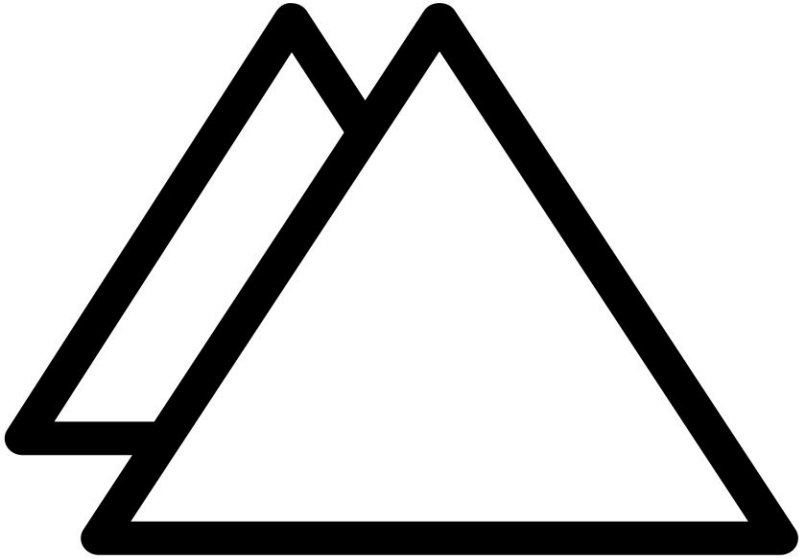
# Kubelet to Istio: Kubernetes Network Security Demystified

@sublimino and @controlplaneio


I’m:
- Andy
- Dev-like
- Sec-ish
- Ops-y



controlplane

# What is Network Security


controlplane

# Why do we need Network Security?


controlplane

# Happy Path Application Design

controlplane

# How Applications Run in “Piratical Reality™”

controlplane

# How Kubernetes does it

kubernetes

controlplane

# Self Signed Certs. Always a bad thing?


controlplane

# Takeaway: Encrypt Everything Everywhere

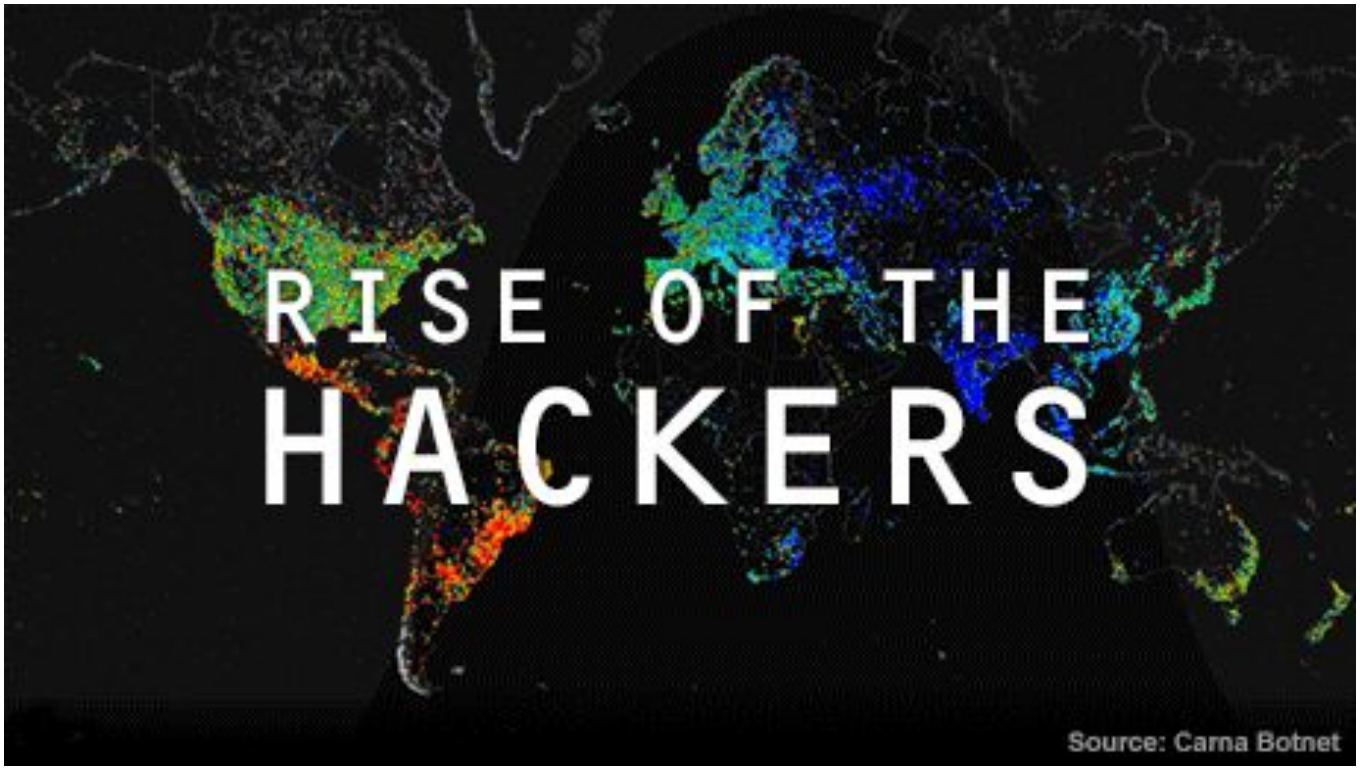
controlplane

# What this talk is about

- Network Security 101
- Kubernetes API Components
- TLS, X.509, and Mutual Authentication
- CNI and Network Policies for Applications
- Bootstrapping Identity with SPIFFE

