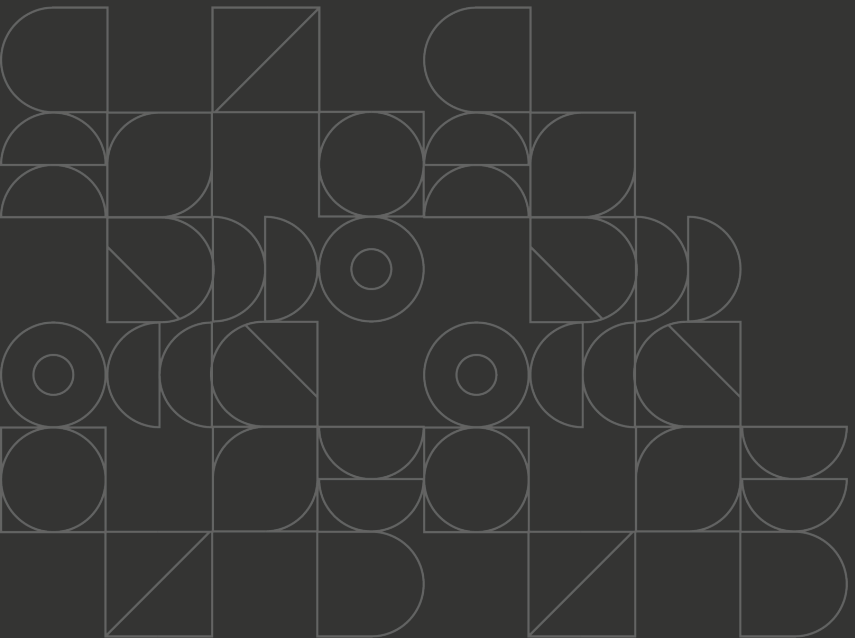
WireGuard from the ground up

BSidesSF | June 4 2022



Maya Kaczorowski & David Crawshaw

tailscale

@MayaKaczorowski, @davidcrawshaw

David Crawshaw
CTO
he/him

Maya Kaczorowski
Product Manager
she/her

tailscale

BSIDES 2022 SAN FRANCISCO
FROM THE GROUND UP

Agenda

- What WireGuard is
- Why WireGuard is different
- How WireGuard works
  - Handshake
  - Key rotation, timers
  - Cryptography

There’s nothing we’re covering you can’t also read in the WireGuard whitepaper

WireGuard is a layer 3 tunneling protocol that lets two peers privately establish an end-to-end encrypted connection

Designed for **security**, **performance**, and **ease of use**

modern cryptography

auditable

formally verified

perfect forward secrecy

kernel and userspace implementations

fast despite not relying on hardware crypto

can handle load

appears stateless

key-based identity

opinionated

handles key rotation and keep-alive

@MayaKaczorowski @davidcrawshaw

WireGuard vs TLS

- No client/server, a tunnel is made of two equal peers
- No CA roots, up to you to distribute public keys (like ssh)
- UDP, not TCP
  - No standardized port. No response if key is not recognized. Avoids:

```
$ nc ssh.mycompany.com 22
SSH-2.0-OpenSSH_8.4p1 Debian-5
```

- No protocol versioning.
  - Whatever WireGuard V2 ends up being built of (or named), the packet format will change.

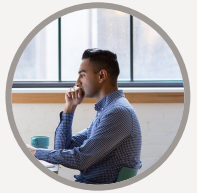
How WireGuard works

Alice

Static (host) keys

priv pub

Bob



Static (host) keys

pub priv

WireGuard configuration

Share public keys and source IPs

WireGuard protocol

Ephemeral (session) keys

priv pub

1-RTT key exchange

Ephemeral (session) keys

pub priv

Session key establishment

Data exchange

@MayaKaczorowski @davidcrawshaw

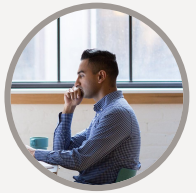
WireGuard protocol: handshake initiation

Alice

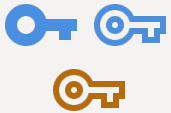
- Her static keys
- Bob's static public key

Ephemeral (session) keys

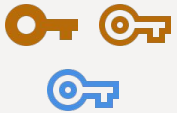
Bob



- His static keys
- Alice's static public key



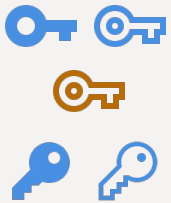
- Alice shares her ephemeral public key
- Alice's static public key, encrypted with both the ephemeral private key and Bob's static public key
- Starts a timer for the response



WireGuard protocol: handshake response

Alice

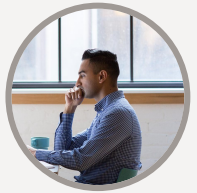
- Her static keys
- Bob's static public key
- Her ephemeral keys



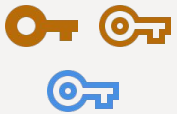
Ephemeral (session) keys

- Bob shares his ephemeral public key
- Bob confirms the session with Alice
- By being able to reply to Alice's message, Bob's reply authenticates Bob

