Friends Don't Let Friends Hard-Code Their Secrets

"It's turtles all the way down!"

GopherCon TR

December 2021

Volkan Özçelik

@linkibol

Hi 👋, I'm Volkan.

— *Technical Lead at Cisco*

Social

- twitter.com/**linkibol**
- twitch.tv/**ZeroToHeroDev**
- linkedin.com/in/**volkanozcelik**

GitHub

- github.com/**v0lkan**
- github.com/**zerotohero-dev**

Web

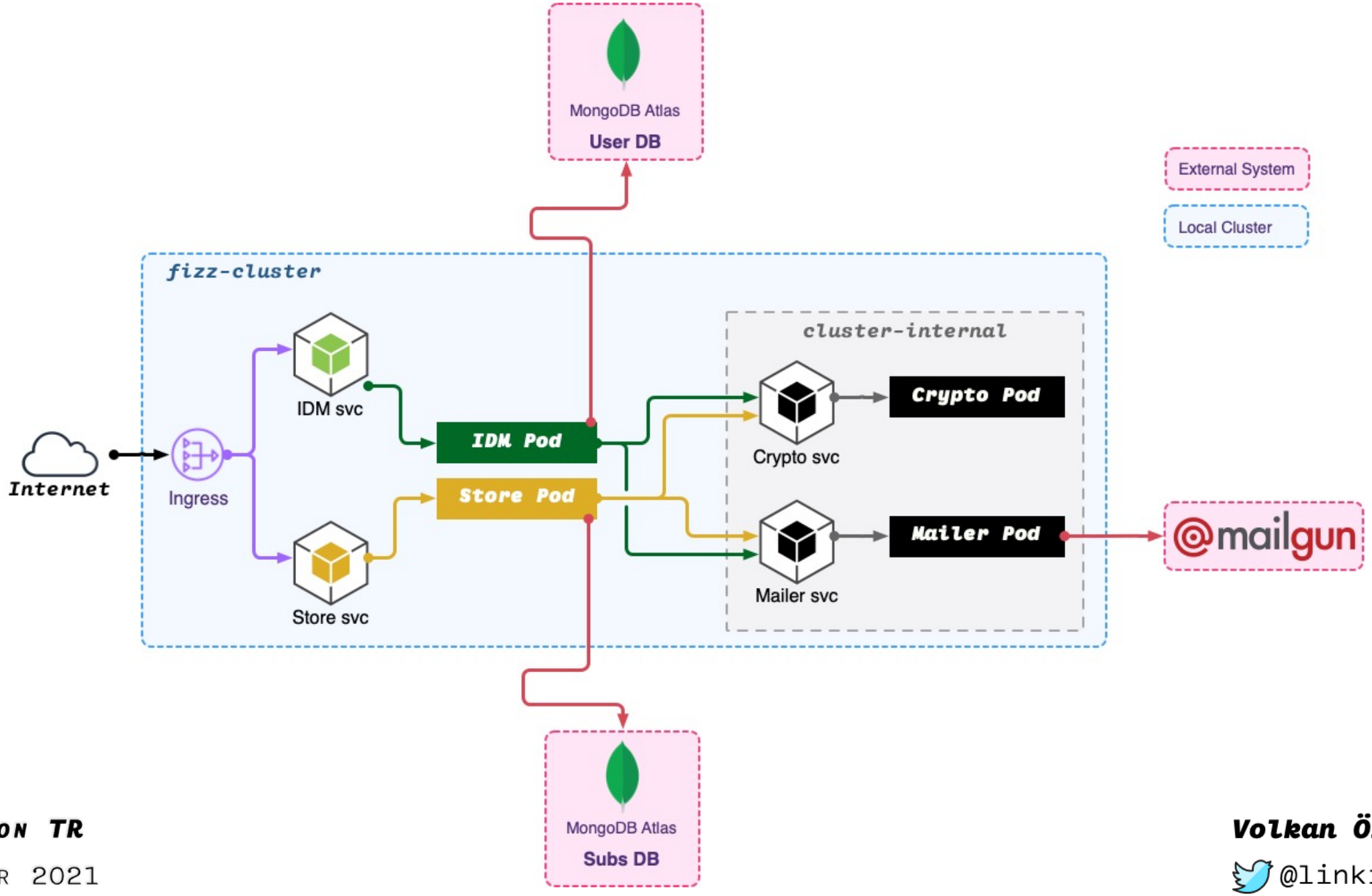
- **volkan.io**
- zerotohero.dev
- fizzbuzz.pro

Resources

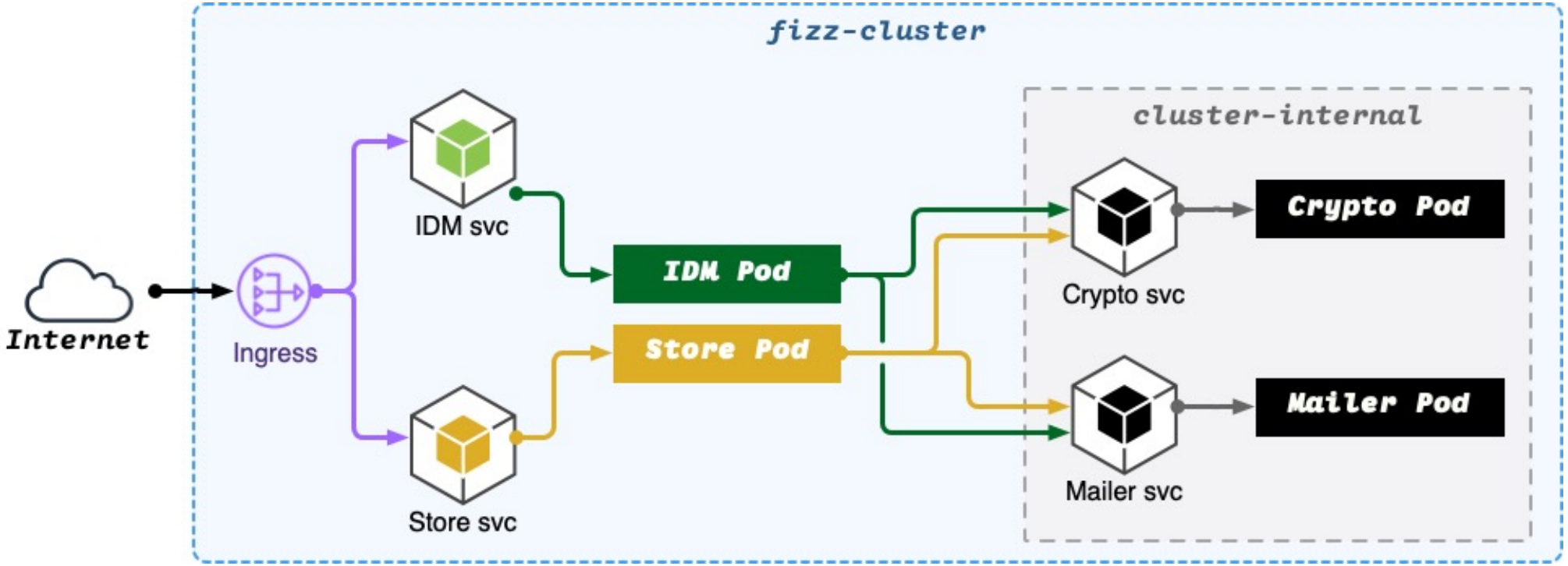
- https://zerotohero.dev/**spire-rocks**
  - ☝️ slides, source code, …etc.