# Network Security 101

# Private & Trusted Communications

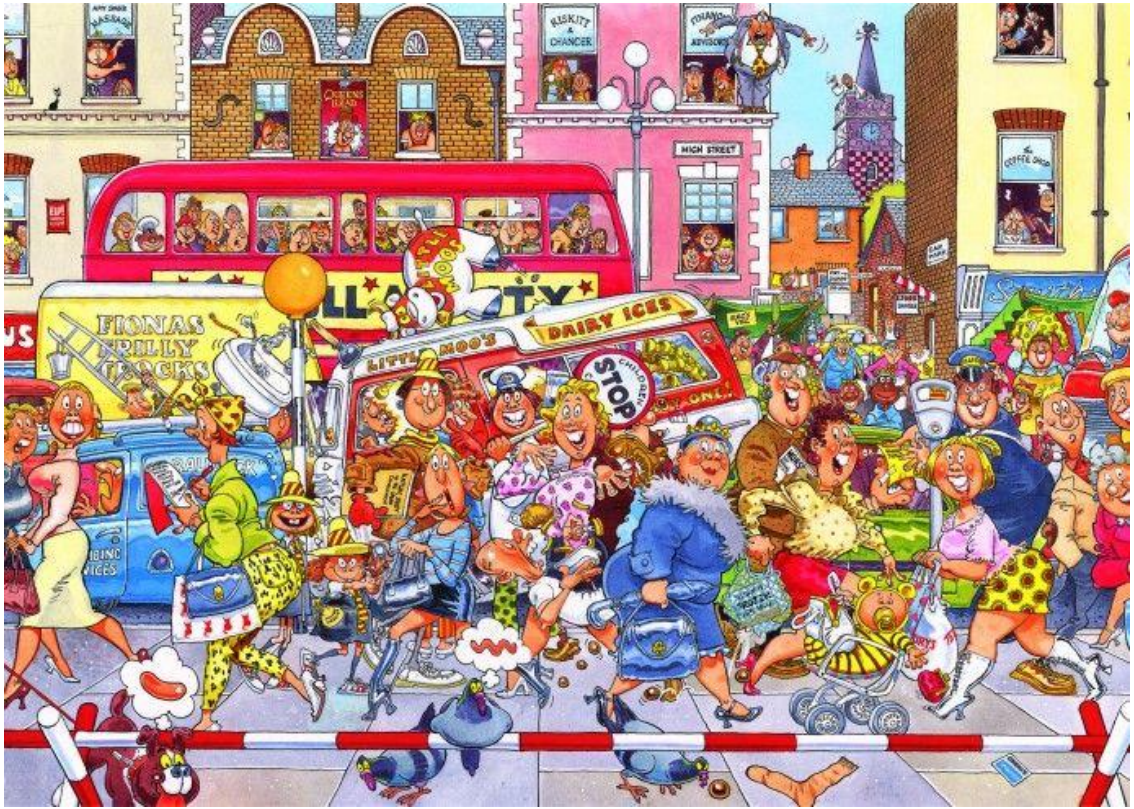



controlplane

# Human Communication: Trusted and Local

controlplane

# Human Communication: Untrusted and Local




controlplane

# Human Communication: Untrusted and Remote

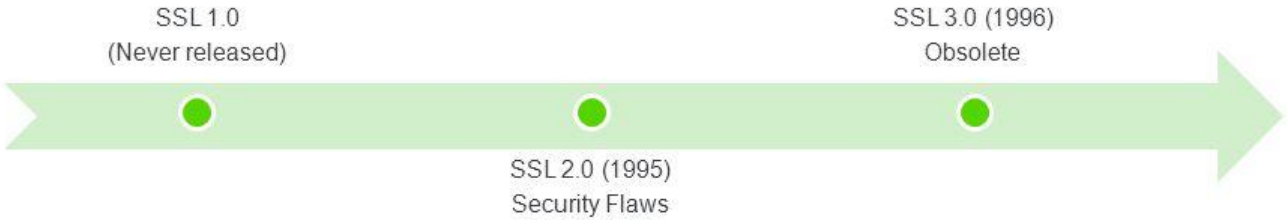
controlplane

# In Internet Prehistory...

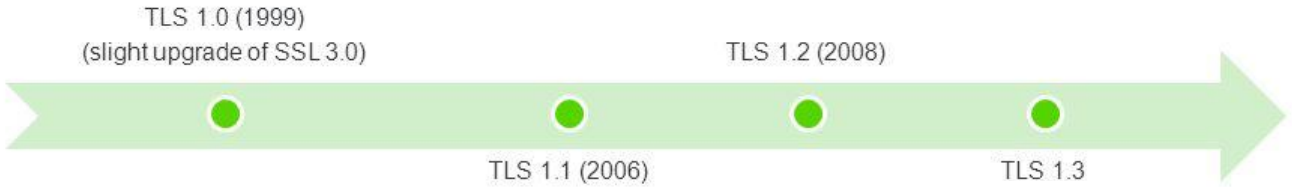

controlplane

# SSL/TLS – A brief history lesson

## Secure Socket Layer (SSL) – Originated in Netscape Web browser




## Transport Layer Security(TLS) – Standardized by IETF






http:/slideplayer.com/slide/11456904/

# Is it really that simple?

controlplane

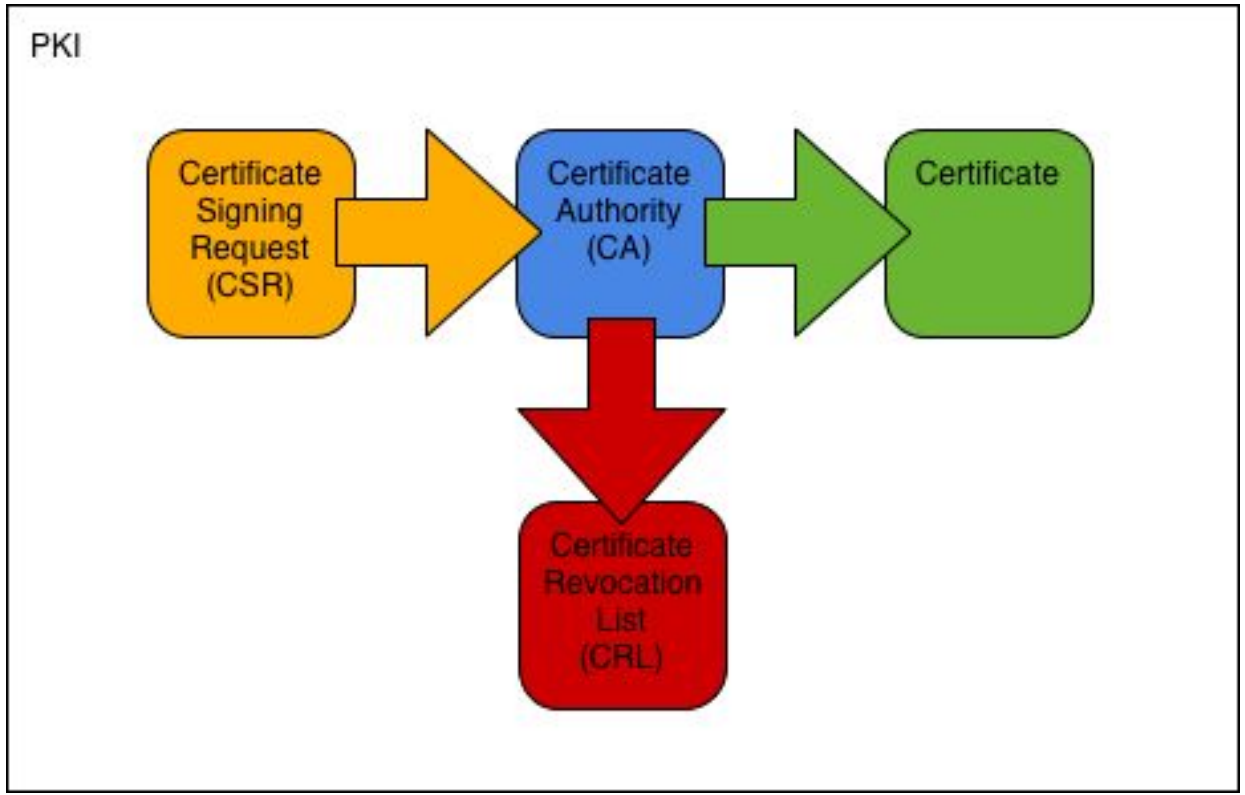

controlplane

# Securing API Server Traffic

# Securing API Server Traffic

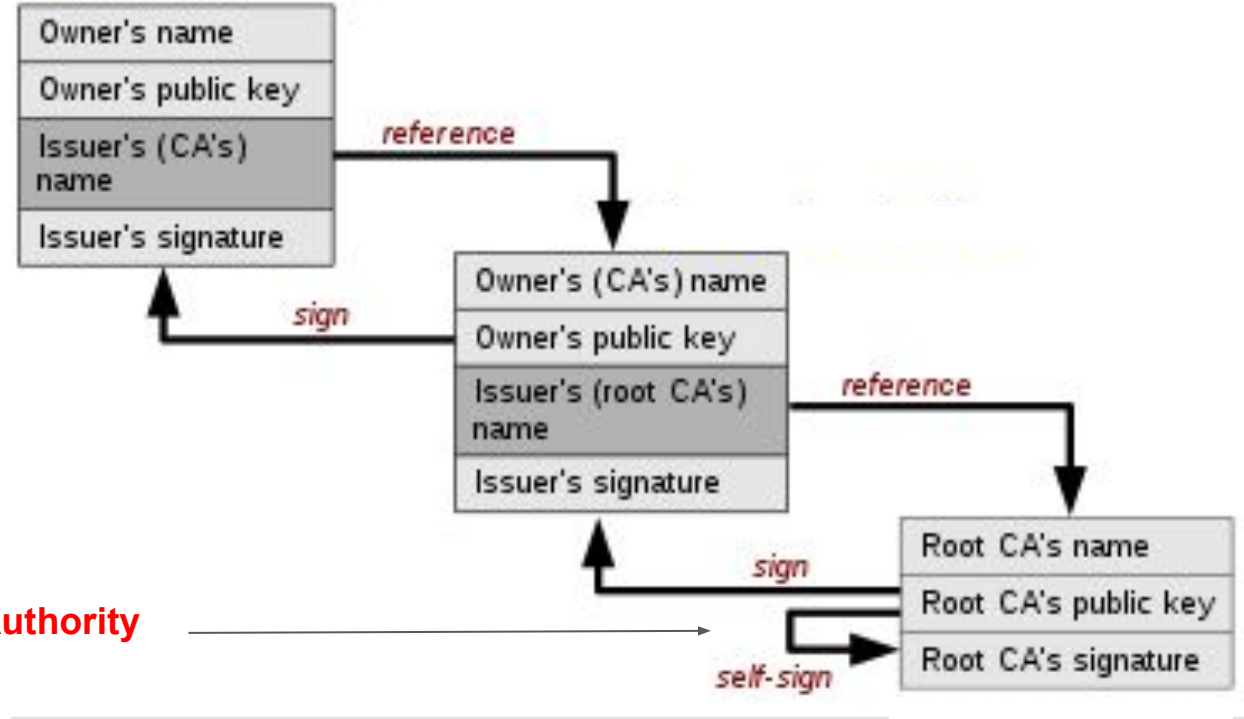
```
kube-apiserver
...
--client-ca-file=/secret/authca.pem
...
--etcd-cafile=/secret/ca.pem
--etcd-certfile=/secret/cert.pem
--etcd-keyfile=/secret/key.pem
--experimental-encryption-provider-config=/secret/encryption.cfg
...
--kubelet-certificate-authority=/secret/ca.pem
--kubelet-client-certificate=/secret/cert.pem
--kubelet-client-key=/secret/key.pem
...
--oidc-ca-file=/secret/ca.pem
...
--service-account-key-file=/secret/service_account_key.pem
...
--tls-ca-file=/secret/ca.pem
--tls-cert-file=/secret/cert.pem
--tls-private-key-file=/secret/key.pem
--tls-sni-cert-key=/secret/cert.pem,/secret/key.pem:localhost
--tls-sni-cert-key=/secret/controller/cert.pem,/secret/controller/key.pem
...
```

