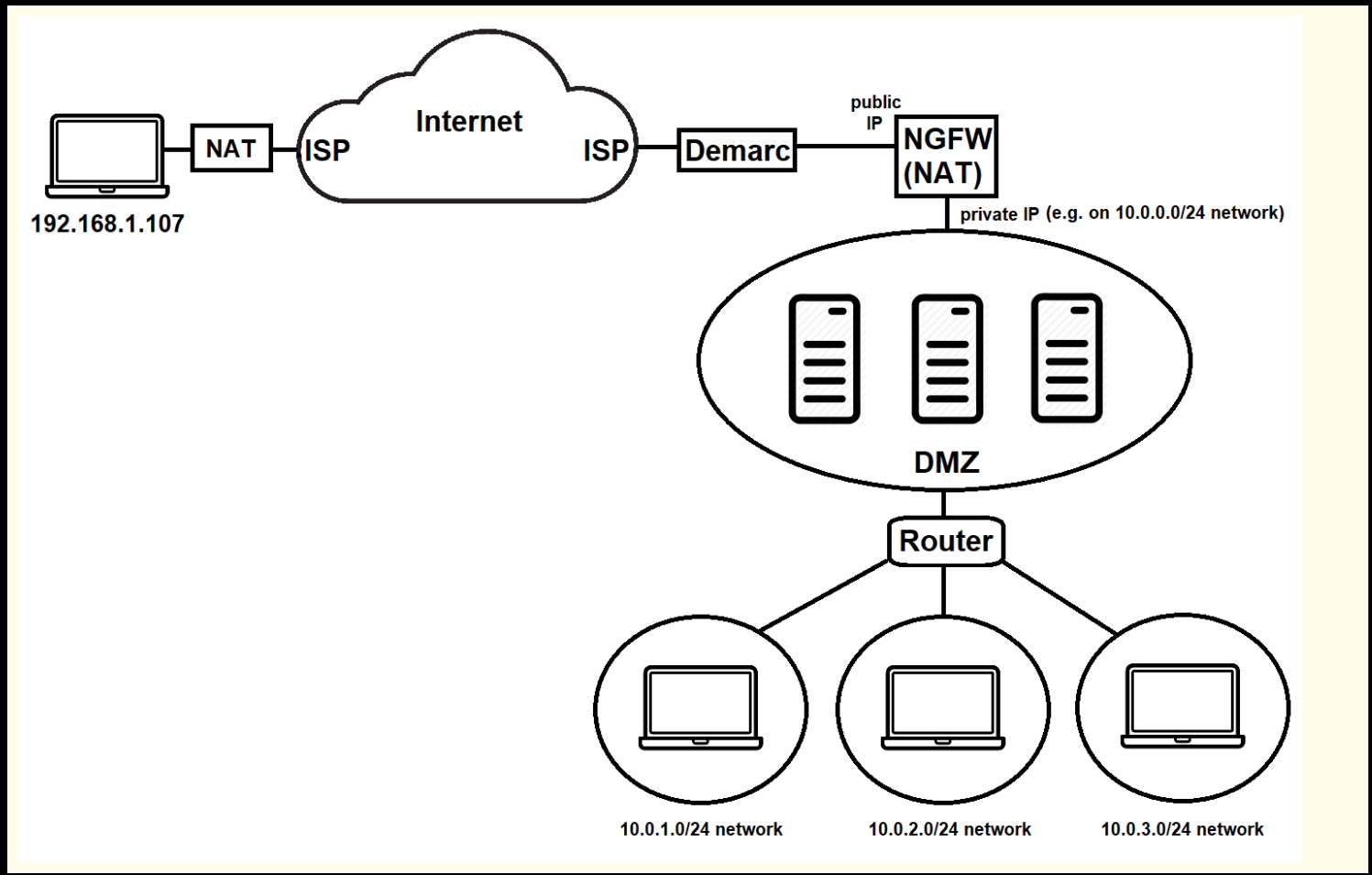
source : https://www.procustodibus.com/blog/2020/10/wireguard-topologies/