Agenda

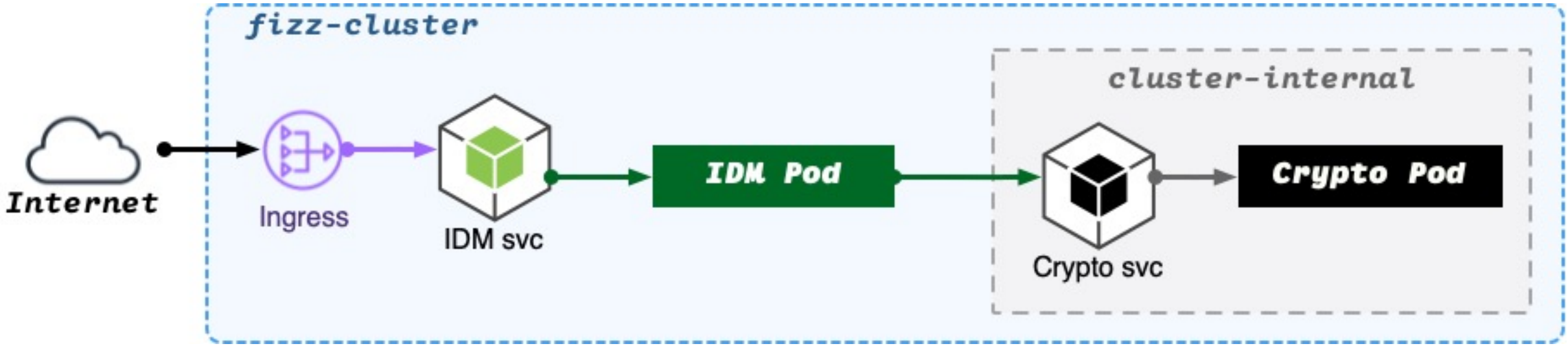
- High-Level System Overview
- What Is A Service Identity
- Challenges With Identity at Scale
- The Shift of Trust
- Introducing SPIFFE and SPIRE
- Demo
- Q&A