# Securing API Server Traffic




controlplane

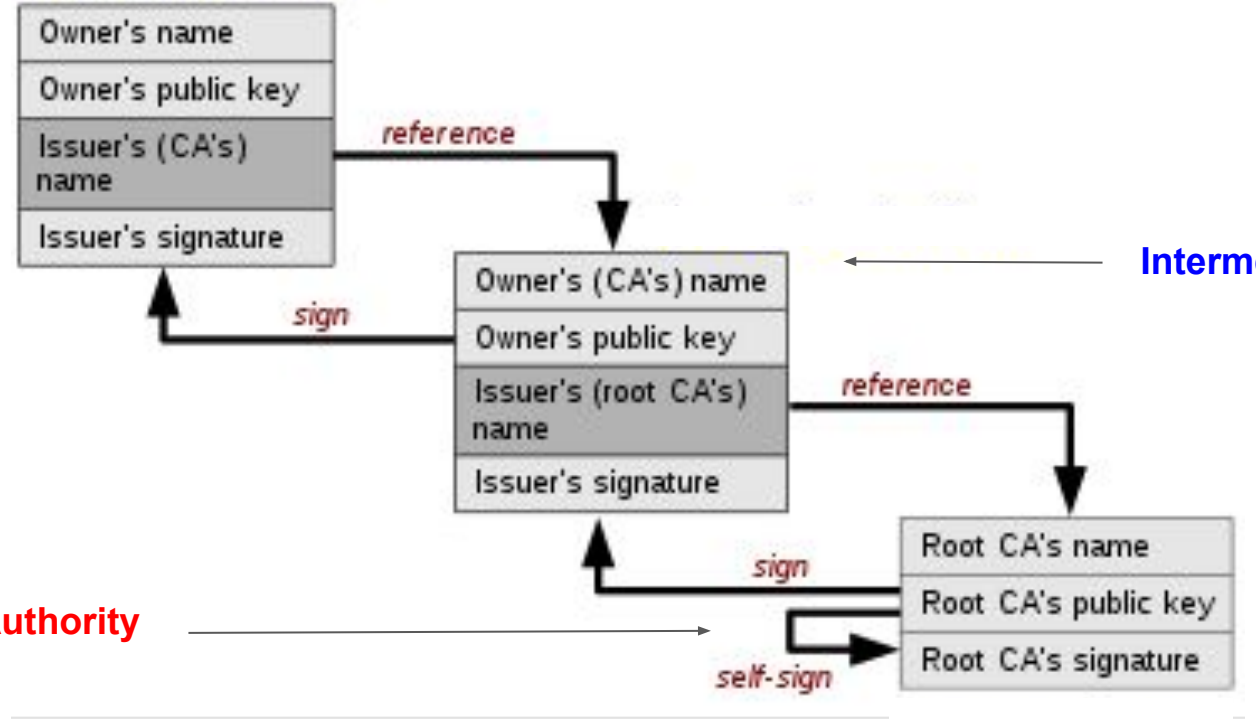
# Securing API Server Traffic




**Certificate Authority**

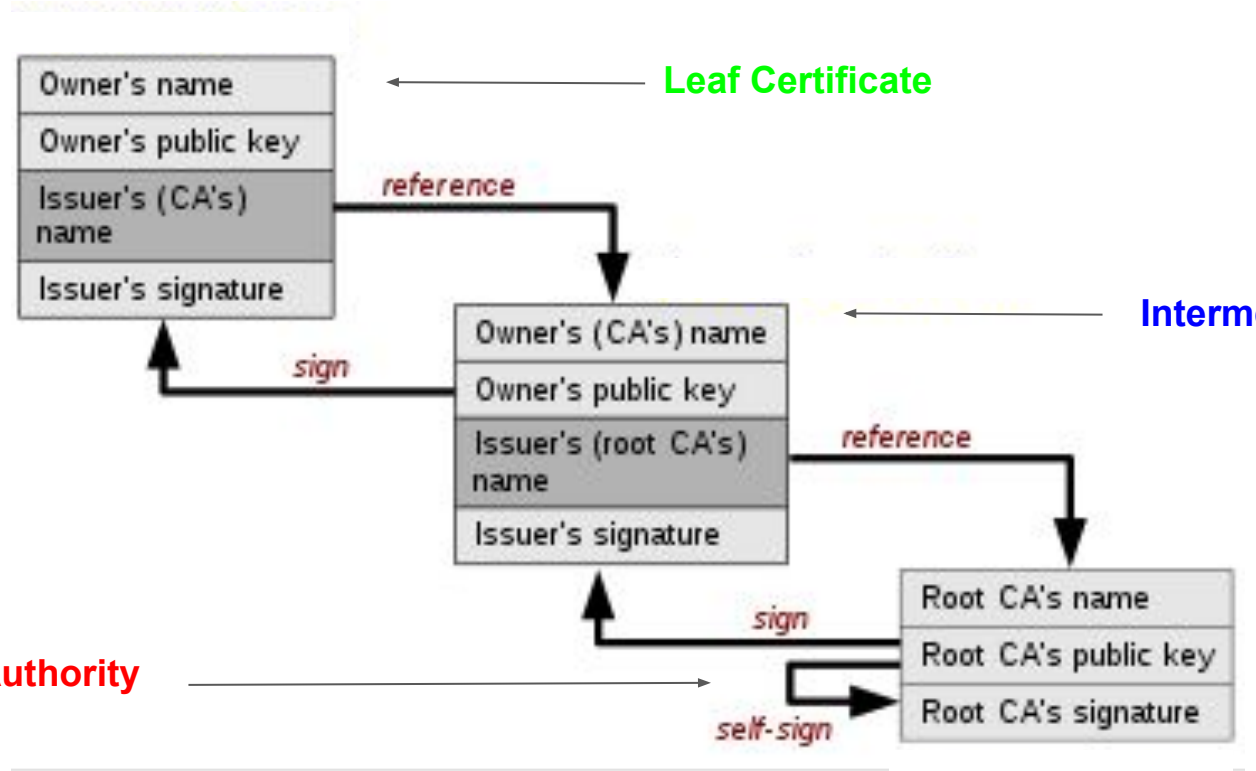
controlplane

# Securing API Server Traffic




controlplane

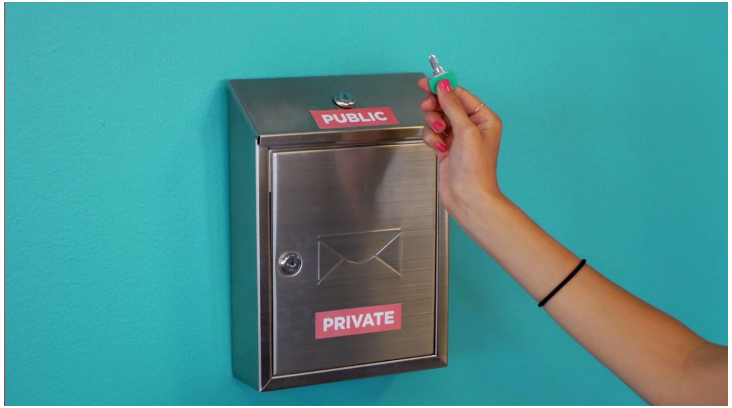
# Securing API Server Traffic




controlplane

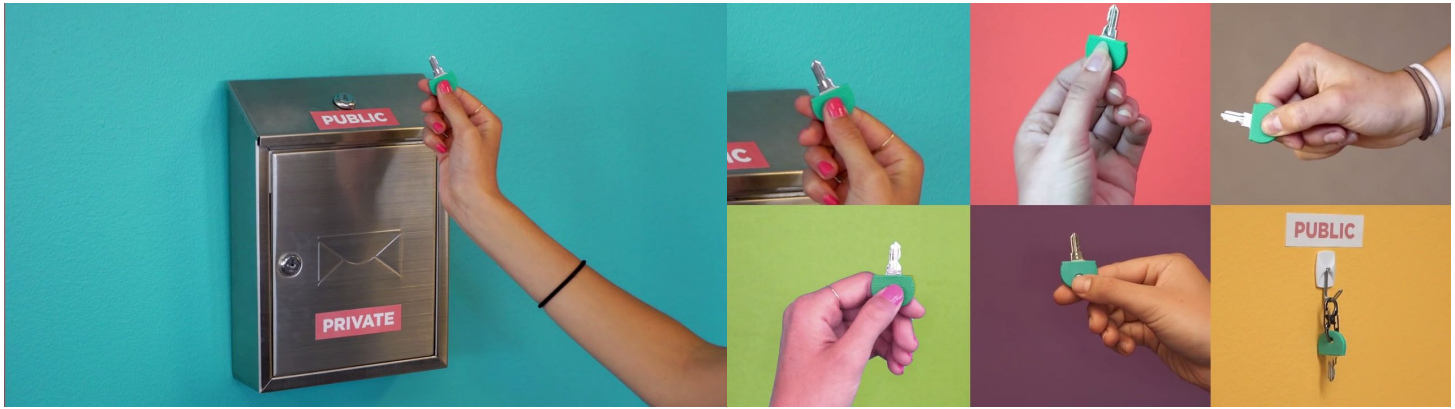
# TLS, X.509, and mutual authentication

# Public Key Cryptography



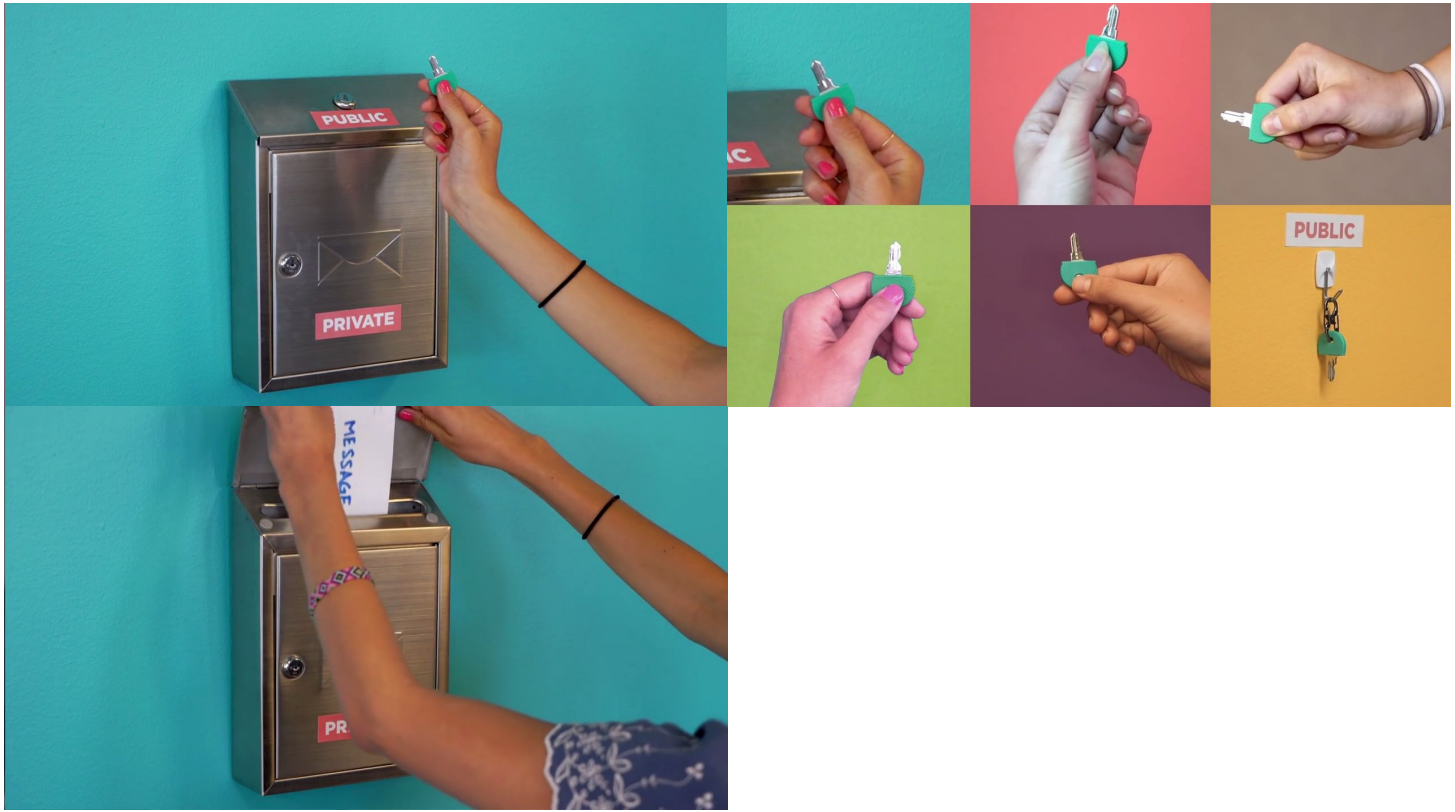

controlplane

# Public Key Cryptography




controlplane

# Public Key Cryptography




controlplane

# Public Key Cryptography


controlplane

# Woah there, how does it work?

# Woah there, how does it work?

- Symmetric encryption

# Woah there, how does it work?

- Symmetric encryption: identical keys to lock and unlock

# Woah there, how does it work?

- Symmetric encryption: identical keys to lock and unlock
- Asymmetric encryption

# Woah there, how does it work?

- Symmetric encryption: identical keys to lock and unlock
- Asymmetric encryption: different keys to lock and unlock

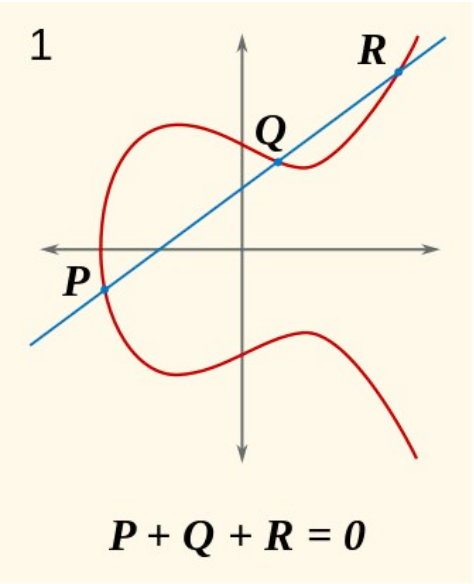
# Woah there, how does it work?

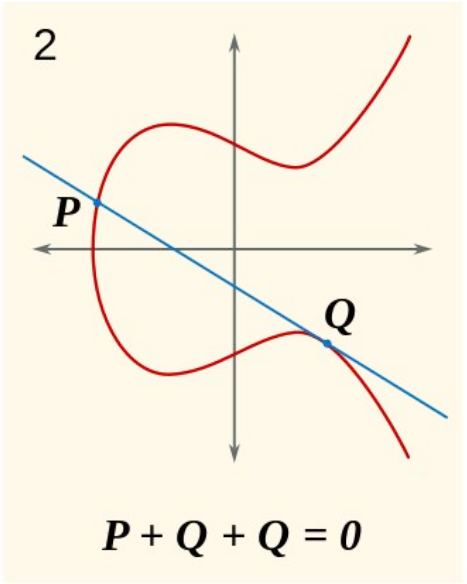
- Symmetric encryption: identical keys to lock and unlock
- Asymmetric encryption: different keys to lock and unlock
- Elliptic-curve cryptography

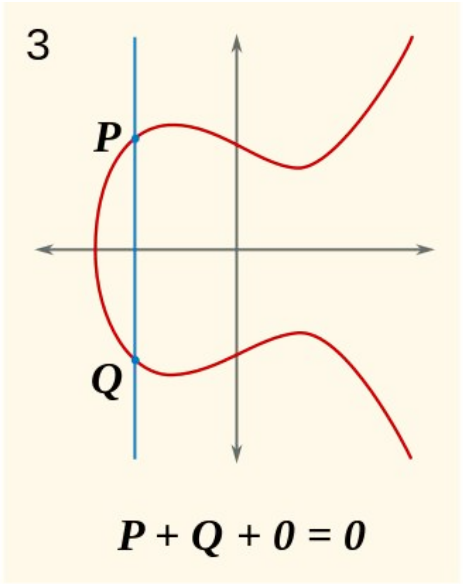
# Woah there, how does it work?

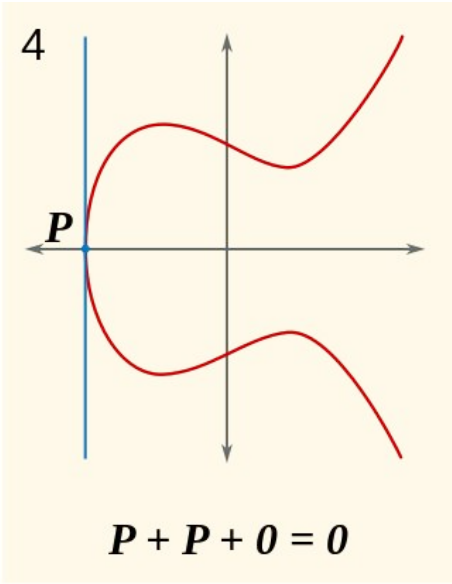
- Symmetric encryption: identical keys to lock and unlock
- Asymmetric encryption: different keys to lock and unlock
- Elliptic-curve cryptography: using the properties of certain graphed shapes to make brute forcing more difficult

# Elliptic Curves




controlplane

# Woah there, how does it work?

- Symmetric encryption: identical keys to lock and unlock
- Asymmetric encryption: different keys to lock and unlock
- Elliptic-curve cryptography: using the properties of certain graphed shapes to make brute forcing more difficult
- Diffie-Hellman

controlplane

# Woah there, how does it work?

- Symmetric encryption: identical keys to lock and unlock
- Asymmetric encryption: different keys to lock and unlock
- Elliptic-curve cryptography: using the properties of certain graphed shapes to make brute forcing more difficult
- Diffie-Hellman: a way to create a shared encryption key without ever communicating it publically

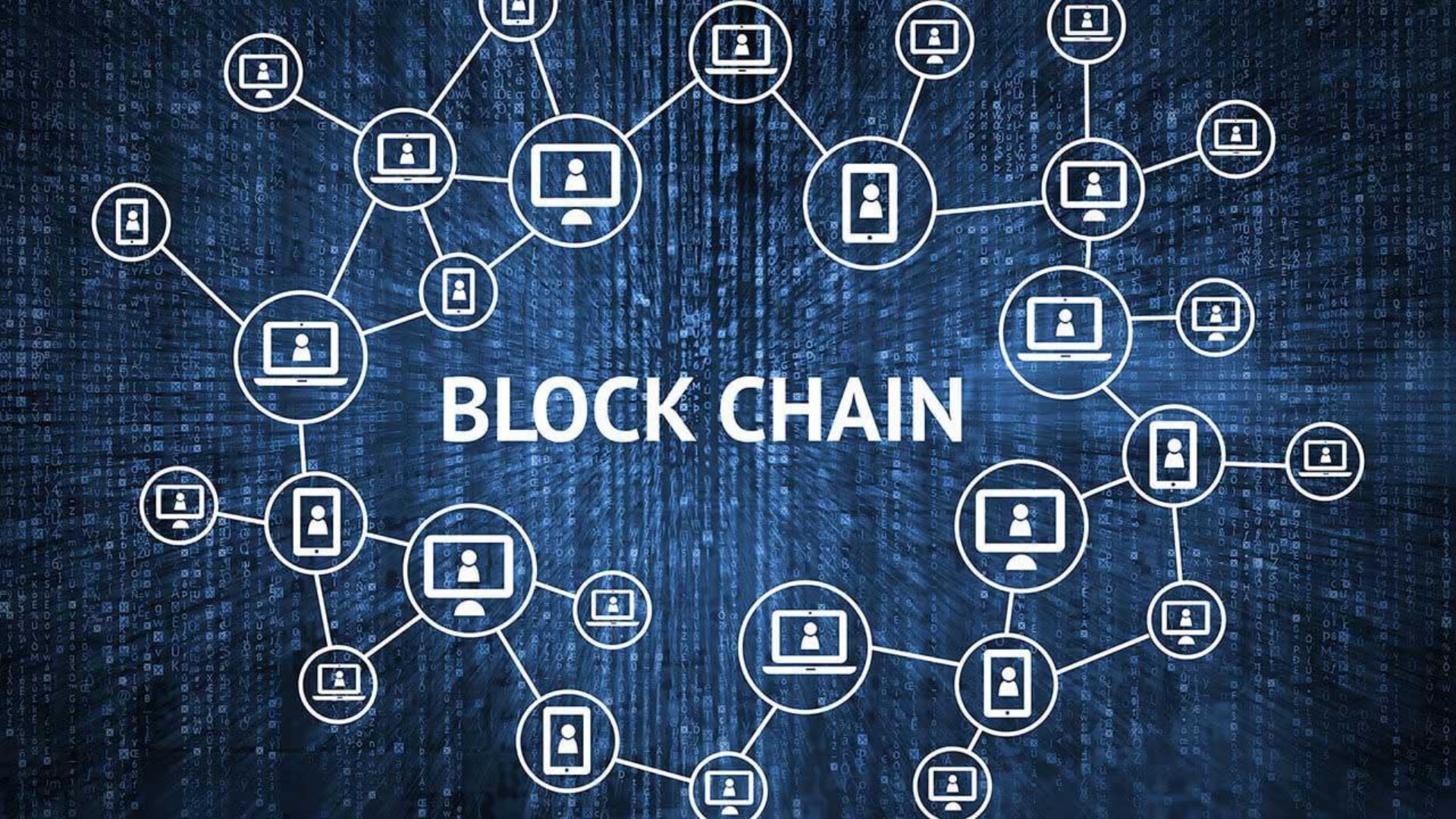
# Cryptography!

- Symmetric encryption: identical keys to lock and unlock
- Asymmetric encryption: different keys to lock and unlock
- Elliptic-curve cryptography: using the properties of certain graphed shapes to make brute forcing more difficult
- Diffie-Hellman: a way to create a shared encryption key without ever communicating it publically

# More Info on TLS

- https://howhttps.works/ by DNSimple

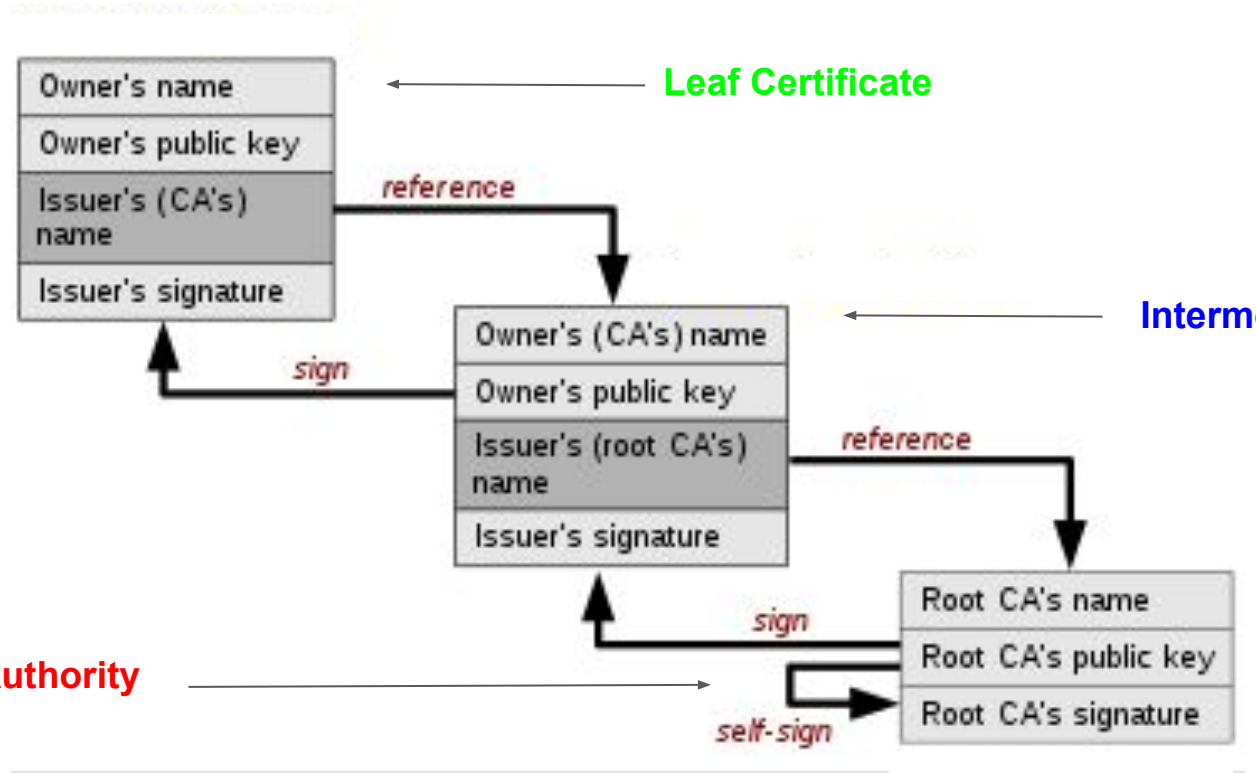
# The Original Crypto!