Bob

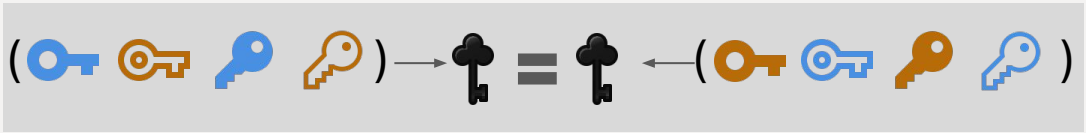


- His static keys
- Alice's static public key
- Alice's ephemeral public key

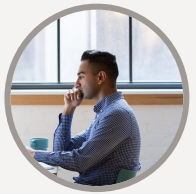


WireGuard protocol: generate session key

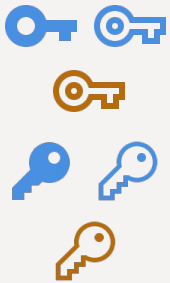
Alice



Bob

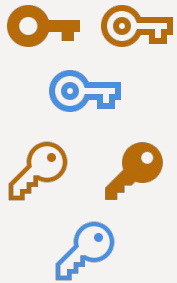


- Her static keys
- Bob's static public key
- Her ephemeral keys
- Bob's ephemeral public key



Alice computes:

Alice's private static key	+	Bob's public ephemeral key	Used for generating a session key
Alice's private ephemeral key	+	Bob's public static key	
Alice's private ephemeral key	+	Bob's public ephemeral key	
Alice's private static key	+	Bob's public static key	Used for authentication



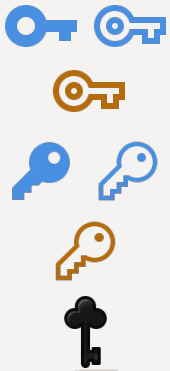
- His static keys
- Alice's static public key
- His ephemeral keys
- Alice's ephemeral public key

@MayaKaczorowski @davidcrawshaw

WireGuard protocol: data exchange

Alice

- Her static keys
- Bob's static public key
- Her ephemeral keys
- Bob's ephemeral public key
- A shared session key



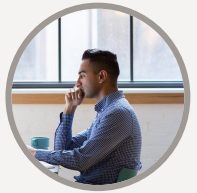
At this point we have a shared ChaCha20 session key

The overhead:

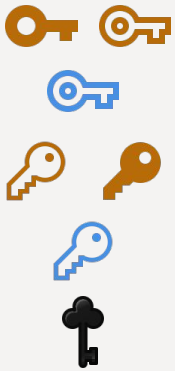
```
20/40 bytes IPv4/v6 header
    8 byte  UDP header
    4 byte  type
    4 byte  key index
    8 byte  nonce
    … encrypted data
   16 byte  auth tag
```

Total: 60 or 80 bytes per packet

Bob



- His static keys
- Alice's static public key
- His ephemeral keys
- Alice's ephemeral public key
- A shared session key



2022 SAN FRANCISCO BSIDES FROM THE GROUND UP

Timers and key rotation

- Handshake initiation is retried after a timeout, with jitter (to avoid both sides constantly racing a handshake init).
- Periodically, a new handshake is initiated to reset session keys.
- Ephemeral and session keys are zeroed after a fixed time, regardless of data transmitted.
- (and more)

WireGuard's cryptography

- Key exchange protocol: NoiseIK
- For optional pre-shared symmetric key: 256-bit random key
- For static and ephemeral keys: Curve25519
- For key derivation: HKDF
- For symmetric encryption: ChaCha20-Poly1305 (AEAD)
- For hashing: BLAKE2s

Noise_IKpsk2_25519_ChaChaPoly_BLAKE2s

Demo

```
$ wg set wg0 peer
MCowBQYDK2VuAyEASvp5enJ8rEZ
jIf7bPIssNNfLpH1B/yOGDHhcZ0
dCflE= allowed-ips
10.0.0.10/32 endpoint
192.168.1.1:51820
```

- Set up WireGuard configs on two devices
- Get them talking to each other

Summary

- WireGuard is a layer 3 tunnelling protocol that lets two peers privately establish an end-to-end encrypted connection
- In WireGuard, two peers complete a 1-RTT handshake, generate shared session keys, and then can communicate with end-to-end encryption
    - WireGuard is stateless to the user, handling key rotation and keep alive
    - WireGuard uses modern cryptography

Learn more

WireGuard: wireguard.com

Generate a config: wireguardconfig.com

Whitepaper: wireguard.com/papers/wireguard.pdf

Cryptography: wireguard.com/protocol

Black Hat 2018 talk: youtube.com/watch?v=88GyLoZbDNw

NoiseIK explainer: noiseexplorer.com/patterns/IK

Get these slides: bit.ly/3z4hZh5

Get some goodies: tailscale.com/bsides-sf