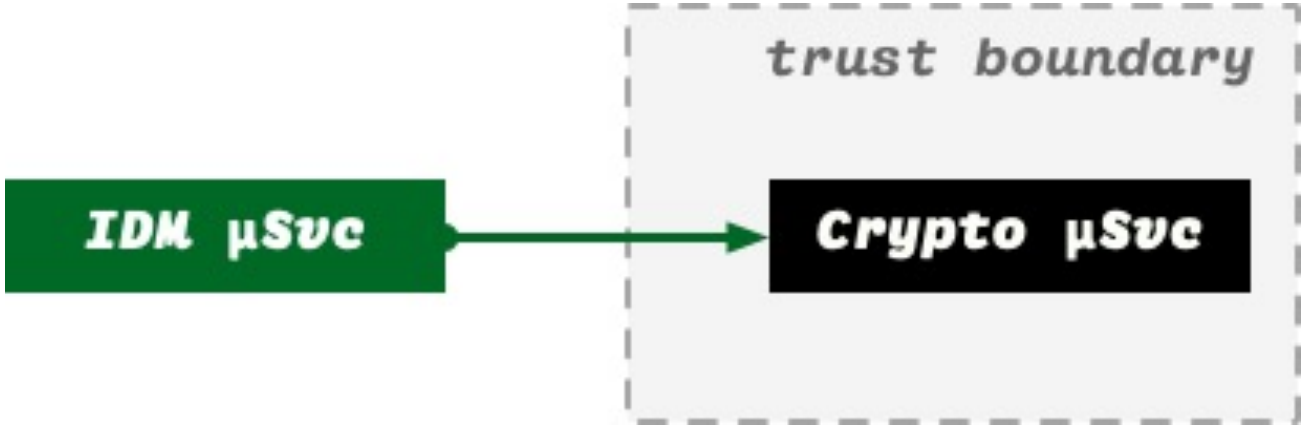
Volkan Özçelik
@linkibol

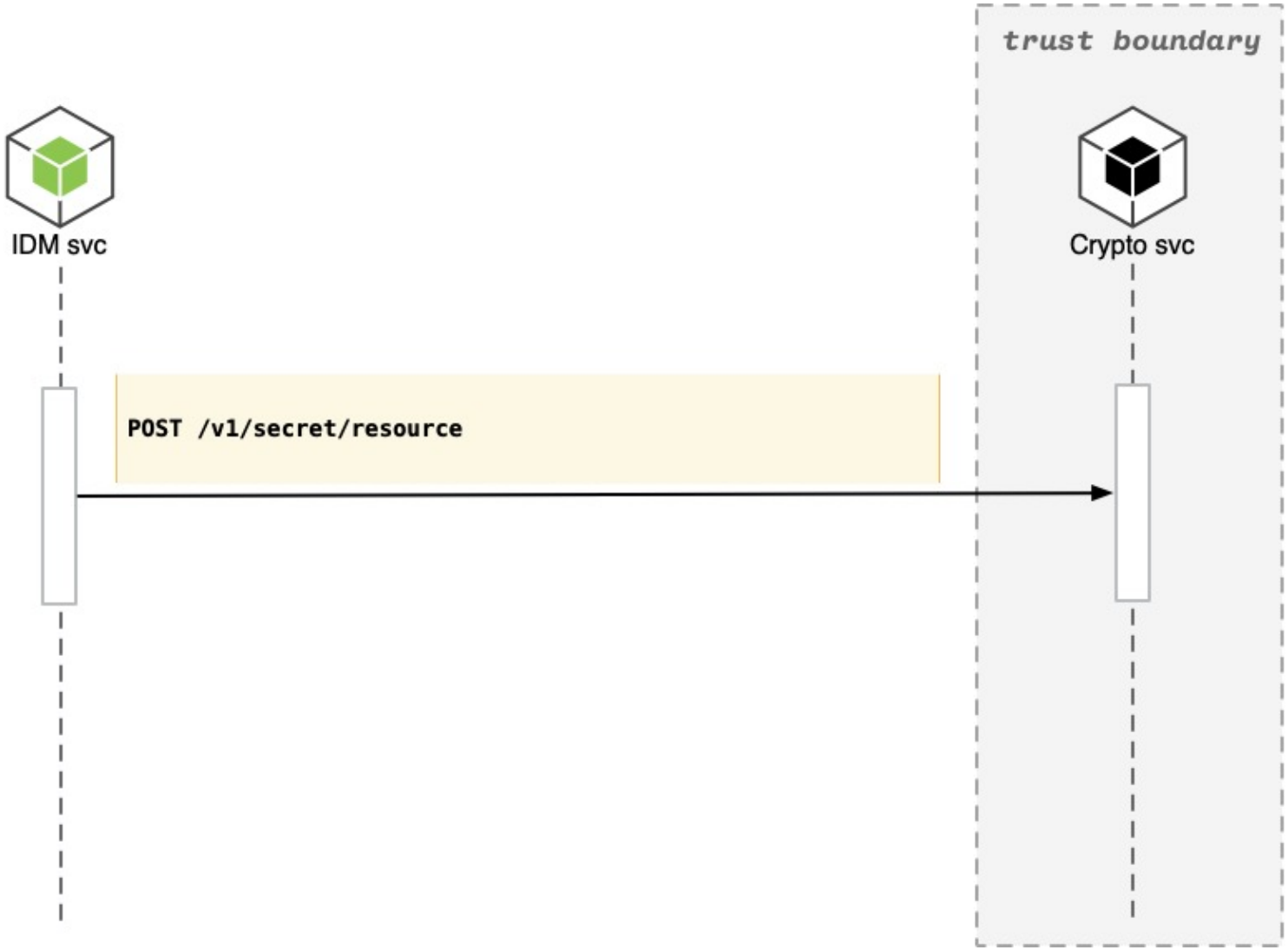


Volkan Özçelik
@linkibol



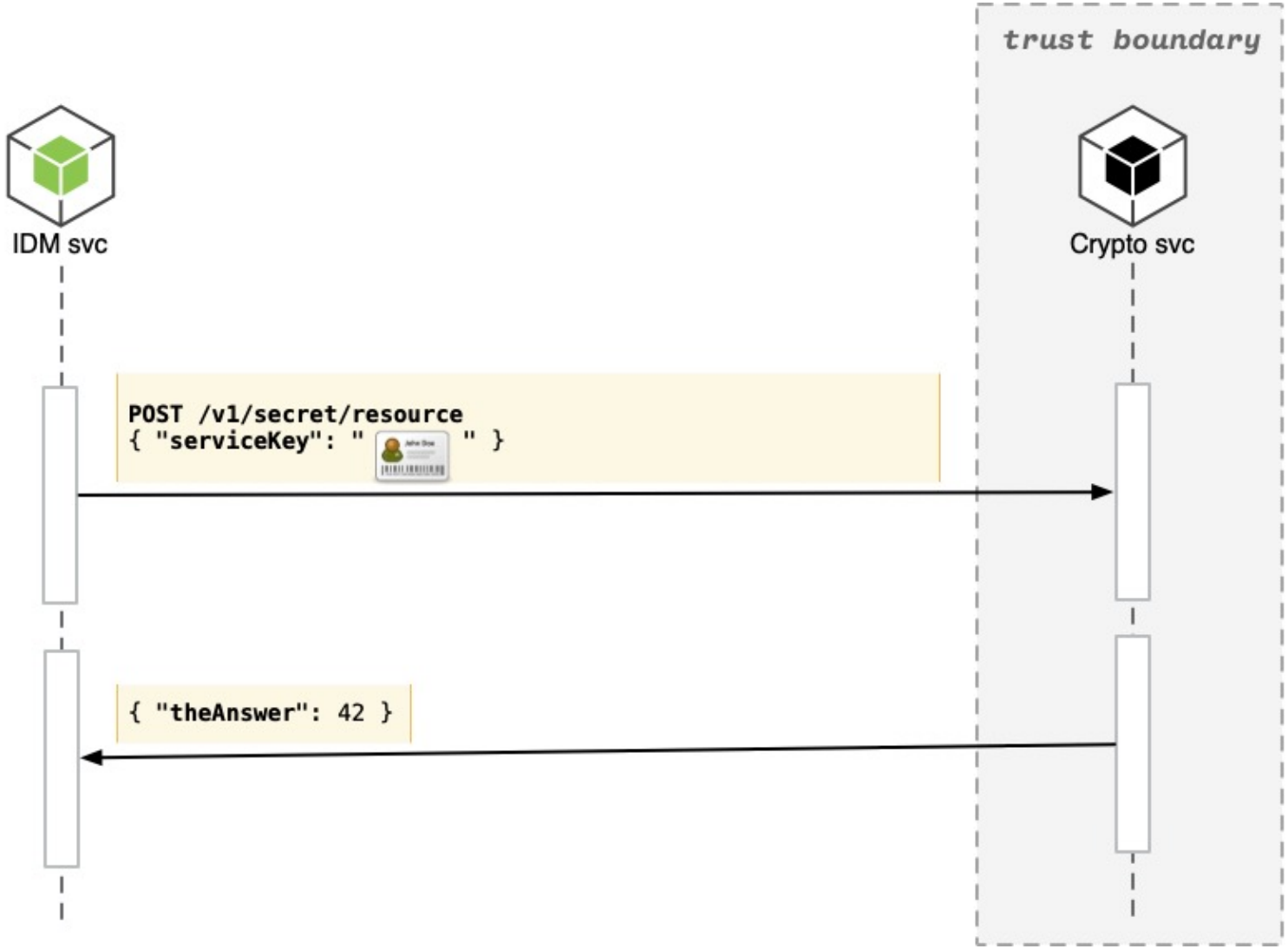
Volkan Özçelik
@linkibol


trust boundary
IDM µSvc