BLOCK CHAIN

# TLS in Kubernetes

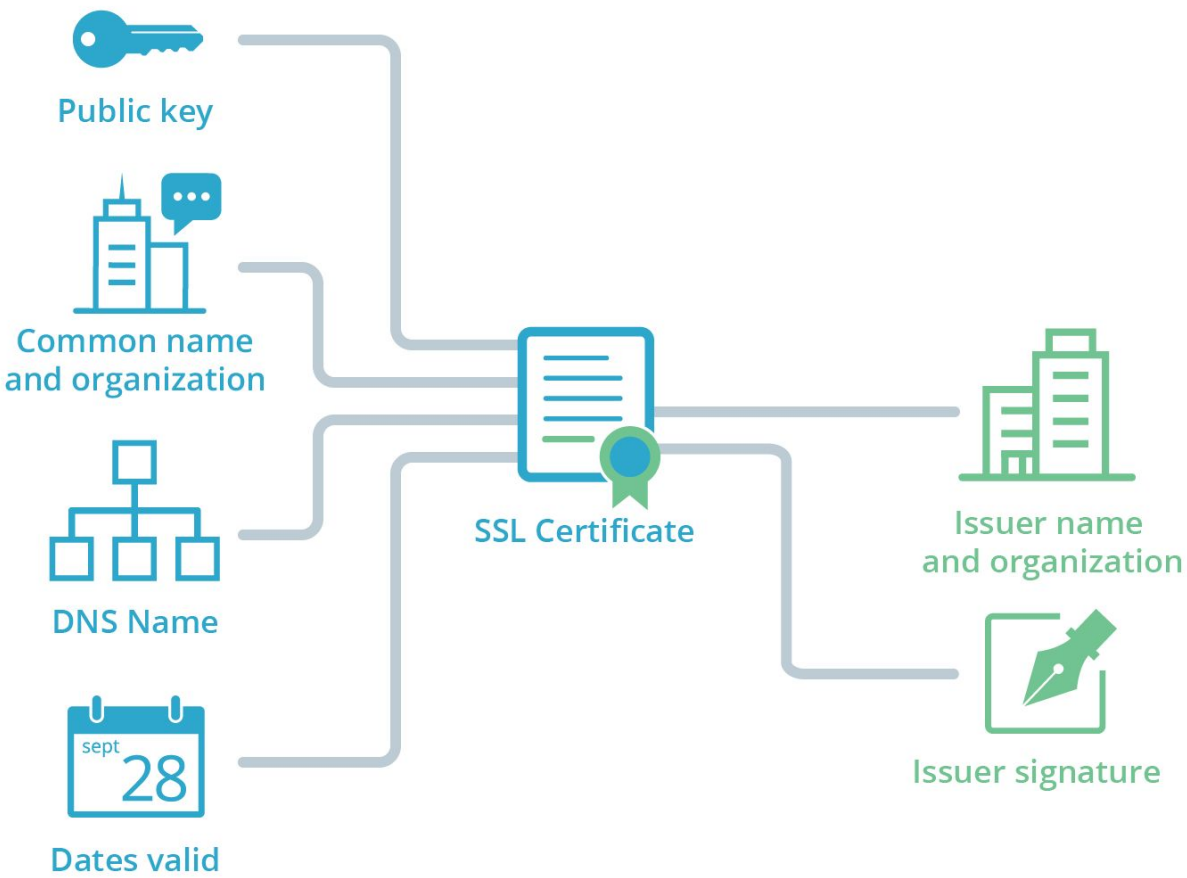
controlplane

# Certificate Path Validation




controlplane

# X.509




https://blog.cloudflare.com/content/images/2018/03/image8.png

controlplane

# X.509 RFC Format

```
Certificate  ::=  SEQUENCE  {
     tbsCertificate       TBSCertificate,
     signatureAlgorithm   AlgorithmIdentifier,
     signatureValue       BIT STRING  }

TBSCertificate  ::=  SEQUENCE  {
     version         [0]  EXPLICIT Version DEFAULT v1,
     serialNumber         CertificateSerialNumber,
     signature            AlgorithmIdentifier,
     issuer               Name,
     validity             Validity,
     subject              Name,
     subjectPublicKeyInfo SubjectPublicKeyInfo,
     issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version MUST be v2 or v3
     subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version MUST be v2 or v3
     extensions      [3]  EXPLICIT Extensions OPTIONAL
                          -- If present, version MUST be v3
     }

Version  ::=  INTEGER  {  v1(0), v2(1), v3(2)  }

CertificateSerialNumber  ::=  INTEGER
```

```
Validity ::= SEQUENCE {
     notBefore      Time,
     notAfter       Time }

Time ::= CHOICE {
     utcTime        UTCTime,
     generalTime    GeneralizedTime }

UniqueIdentifier  ::=  BIT STRING

SubjectPublicKeyInfo  ::=  SEQUENCE  {
     algorithm            AlgorithmIdentifier,
     subjectPublicKey     BIT STRING  }

Extensions  ::=  SEQUENCE SIZE (1..MAX) OF Extension

Extension  ::=  SEQUENCE  {
     extnID      OBJECT IDENTIFIER,
     critical    BOOLEAN DEFAULT FALSE,
     extnValue   OCTET STRING
                 -- contains the DER encoding of an ASN.1 value
                 -- corresponding to the extension type identified
                 -- by extnID
     }
```

https://tools.ietf.org/html/rfc5280#page-35

controlplane

# X.509 Example Cert

```
-----BEGIN CERTIFICATE-----
MIIC2jCCAkMCAg38MA0GCSqGSIb3DQEBBQUAMIGbMQswCQYDVQQGEwJKUDEOMAwG
A1UECBMFVG9reW8xEDAOBgNVBAcTB0NodW8ta3UxETAPBgNVBAoTCEZyYW5rNERE
MRgwFgYDVQQLEw9XZWJDZXJ0IFN1cHBvcnQxGDAWBgNVBAMTD0ZyYW5rNEREIFdl
YiBDQTEjMCEGCSqGSIb3DQEJARYUc3VwcG9ydEBmcmFuazRkZC5jb20wHhcNMTIw
ODIyMDUyNzQxWhcNMTcwODIxMDUyNzQxWjBKMQswCQYDVQQGEwJKUDEOMAwGA1UE
CAwFVG9reW8xETAPBgNVBAoMCEZyYW5rNEREMRgwFgYDVQQDDA93d3cuZXhhbXBs
ZS5jb20wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC0z9FeMynsC8+u
dvX+LciZxnh5uRj4C9S6tNeeAlIGCfQYk0zUcNFCoCkTknNQd/YEiawDLNbxBqut
bMDZ1aarys1a0lYmUeVLCIqvzBkPJTSQsCopQQ9V8WuT252zzNzs68dVGNdCJd5J
NRQykpwexmnjPPv0mvj7i8XgG379TyW6P+WWV5okeUkXJ9eJS2ouDYdR2SM9BoVW
+FgxDu6BmXhozW5EfsnajFp7HL8kQClI0QOc79yuKl3492rH6bzFsFn2lfwWy9ic
7cP8EpCTeFp1tFaD+vxBhPZkeTQ1HKx6hQ5zeHIB5ySJJZ7af2W8r4eTGYzbdRW2
4DDHCPhZAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAQMv+BFvGdMVzkQaQ3/+2noVz
/uAKbzpEL8xTcxYyP3lkOeh4FoxiSWqy5pGFALdPONoDuYFpLhjJSZaEwuvjI/Tr
rGhLV1pRG9frwDFshqD2Vaj4ENBCBh6UpeBop5+285zQ4SI7q4U9oSebUDJiuOx6
+tZ9KynmrbJpTSi0+BM=
-----END CERTIFICATE-----
```

controlplane

# How to decode an X.509 Cert

```
$ openssl s_client -connect wikipedia.org:443
CONNECTED(00000003)
depth=2 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert High Assurance EV Root CA
verify return:1
depth=1 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert SHA2 High Assurance Server CA
verify return:1
depth=0 C = US, ST = California, L = San Francisco, O = "Wikimedia Foundation, Inc.", CN = *.wikipedia.org
verify return:1
---
Certificate chain
 0 s:/C=US/ST=California/L=San Francisco/O=Wikimedia Foundation, Inc./CN=*.wikipedia.org
   i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert SHA2 High Assurance Server CA
 1 s:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert SHA2 High Assurance Server CA
   i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert High Assurance EV Root CA
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIIfDCCB2SgAwIBAgIQCDCUYtH+pgrgur/174vFRTANBgkqhkiG9w0BAQsFADBw
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
d3cuZGlnaWNlcnQuY29tMS8wLQYDVQQDEyZEaWdpQ2VydCBTSEEyIEhpZ2ggQXNz
dXJhbmNlIFNlcnZlciBDQTAeFw0xNzEyMjEwMDAwMDBaFw0xOTAxMjQxMjAwMDBa
...
```

controlplane

# X.509 Example Decoded Cert

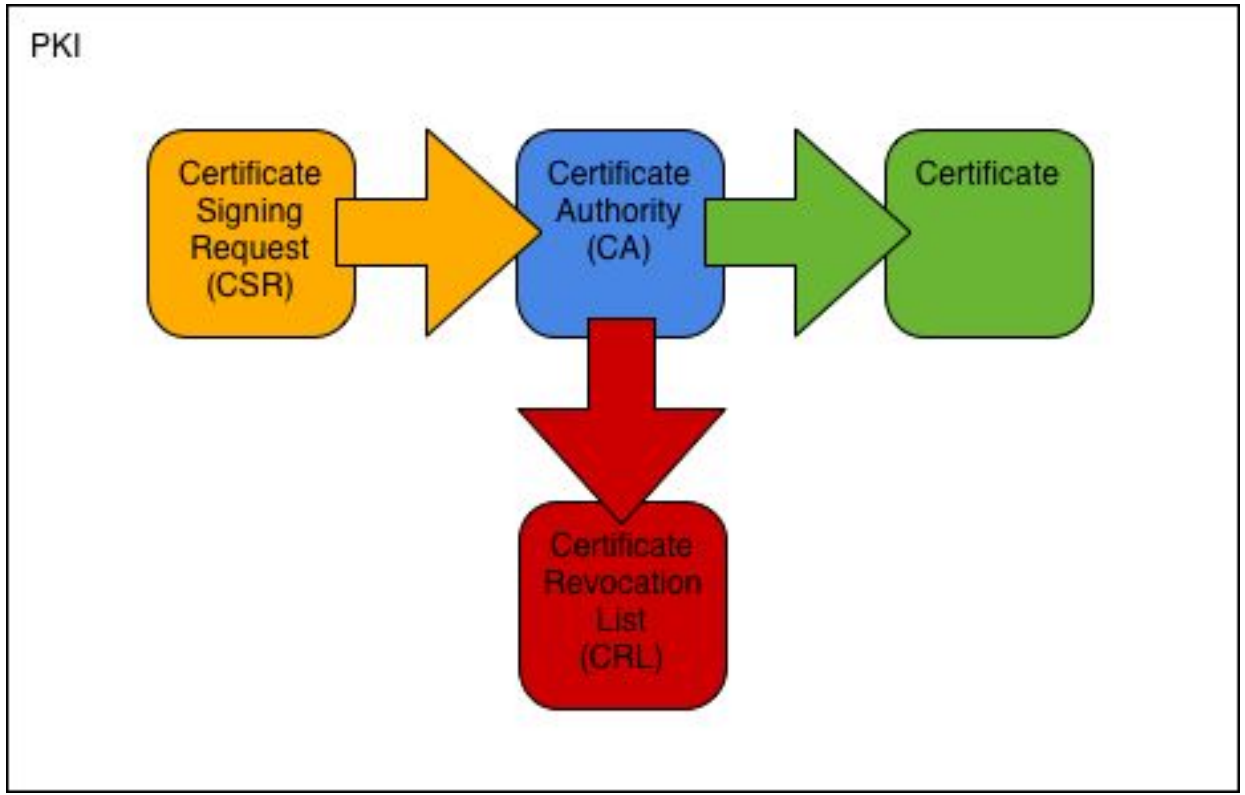
```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            10:e6:fc:62:b7:41:8a:d5:00:5e:45:b6
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=BE, O=GlobalSign nv-sa, CN=GlobalSign Organization Validation CA - SHA256 - G2
        Validity
            Not Before: Nov 21 08:00:00 2016 GMT
            Not After : Nov 22 07:59:59 2017 GMT
        Subject: C=US, ST=California, L=San Francisco, O=Wikimedia Foundation, Inc.,
CN=*.wikipedia.org
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:c9:22:69:31:8a:d6:6c:ea:da:c3:7f:2c:ac:a5:
                    af:c0:02:ea:81:cb:65:b9:fd:0c:6d:46:5b:c9:1e:
                    ed:b2:ac:2a:1b:4a:ec:80:7b:e7:1a:51:e0:df:f7:
                    c7:4a:20:7b:91:4b:20:07:21:ce:cf:68:65:8c:c6:
                    9d:3b:ef:d5:c1
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Agreement
            Authority Information Access:
                CA Issuers - URI:http://secure.globalsign.com/cacert/gsorganizationvalsha2g2r1.crt
                OCSP - URI:http://ocsp2.globalsign.com/gsorganizationvalsha2g2
```

```
X509v3 Certificate Policies:
                Policy: 1.3.6.1.4.1.4146.1.20
                  CPS: https://www.globalsign.com/repository/
                Policy: 2.23.140.1.2.2

            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 CRL Distribution Points:

                Full Name:
                  URI:http://crl.globalsign.com/gs/gsorganizationvalsha2g2.crl

            X509v3 Subject Alternative Name:
                DNS:*.wikipedia.org, DNS:*.m.mediawiki.org, DNS:*.m.wikibooks.org, ...
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
            X509v3 Subject Key Identifier:
                28:2A:26:2A:57:8B:3B:CE:B4:D6:AB:54:EF:D7:38:21:2C:49:5C:36
            X509v3 Authority Key Identifier:
                keyid:96:DE:61:F1:BD:1C:16:29:53:1C:C0:CC:7D:3B:83:00:40:E6:1A:7C

    Signature Algorithm: sha256WithRSAEncryption
         8b:c3:ed:d1:9d:39:6f:af:40:72:bd:1e:18:5e:30:54:23:35:
         ...
```