Hub and Spoke avec firewall



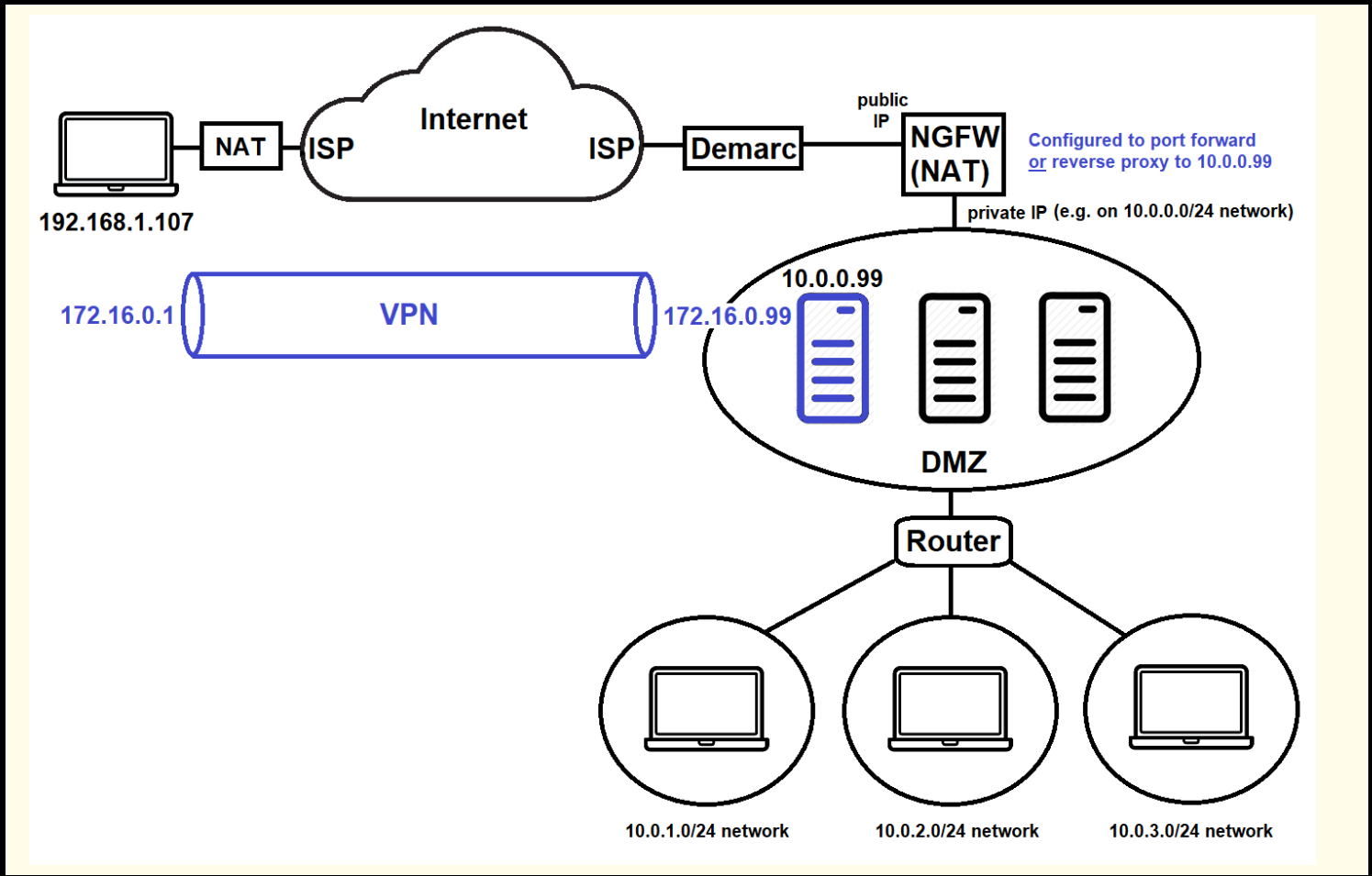
source : https://www.procustodibus.com/blog/2020/10/wireguard-topologies/