Crypto µSvc

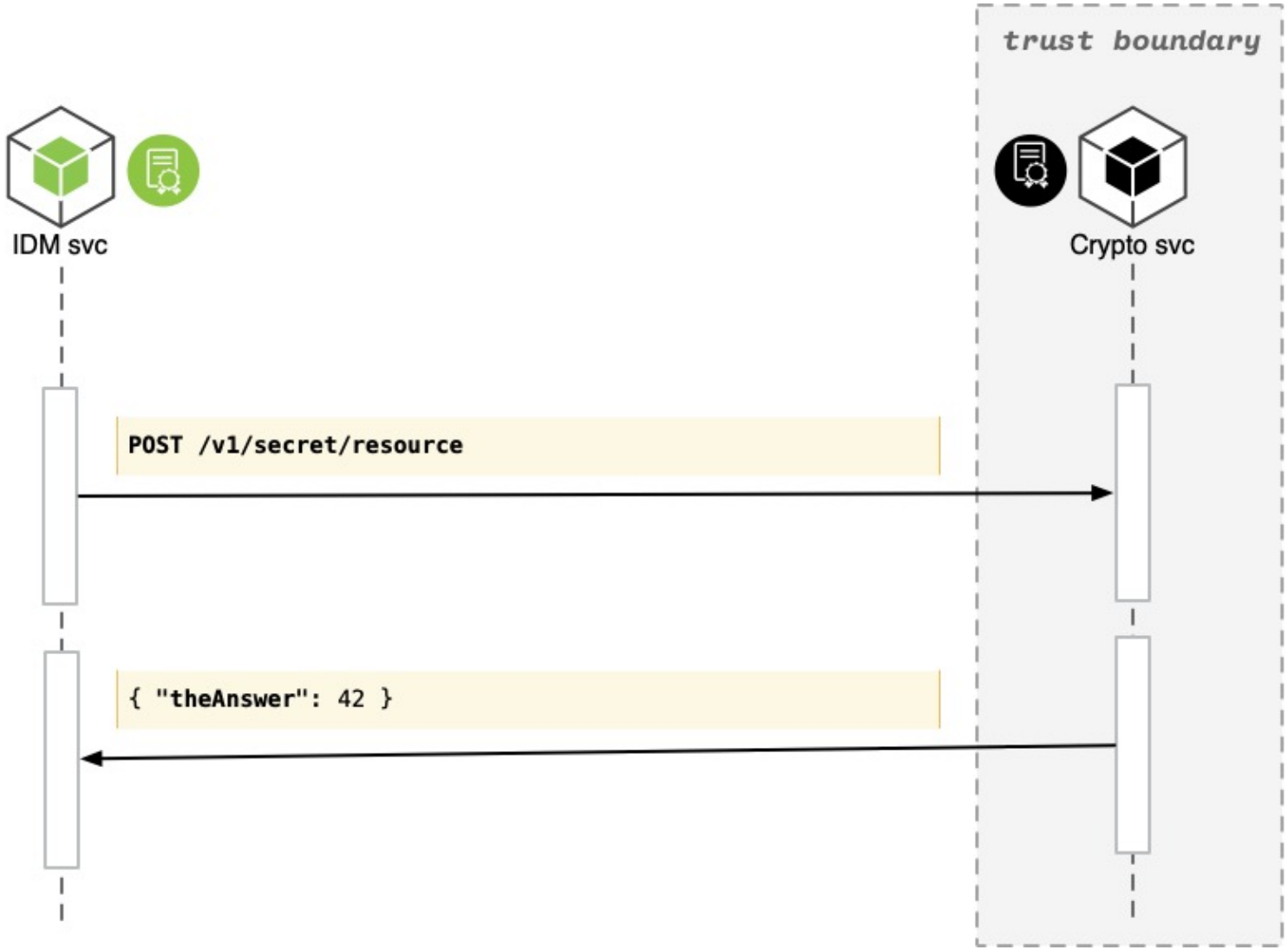

trust boundary
IDM svc
Crypto svc
POST /v1/secret/resource


trust boundary
IDM svc
Crypto svc
POST /v1/secret/resource
{ "serviceKey": "verySecureKeyToBeKeptAbsolutelySecret" }