https://tools.ietf.org/html/rfc5280#page-35

controlplane

# Self Signed Certs aka Signing Your Own Homework




controlplane

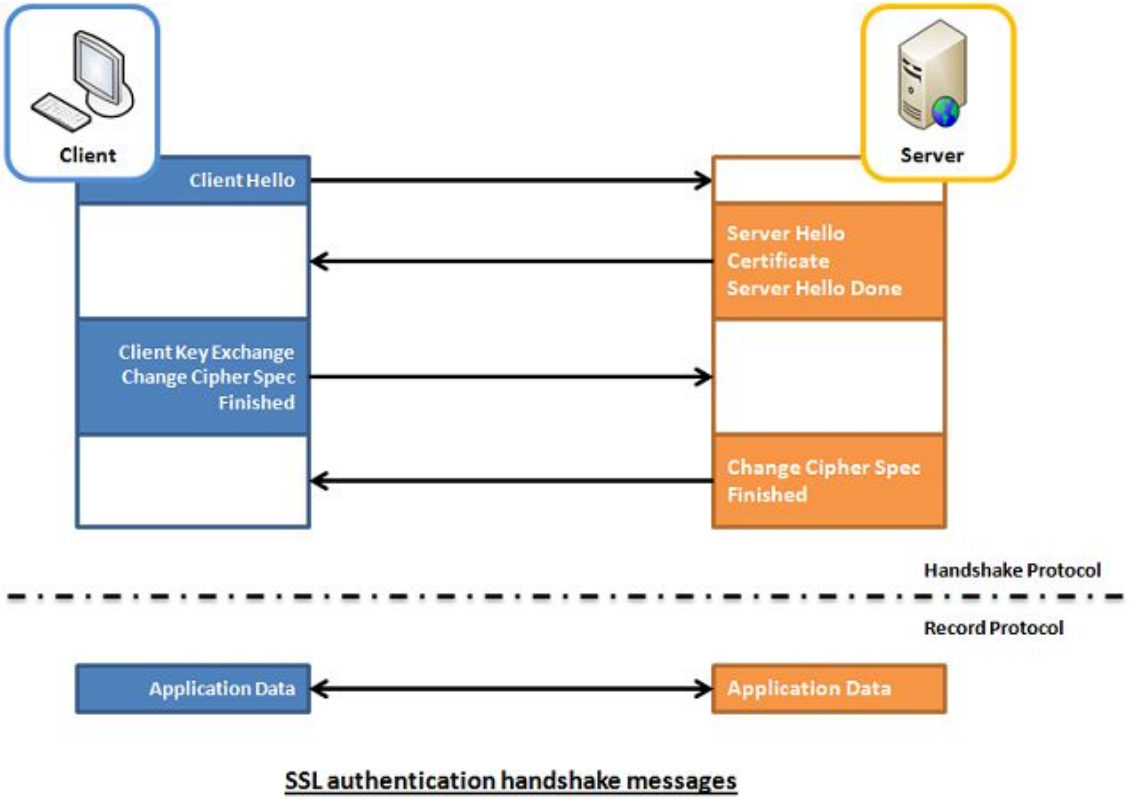
# PKI

PKI

# Private PKI and Self-Signed Certs

Private PKI and Self-Signed Certs

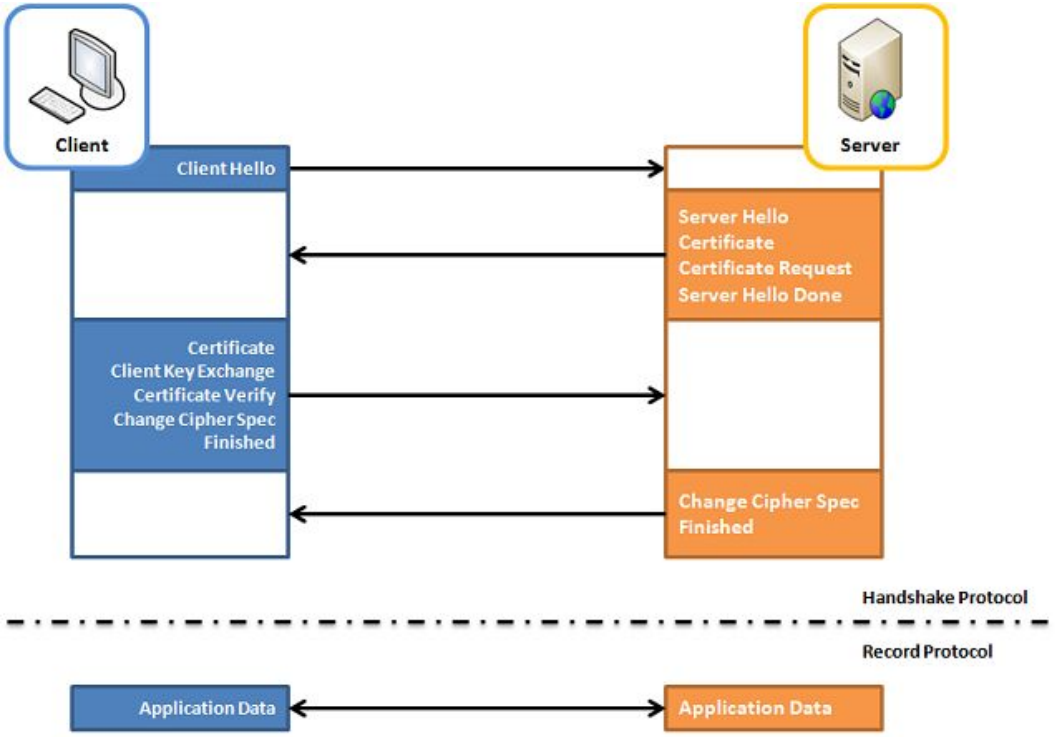
# One-Way (Traditional) TLS Handshake




SSL authentication handshake messages

controlplane

# Mutual TLS Handshake (mTLS)




Mutual SSL authentication handshake messages

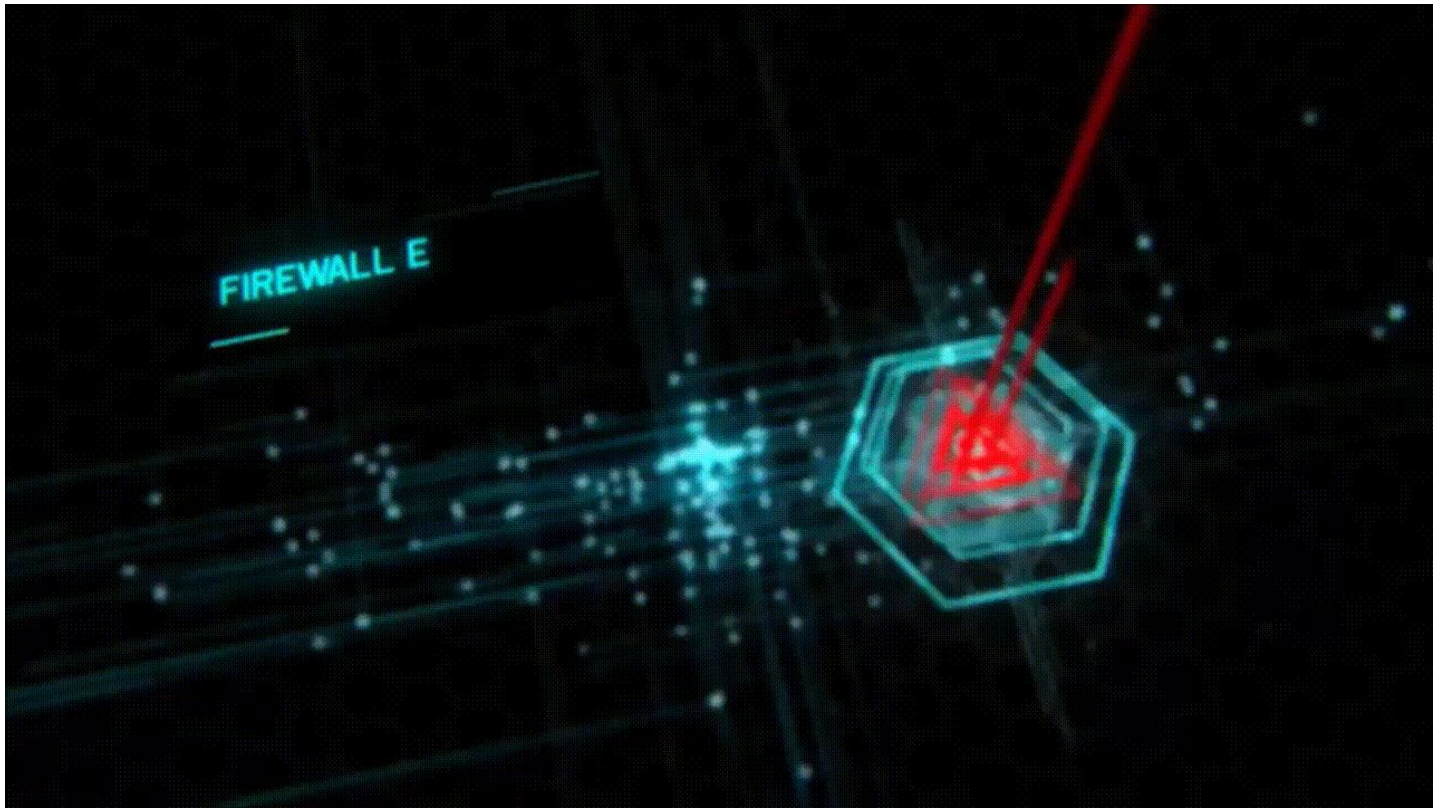
controlplane

# Private & Trusted Communications


controlplane

# Securing API Server Traffic

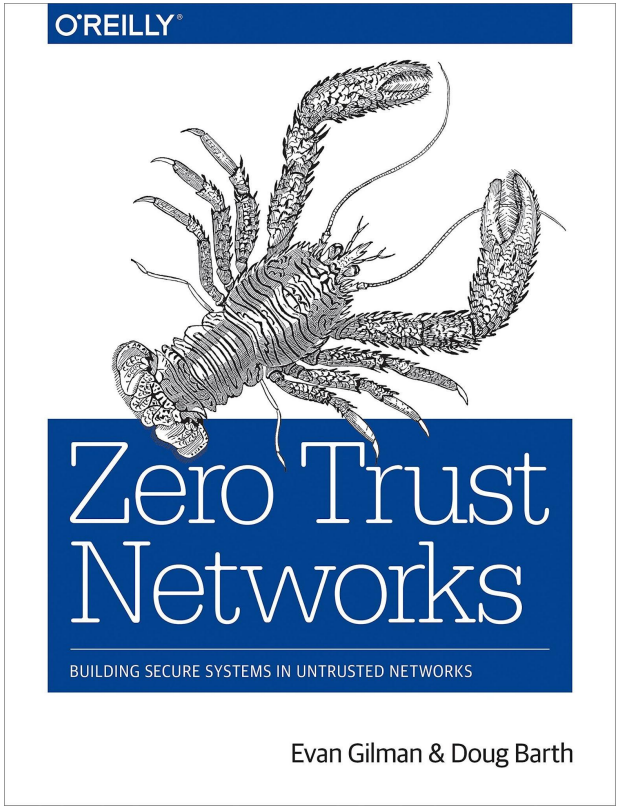
# Don't we trust our networks and firewalls?