VPN + DMZ



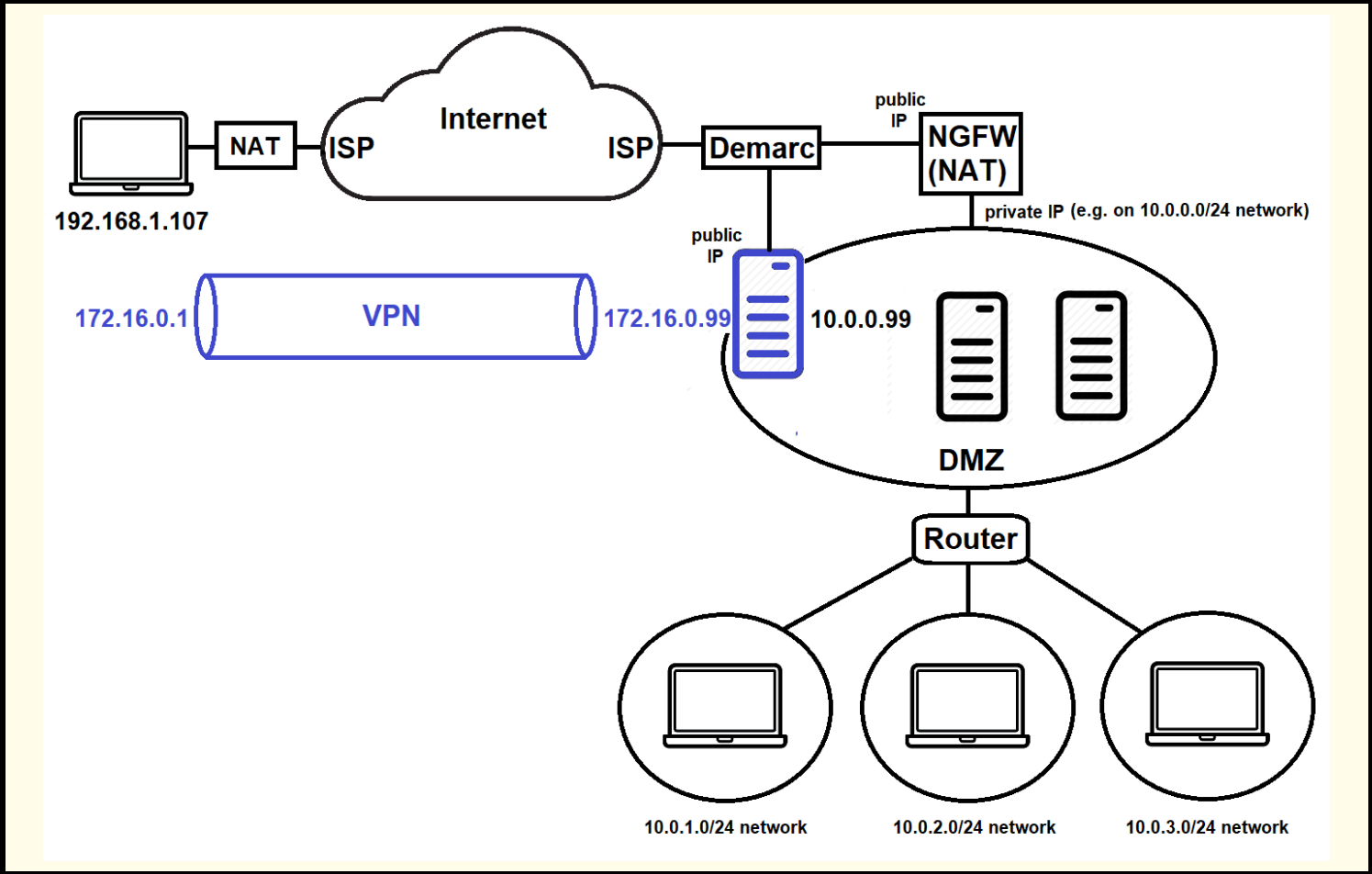
source : https://www.procustodibus.com/blog/2020/10/wireguard-topologies/

VPN + DMZ



source : https://www.procustodibus.com/blog/2020/10/wireguard-topologies/

VPN + DMZ



source : https://www.procustodibus.com/blog/2020/10/wireguard-topologies/

Synthèse

- WireGuard = sécurité + performance via simplicité
- Adoption croissante

Ressources

- https://www.wireguard.com
- Jason A. Donenfeld. *WireGuard: Next Generation Kernel Network Tunnel*. 2017. Whitepaper fondateur de WireGuard. Disponible en ligne : https://www.wireguard.com/papers/wireguard.pdf
- Stephen Kent, Karen Seo. *Security Architecture for the Internet Protocol*. RFC 4301, IETF, 2005. Disponible en ligne : https://datatracker.ietf.org/doc/html/rfc4301
- OpenVPN Project. *OpenVPN 2.5 Reference Manual*. 2021. Disponible en ligne : https://openvpn.net/community-resources/reference-manual-for-openvpn-2-5/
- Janik Dreier et al. *A Comparative Study on Virtual Private Networks*. Karlsruhe Institute of Technology, 2022. Disponible en ligne : https://publikationen.bibliothek.kit.edu/1000162550