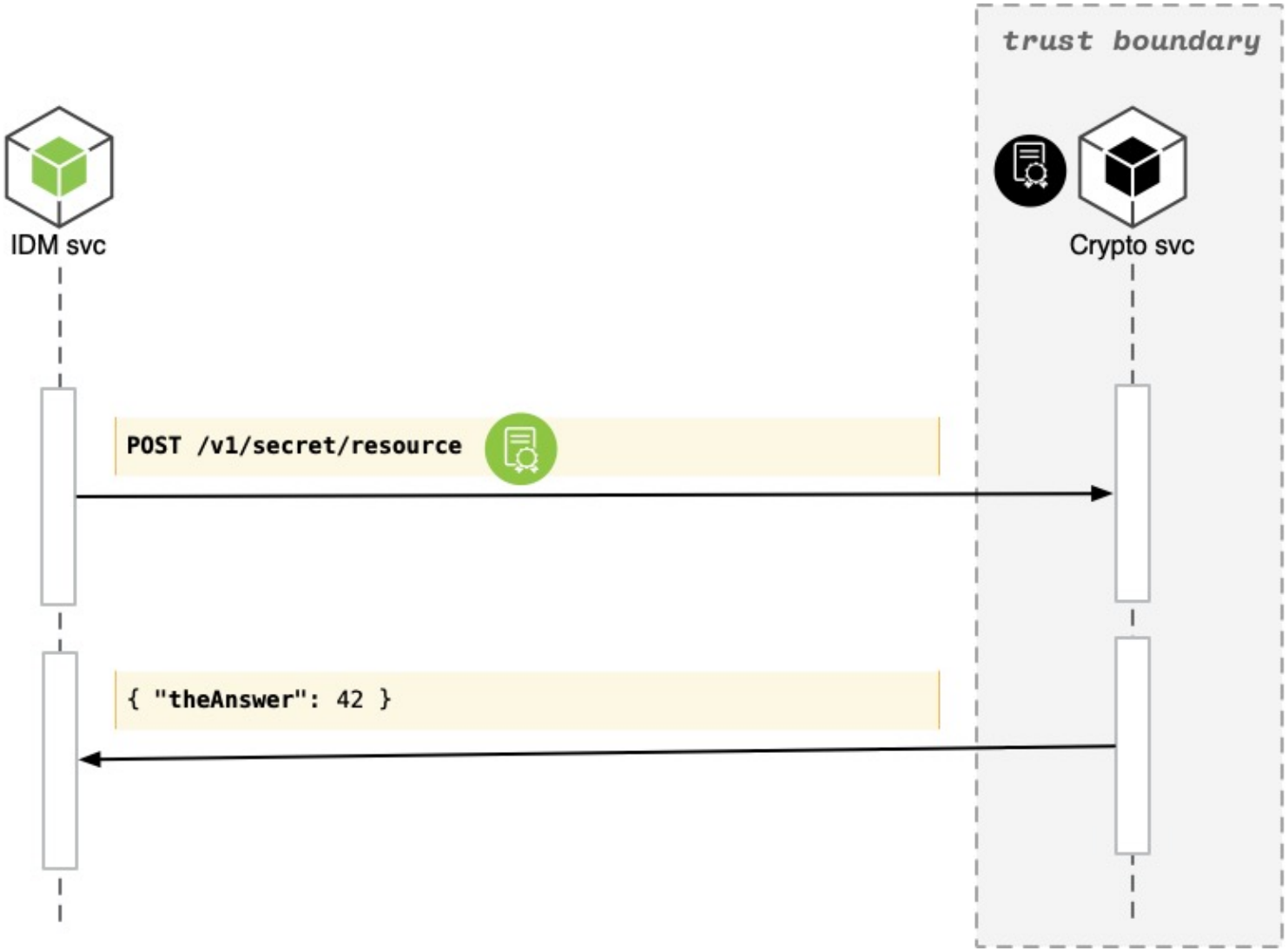
Volkan Özçelik
@linkibol

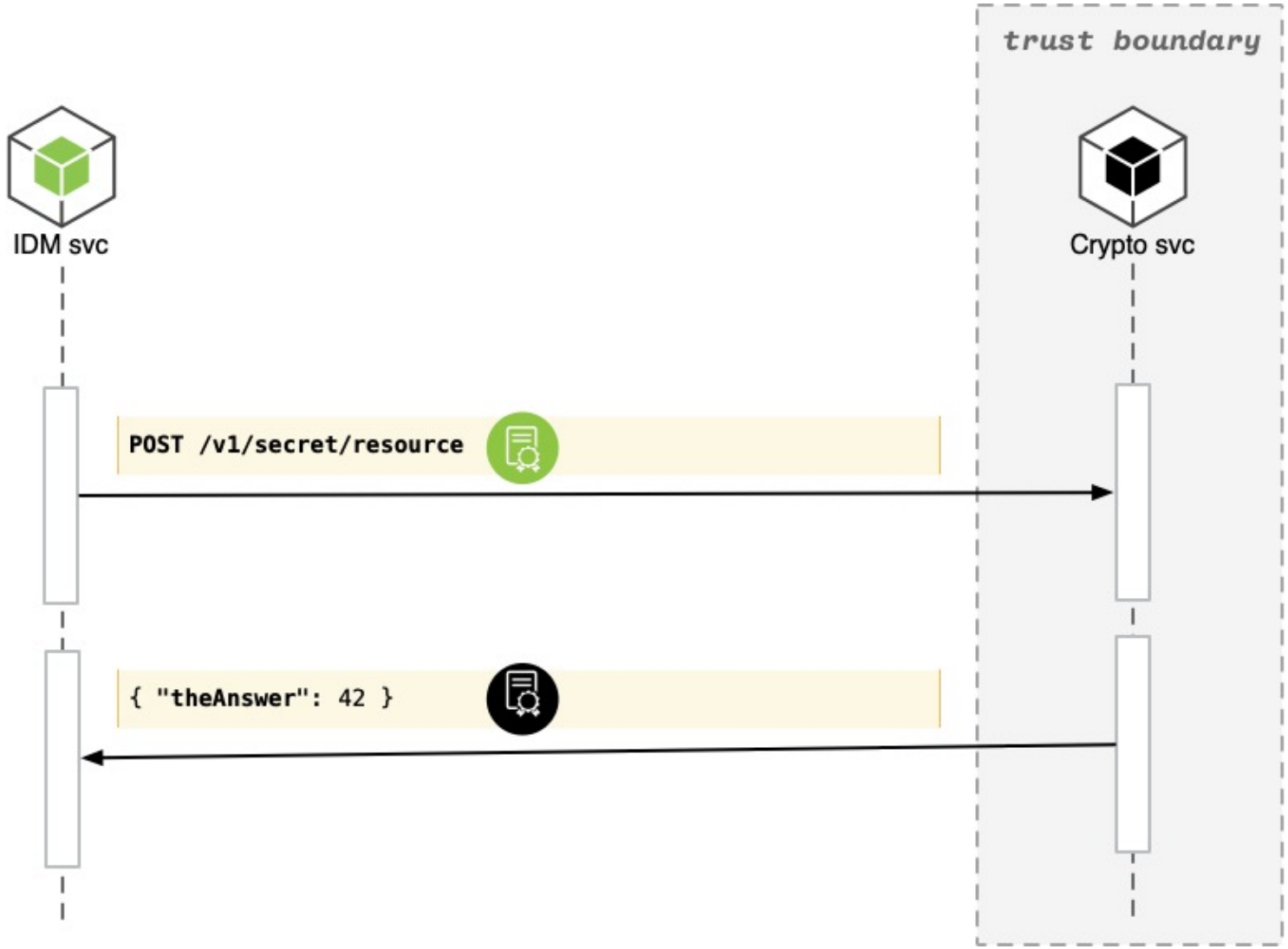


Volkan Özçelik
@linkibol

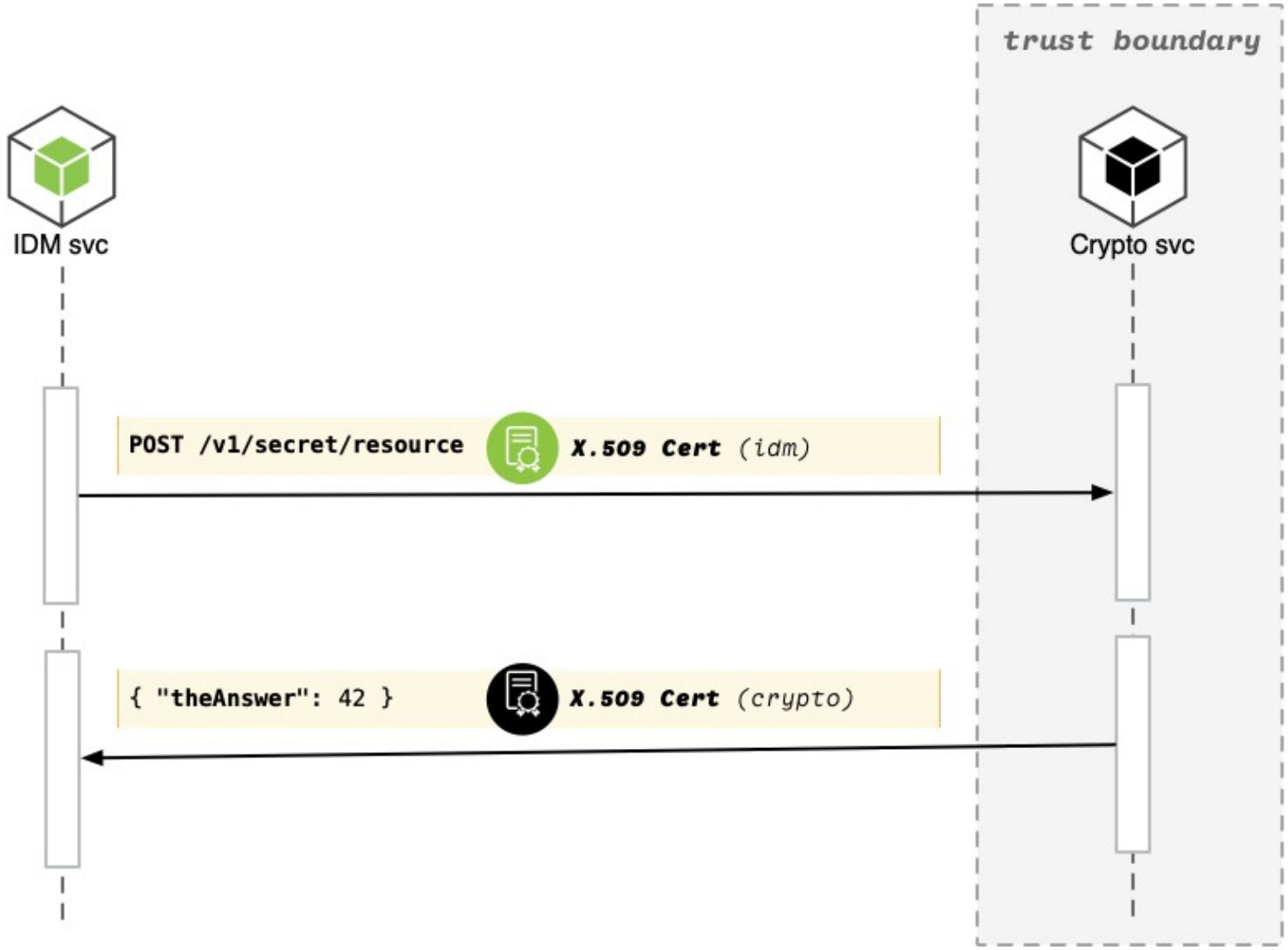


Volkan Özçelik
@linkibol

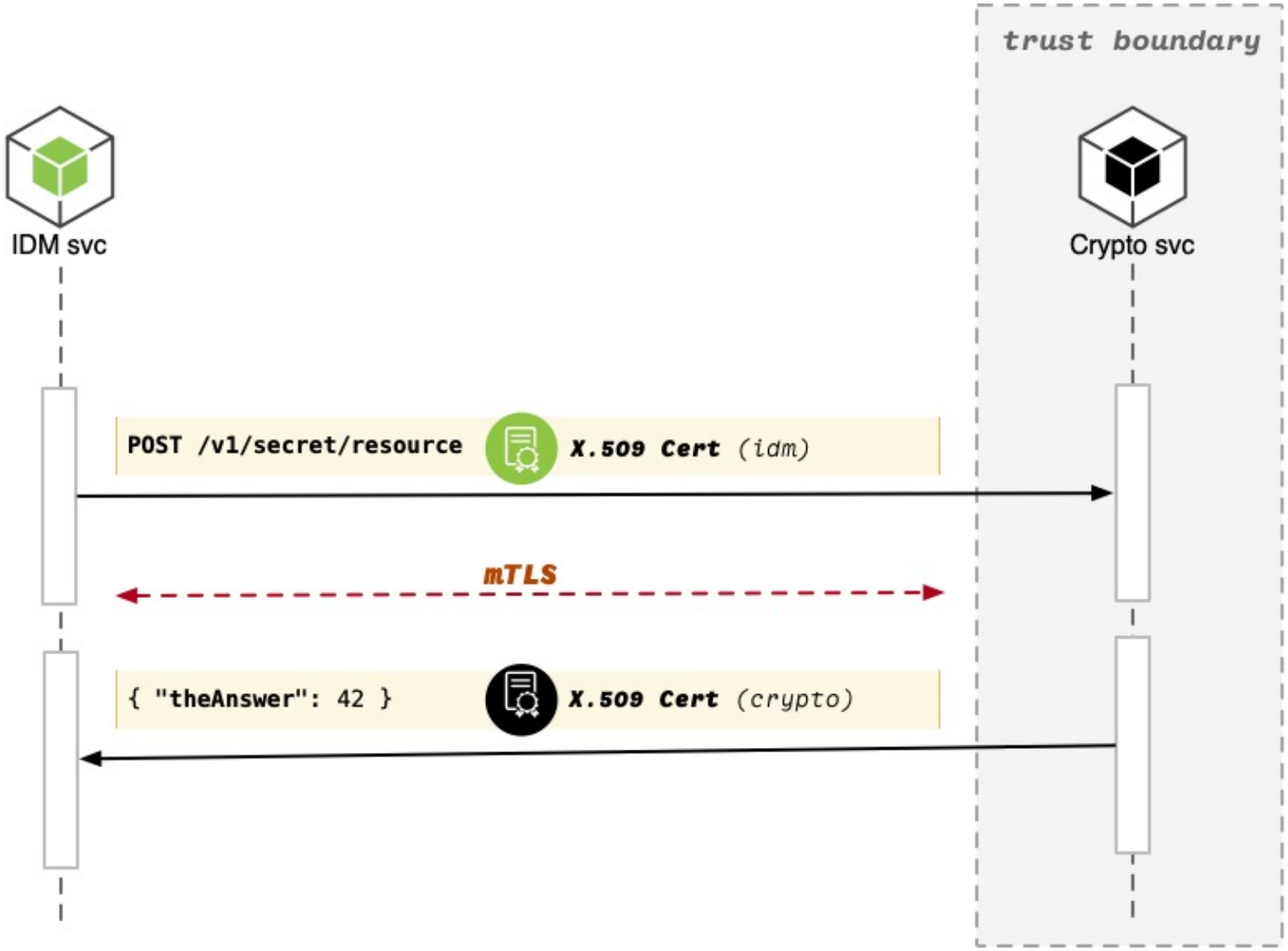

trust boundary
IDM svc
Crypto svc
POST /v1/secret/resource