controlplane

# BeyondCorp

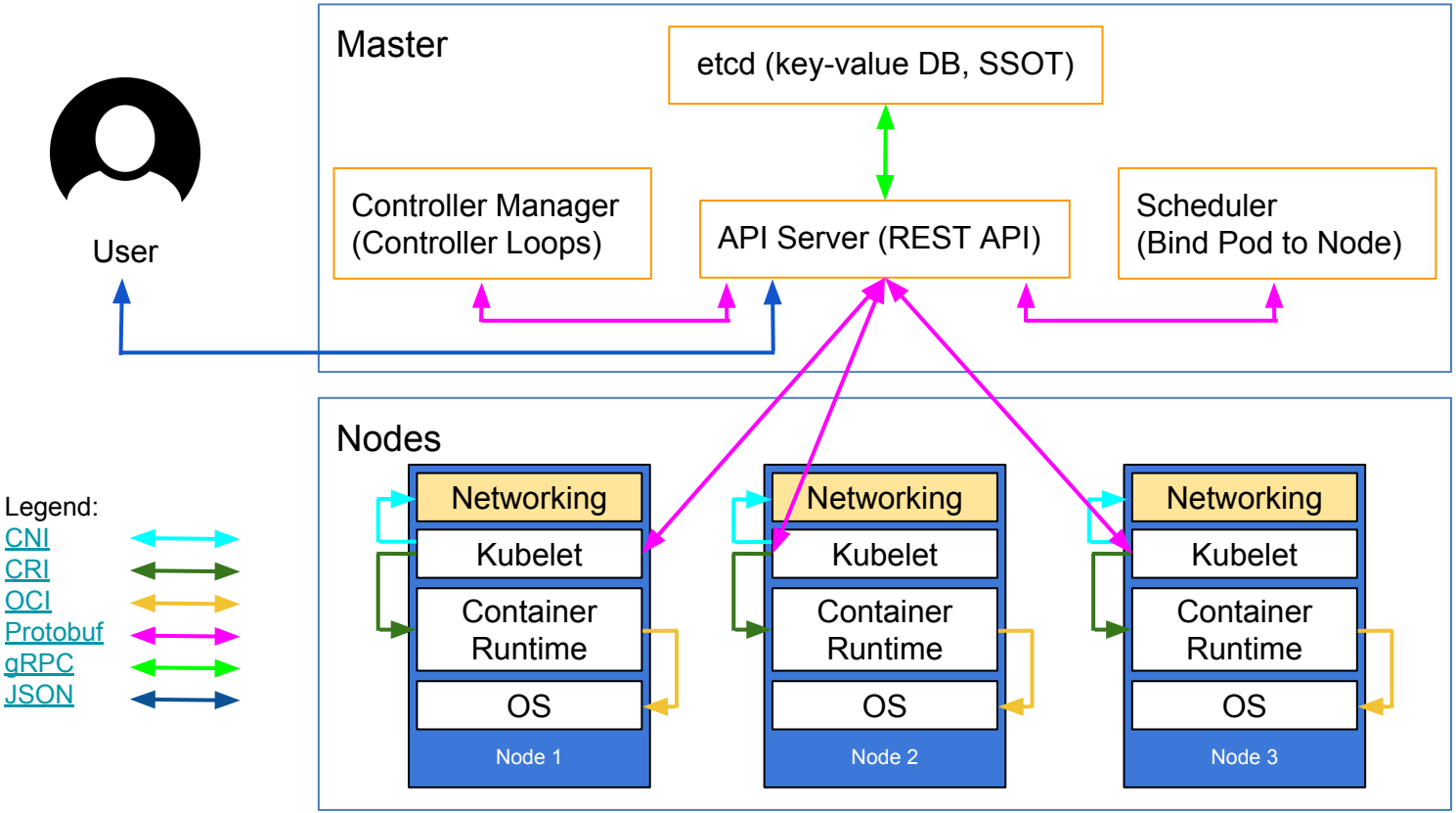
# Zero Trust Networking




controlplane

# Zero Trust API Server?

controlplane




By Lucas Käldström

controlplane

# What could possibly go wrong?

controlplane


GAME
OVER

# What could possibly go wrong?

controlplane

Game Over?

# What could possibly go wrong?

controlplane


Would you like to
continue?

# Continuous (Kubernetes) Security

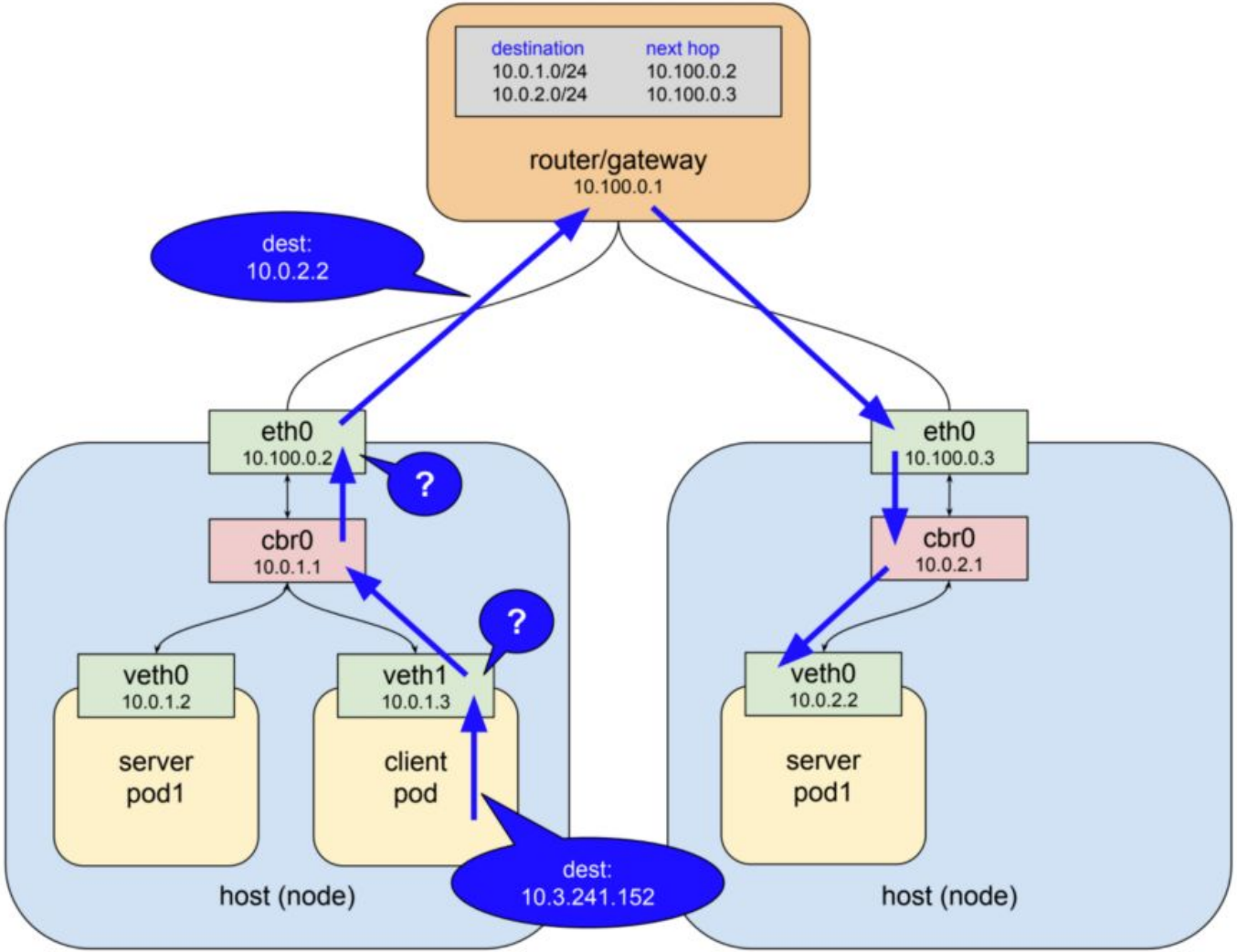

[Slides](#) / [@sublimino](#)

controlplane

# Application Layer

# Containers and Traditional Network Security?


controlplane




https://medium.com/google-cloud/understanding-kubernetes-networking-services-f0cb48e4cc82

controlplane

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny
spec:
  podSelector:
```

https://github.com/ahmetb/kubernetes-network-policy-recipes

Kubernetes NetworkPolicy: default deny

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny
spec:
  podSelector:
    - “*”
```

Illegal syntax, but represents what it actually does (effectively a wildcard)

https://github.com/ahmetb/kubernetes-network-policy-recipes

# Kubernetes NetworkPolicy: default deny

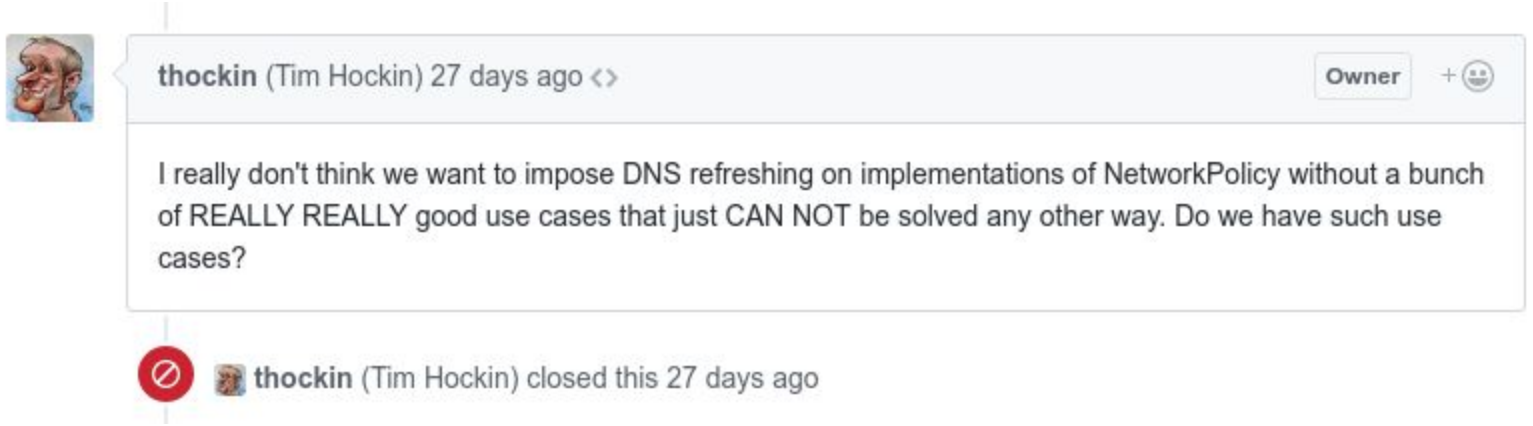
controlplane

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: foo-deny-external-egress
spec:
  podSelector:
    matchLabels:
      app: foo
  policyTypes:
  - Egress
  egress:
  - ports:
    - port: 53
      protocol: UDP
    - port: 53
      protocol: TCP
  - to:
    - namespaceSelector: {}
```

https://github.com/ahmetb/kubernetes-network-policy-recipes

Kubernetes NetworkPolicy

controlplane




https://github.com/kubernetes/kubernetes/issues/56901

# Kubernetes NetworkPolicy - NO DNS NAMES

controlplane

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: foo-deny-external-egress
spec:
  podSelector:
    dnsName: control-plane.io
  policyTypes:
  - Egress
  egress:
  - ports:
    - port: 53
      protocol: UDP
    - port: 53
      protocol: TCP
  - to:
    - namespaceSelector: {}
```

ILLEGAL! NOT ALLOWED!

https://github.com/ahmetb/kubernetes-network-policy-recipes

# Kubernetes NetworkPolicy - ILLEGAL!

controlplane

# `netassert` - cloud native network testing

- netassert - network security testing for DevSecOps workflows
  https://github.com/controlplaneio/netassert

```
host:
  localhost:
    bitbucket.com:
      - 22
  control-plane.io:
    github.com:
      - 22
```

controlplane

# `netassert` - cloud native network testing

```
k8s: # used for Kubernetes pods
  deployment: # only deployments currently supported
    test-frontend: # pod name, defaults to `default` namespace
      test-microservice: 80 # `test-microservice` is the DNS name of the target service
      test-database: -80    # should not be able to access port 80 of `test-database`

    new-namespace:test-microservice:  # `new-namespace` is the namespace name
      test-database.new-namespace: 80 # longer DNS names can be used for other namespaces
      test-frontend.default: 80

    default:test-database:
      test-frontend.default.svc.cluster.local: 80 # full DNS names can be used
      test-microservice.default.svc.cluster.local: -80
      control-plane.io: 443 # we can check remote services too
```

https://github.com/controlplaneio/netassert

controlplane

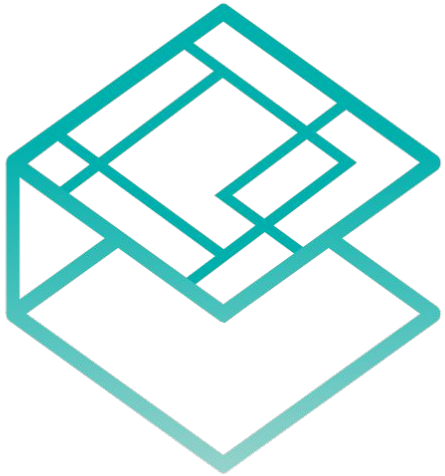
```
[2018-02-02T16:06:49.124+0000] ./netassert: Results: localhost
TAP version 13
# localhost TCP:30731 closed
ok 1 - localhost TCP:30731 closed
# localhost UDP:1234 closed
ok 2 - localhost UDP:1234 closed
# localhost TCP:22 open
ok 3 - localhost TCP:22 open
# binarysludge.com TCP:443 open
ok 4 - binarysludge.com TCP:443 open
# localhost TCP:999 closed
ok 5 - localhost TCP:999 closed
# control-plane.io TCP:443 open
ok 6 - control-plane.io TCP:443 open
# localhost UDP:555 closed
ok 7 - localhost UDP:555 closed
# control-plane.io TCP:80 open
ok 8 - control-plane.io TCP:80 open
# binarysludge.com TCP:22 open
ok 9 - binarysludge.com TCP:22 open
# binarysludge.com TCP:80 open
ok 10 - binarysludge.com TCP:80 open
# 8.8.8.8 UDP:53 open
ok 11 - 8.8.8.8 UDP:53 open
# google.co.uk TCP:443 open
ok 12 - google.co.uk TCP:443 open
# binarysludge.com TCP:81 open
ok 13 - binarysludge.com TCP:81 open
# 8.8.4.4 UDP:53 open
ok 14 - 8.8.4.4 UDP:53 open

1..14
# tests 14
# pass 14
# fail 0

[2018-02-02T16:06:49.129+0000] ./netassert: localhost pass
```

controlplane

# Cloud Native Dynamic Firewalls

- Network Policy recipes - https://github.com/ahmetb/kubernetes-network-policy-recipes
- WeaveNet Network Policy - https://kubernetes.io/docs/tasks/administer-cluster/weave-network-policy/
- NeuVector Container Firewall - https://neuvector.com/products/
- Tesla Compromise mitigation - https://www.tigera.io/tesla-compromise-network-policy/

# Applications: CNI and Network Policy



CNI

controlplane

# Applications: CNI and Network Policy

| Provider | Network Model | Route Distribution | Network Policies | Mesh | External Datastore | Encryption | Ingress/Egress Policies | Commercial Support |
|---|---|---|---|---|---|---|---|---|
| Calico | Layer 3 | Yes | Yes | Yes | Etcd [1] | Yes | Yes | Yes |
| Canal | Layer 2 vxlan | N/A | Yes | No | Etcd [1] | No | Yes | No |
| flannel | vxlan | No | No | No | None | No | No | No |
| kopeio-networking | Layer 2 vxlan [2] | N/A | No | No | None | Yes [3] | No | No |
| kube-router | Layer 3 | BGP | Yes | No | No | No | No | No |
| romana | Layer 3 | OSPF | Yes | No | Etcd | No | Yes | Yes |
| Weave Net | Layer 2 vxlan [4] | N/A | Yes | Yes | No | Yes | Yes [5] | Yes |

[Choosing a CNI Provider]

controlplane

# Bootstrapping identity with SPIFFE

# Attestation Example: Kubernetes

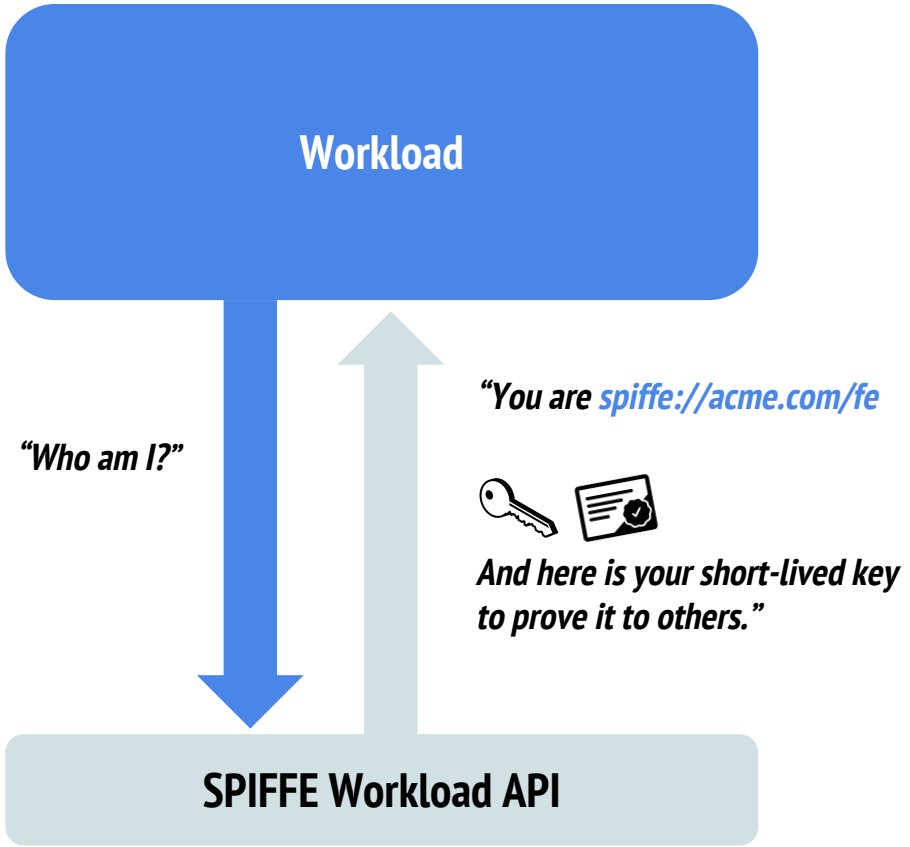
```
/proc/[pid]/cgroup
```

spiffe

SCYTALE

TiGERA

heptio




controlplane

# SPIFFE ID

spiffe://acme.com/billing/payments

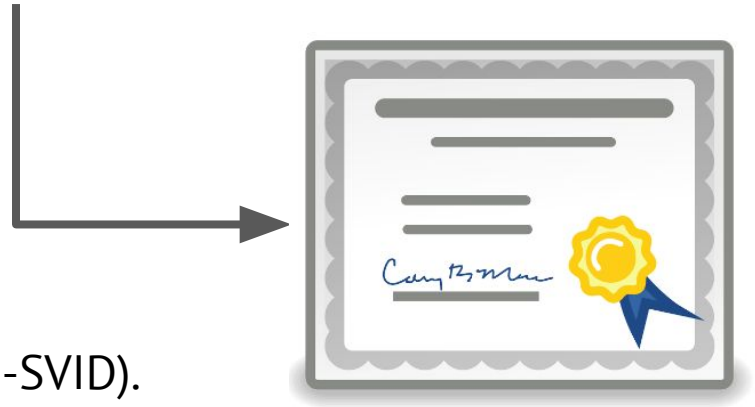
*Trust Domain* *Workload Identifier*

controlplane

# SPIFFE Verifiable Identity Document

spiffe://acme.com/billing/payments

Typically short-lived



Today only one form of SVID (X509-SVID).
Other document types under consideration
(including JWT-SVID)

controlplane

# X.509 RFC Format

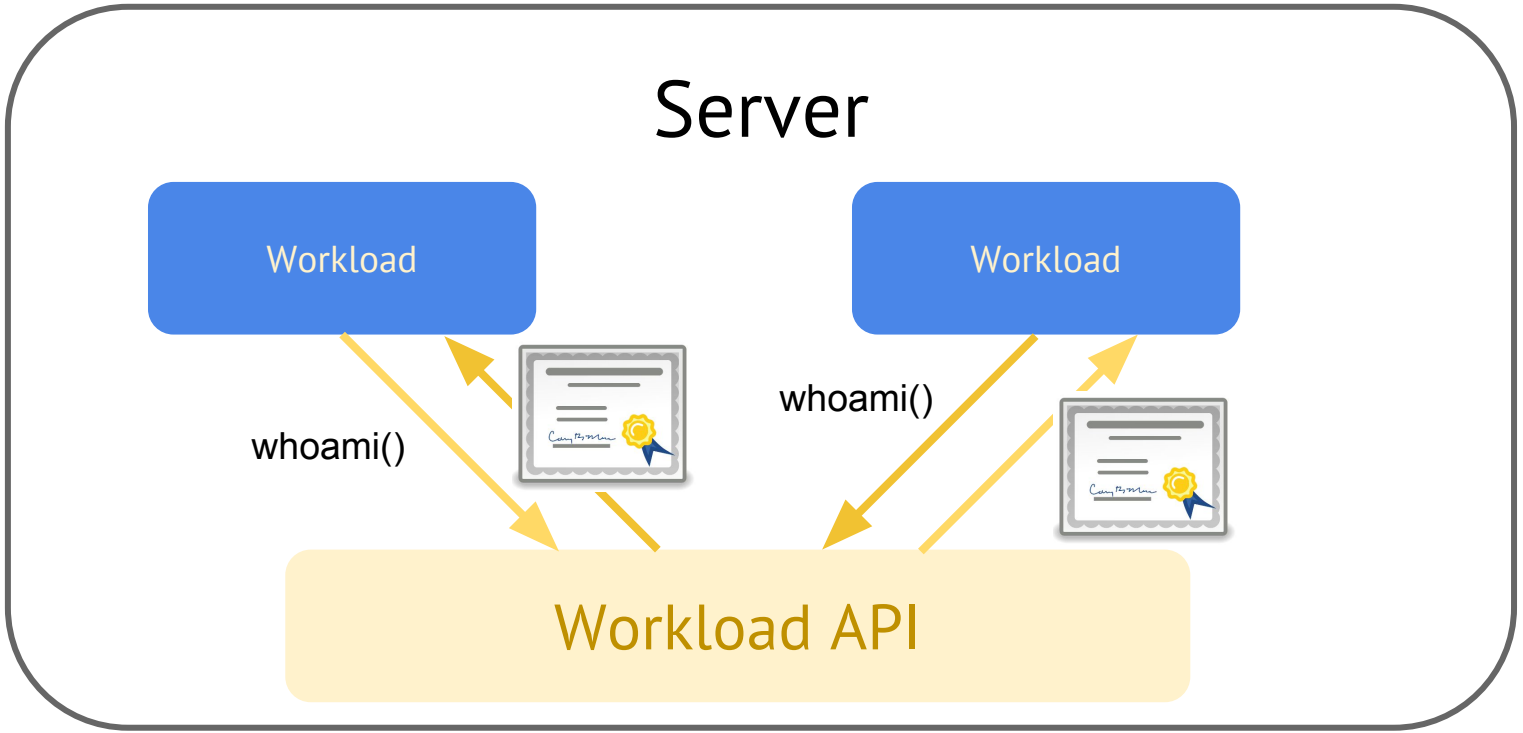
```
Certificate  ::=  SEQUENCE  {
     tbsCertificate       TBSCertificate,
     signatureAlgorithm   AlgorithmIdentifier,
     signatureValue       BIT STRING  }

TBSCertificate  ::=  SEQUENCE  {
     version         [0]  EXPLICIT Version DEFAULT v1,
     serialNumber         CertificateSerialNumber,
     signature            AlgorithmIdentifier,
     issuer               Name,
     validity             Validity,
     subject              Name,
     subjectPublicKeyInfo SubjectPublicKeyInfo,
     issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version MUST be v2 or v3
     subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version MUST be v2 or v3
     extensions      [3]  EXPLICIT Extensions OPTIONAL
                          -- If present, version MUST be v3
     }

Version  ::=  INTEGER  {  v1(0), v2(1), v3(2)  }

CertificateSerialNumber  ::=  INTEGER
```

```
Validity ::= SEQUENCE {
     notBefore      Time,
     notAfter       Time }

Time ::= CHOICE {
     utcTime        UTCTime,
     generalTime    GeneralizedTime }

UniqueIdentifier  ::=  BIT STRING

SubjectPublicKeyInfo  ::=  SEQUENCE  {
     algorithm            AlgorithmIdentifier,
     subjectPublicKey     BIT STRING  }

Extensions  ::=  SEQUENCE SIZE (1..MAX) OF Extension

Extension  ::=  SEQUENCE  {
     extnID      OBJECT IDENTIFIER,
     critical    BOOLEAN DEFAULT FALSE,
     extnValue   OCTET STRING
                 -- contains the DER encoding of an ASN.1 value
                 -- corresponding to the extension type identified
                 -- by extnID
     }
```

https://github.com/spiffe/spiffe/blob/master/standards/X509-SVID.md#appendix-a-x509-field-reference