{ "theAnswer": 42 }



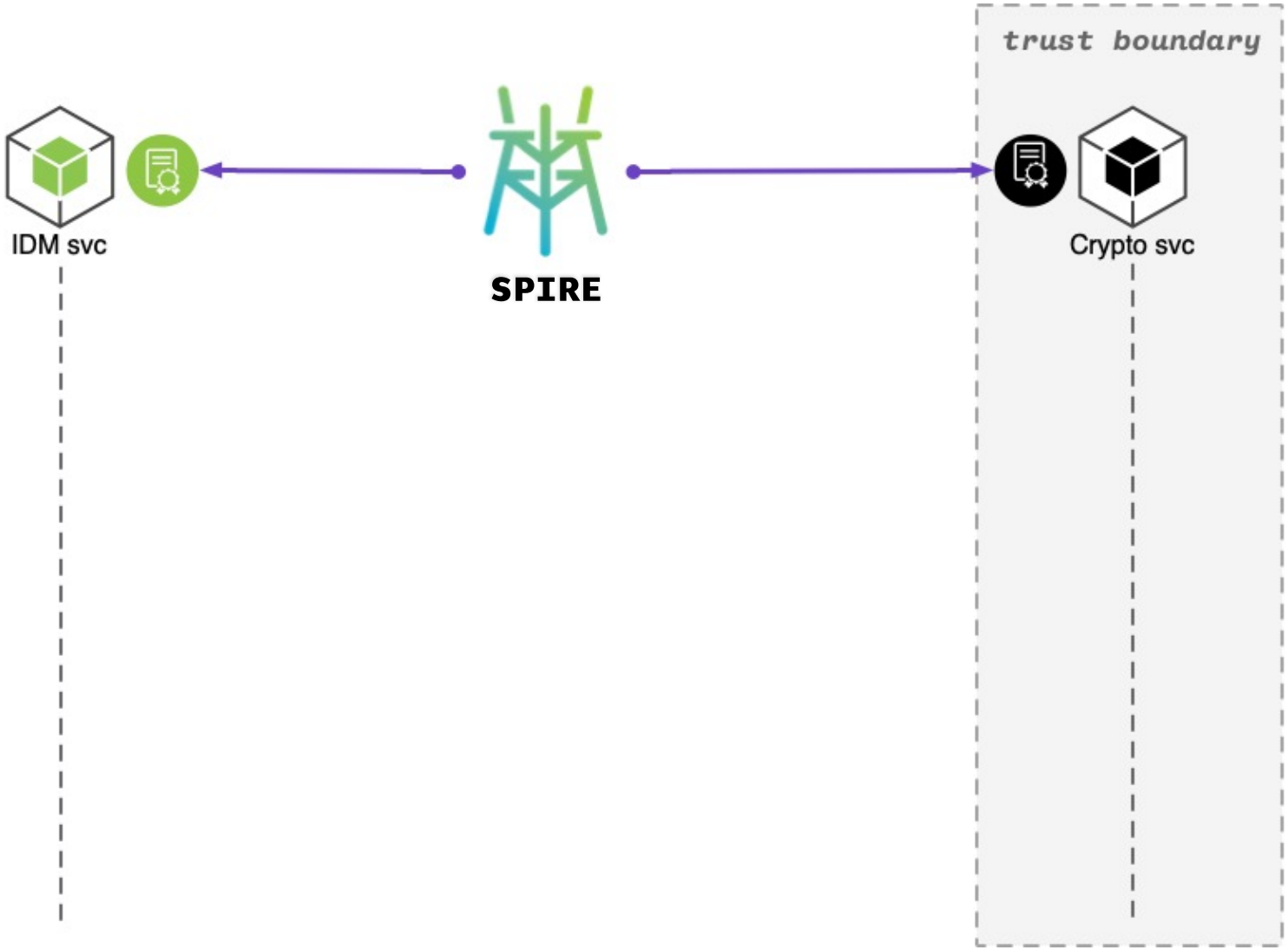
Volkan Özçelik
@linkibol



Volkan Özçelik
@linkibol



Volkan Özçelik
@linkibol



Volkan Özçelik
@linkibol

What Is a Service Identity

- Proof that the service is what it claims to be.
- Identity without authorization is “*public access*”.

Service Identity

- Service Identity Configuration
- Service Identity Operation

Volkan Özçelik
@linkibol

Service Identity Configuration

- Registration of the service identity
- Provisioning of credential(s)
- Access authorization

Volkan Özçelik
@linkibol

Service Identity Operation

- Presenting the service identity
- Authentication by credentials
- Access control

Identity at Scale

- Identity management should be uniform across workloads.
- Identity depending on hosts, or network topology is hard to maintain and scale.

Identity at Scale

- Interoperability
- Non-repudiation
- Compliance
- Auditability

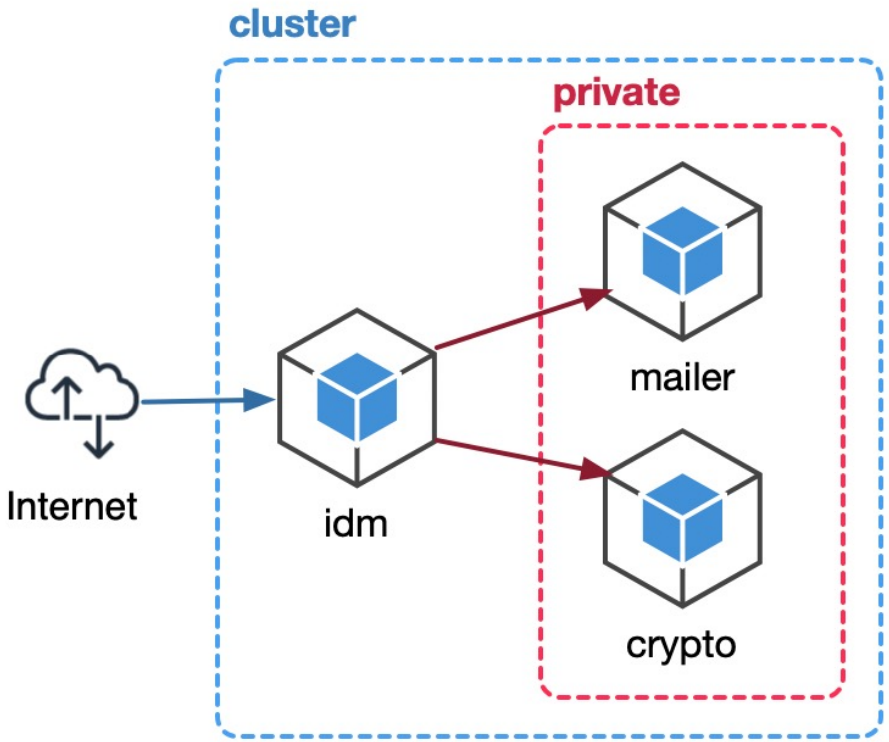
Volkan Özçelik
@linkibol

Identity at Scale

- Rotating Identities
- Revoking Identities
- Distributing Identities
- Compartmentalization