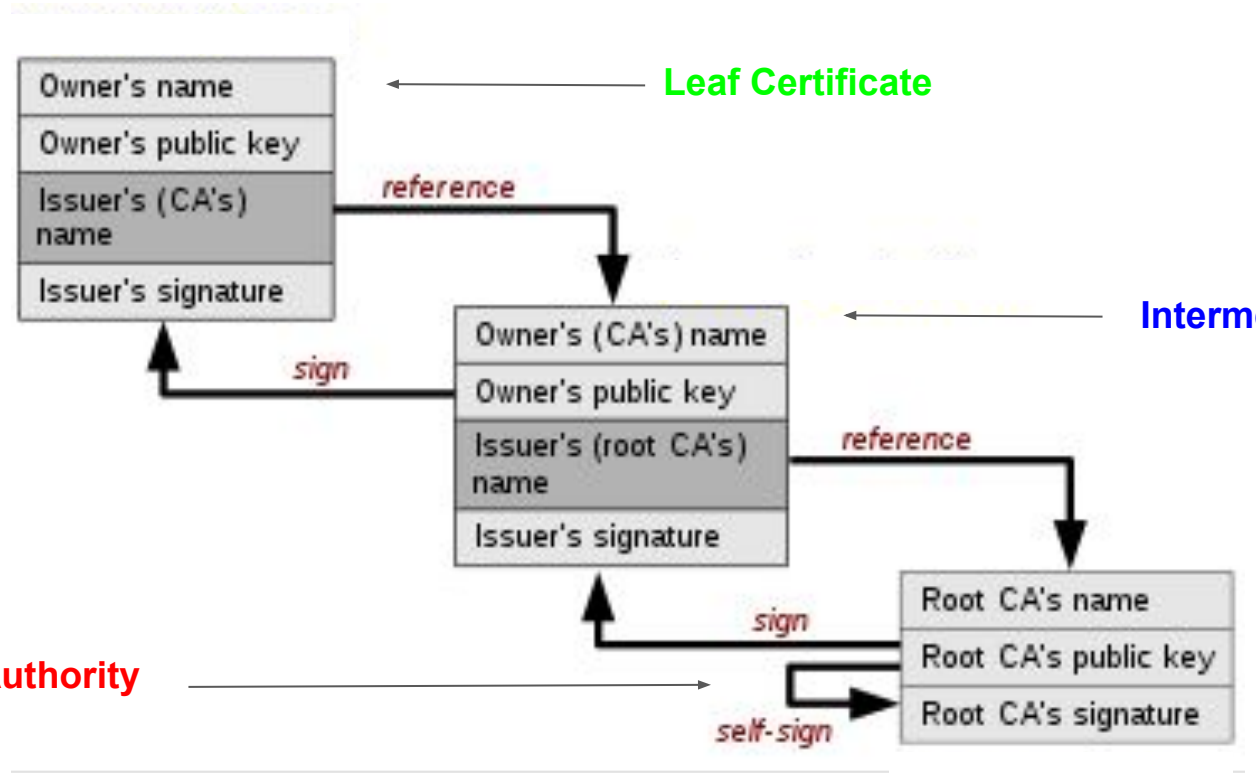
controlplane

# SPIFFE Workload API



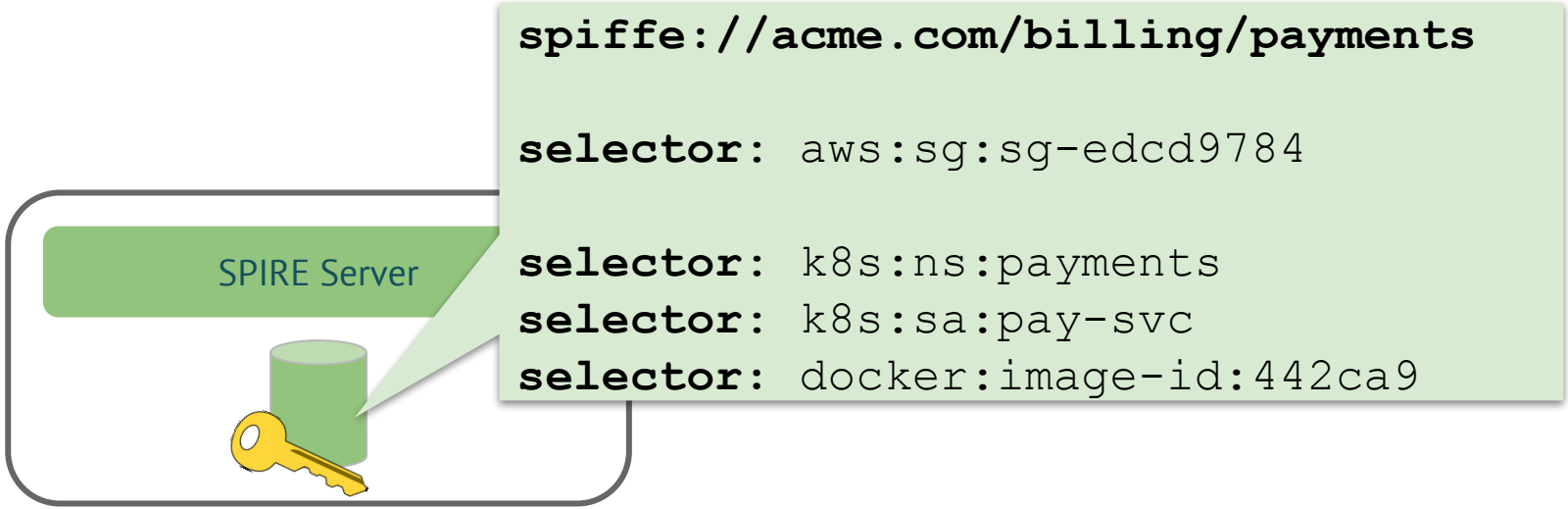

controlplane

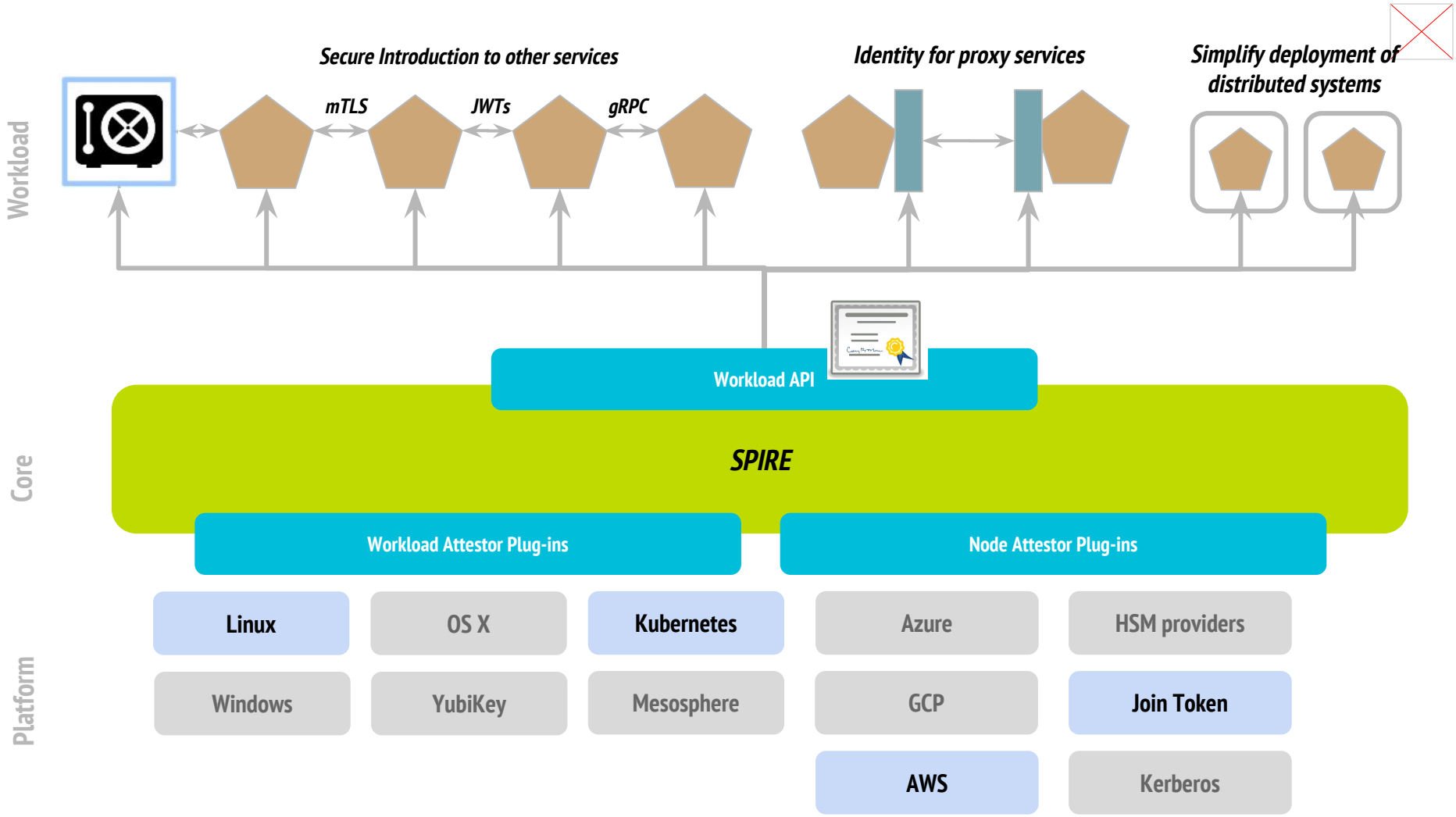
# Certificate Path Validation




controlplane

# SPIFFE Runtime Environment

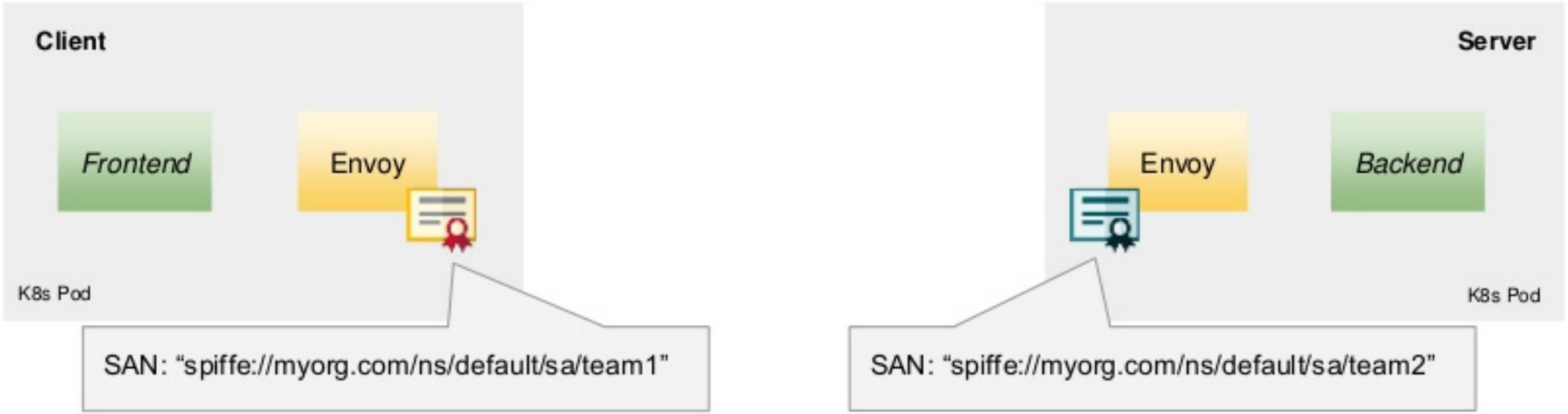



Secure Introduction to other services
Identity for proxy services
Simplify deployment of distributed systems
mTLS
JWTs
gRPC
Workload
Workload API
SPIRE
Core
Workload Attestor Plug-ins
Node Attestor Plug-ins
Linux
OS X
Kubernetes
Azure
HSM providers
Windows
YubiKey
Mesosphere
GCP
Join Token
AWS
Kerberos
Platform

# What SPIFFE is *not*

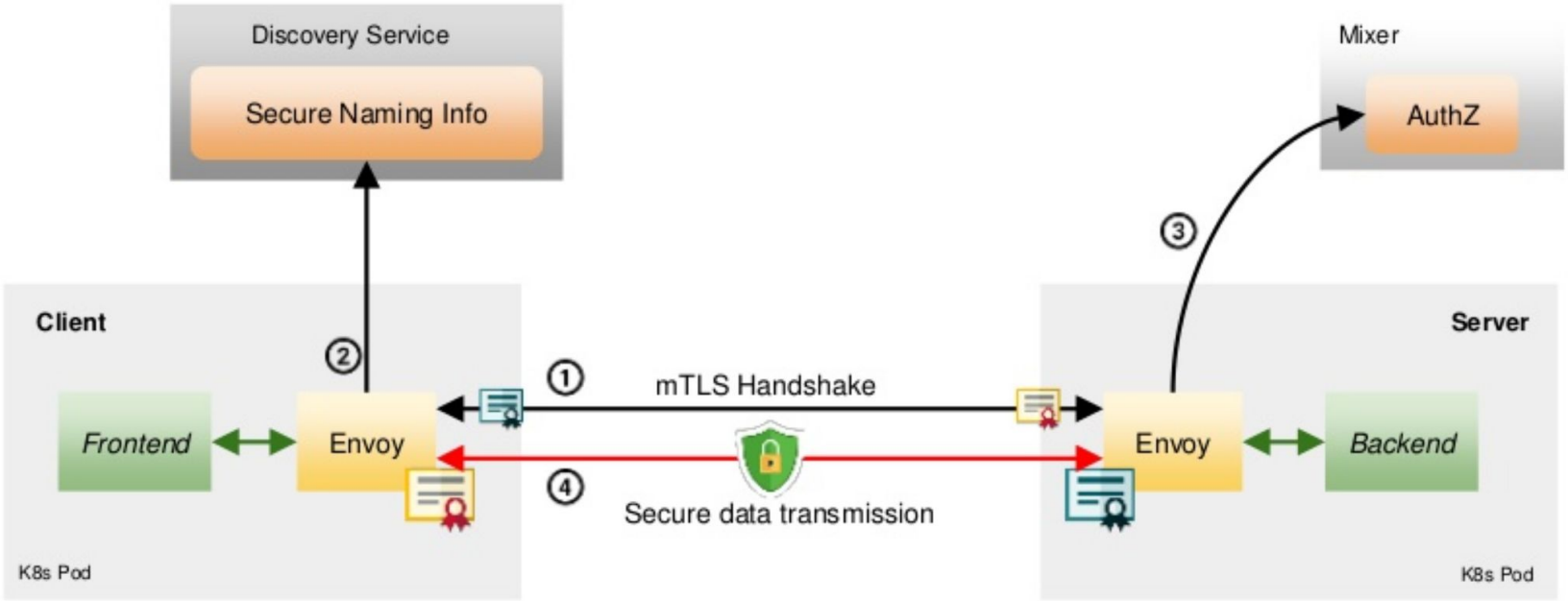
- **Authorization** (however it provides identities upon which authorization schemes can be deployed)
- **Transport level security** (however SVIDs can be used to facilitate things like TLS or JWT signing)

# Using SPIFFE in TLS Certificates




https://www.slideshare.net/MattBaldwin3/istio-cloud-native-online-series-intro-to-istio-security

# Istio and SPIFFE




https://www.slideshare.net/MattBaldwin3/istio-cloud-native-online-series-intro-to-istio-security

# Recap


controlplane

# End to End Encryption

- TLS on API Server Components
- SPIFFE to identify application workloads
- Istio CA to issue TLS certificates to application workloads
- Envoy to proxy application’s HTTPS traffic across the Istio service mesh

# Takeaway: Encrypt Everything Everywhere

- Encrypt

# Takeaway: Encrypt Everything Everywhere

- Encrypt
- Encrypt Everything

# Takeaway: Encrypt Everything Everywhere

- Encrypt
- Encrypt Everything
- Encrypt Everything Everywhere

# Conclusion

- Network Security is important
- TLS, X.509, and Network Policies keep us safe
- Cloud Native applications have more security primitives than ever before
- Istio and SPIFFE give you wings
- Encrypt Everything Everywhere