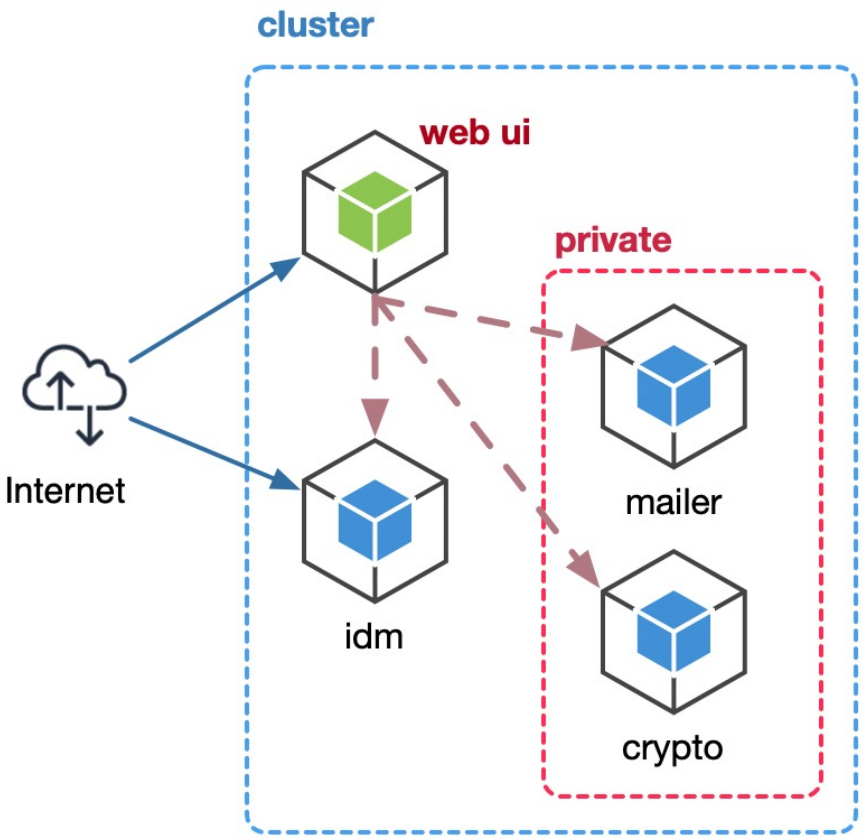
Volkan Özçelik
@linkibol

“Hope” is not a Security Posture



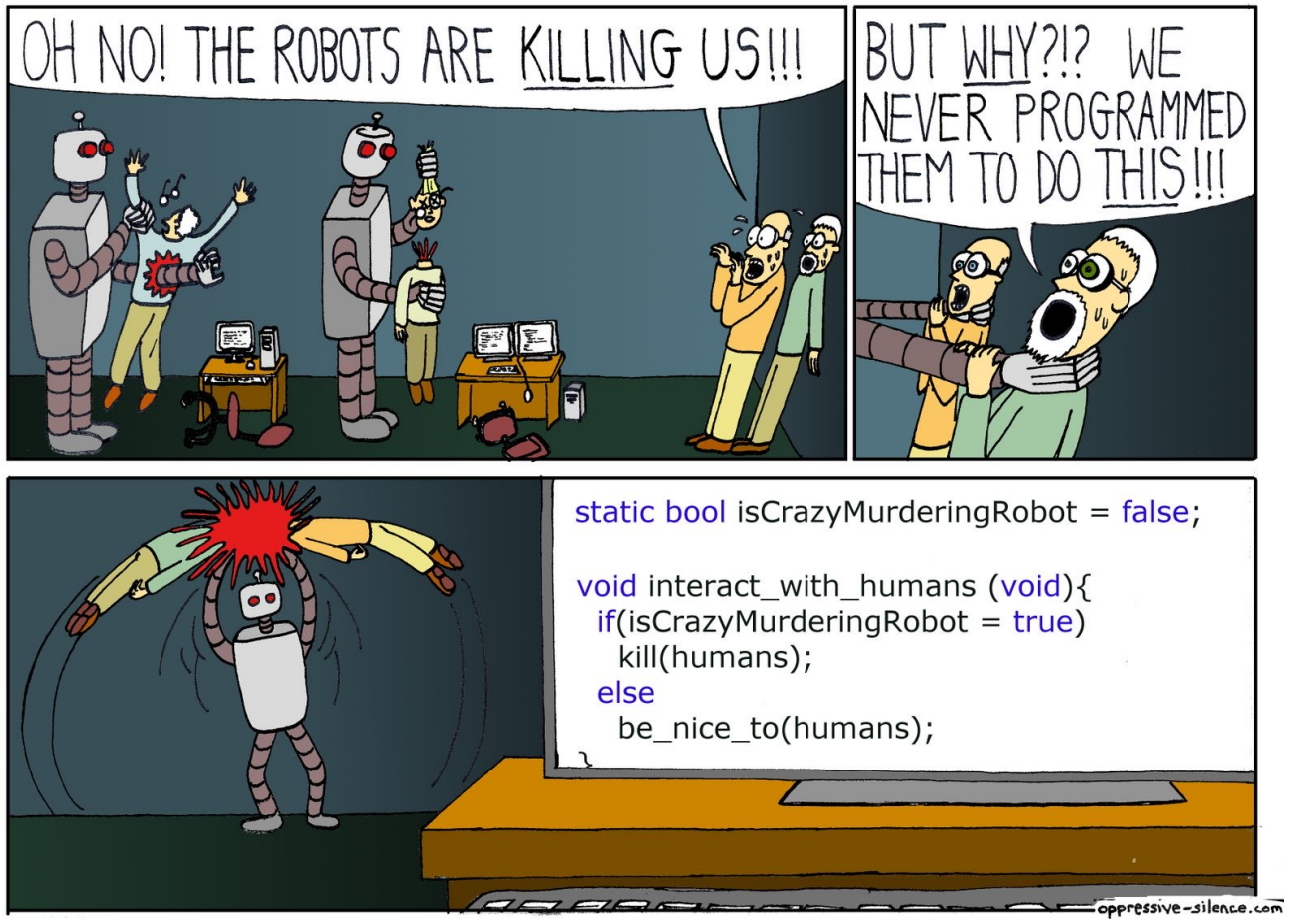
Volkan Özçelik
@linkibol

“Hope” is not a Security Posture



Volkan Özçelik
@linkibol

“Hope” is not a Security Posture



Volkan Özçelik
@linkibol

Zero Trust

- **You cannot trust the network**.
- Assume the network is breached.
- Verify explicitly.

Volkan Özçelik
@linkibol

Service Secrets

- How do you rotate them?
- How do you secure them?
- How do you expire them?
- How do you federate them?
- How do you protect **secret zero**?

Volkan Özçelik
@linkibol

It’s Turtles All the Way Down

GopherCon TR

December 2021

Volkan Özçelik

@linkibol

Solving for the Bottom Turtle

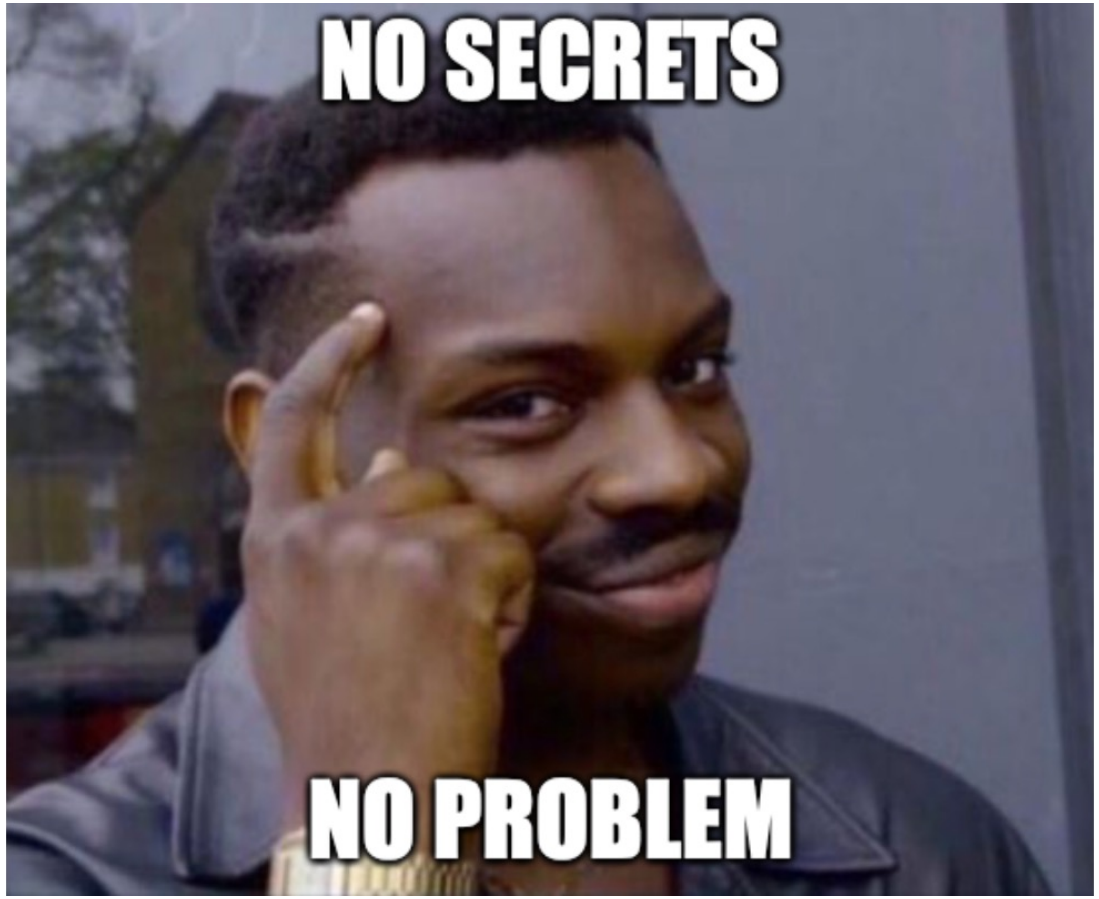
GopherCon TR

December 2021

Volkan Özçelik

@linkibol

The Shift of Trust



Volkan Özçelik
@linkibol

The Shift of Trust

- Query the node's kernel
- Query platform-specific components:
  - Kubernetes (kubelet)
  - Cloud Instance Metadata APIs
- ∴ No need for a secret to prove identity

Meet SPIFFE and SPIRE

SPIFFE

SPIRE

Volkan Özçelik
@linkibol

SPIRE Is the Key Maker

GopherCon TR

December 2021

Volkan Özçelik

@linkibol

SPIRE

- Platform agnostic
- Can attest at ***process level*** and ***kernel level***
- Works well with container schedulers
  - (*e.g., Kubernetes*)

Volkan Özçelik
@linkibol

SPIRE

- **Automate Securely**
  - Automatically establish secure inter-service communication.
- **Authenticate Securely**
  - Connect to databases and services without using passwords.
- **Build the Bridge**
  - Extend service mesh across orgs without sharing keys.

Which Tools Implement SPIFFE?

Istio

Kuma

Consul

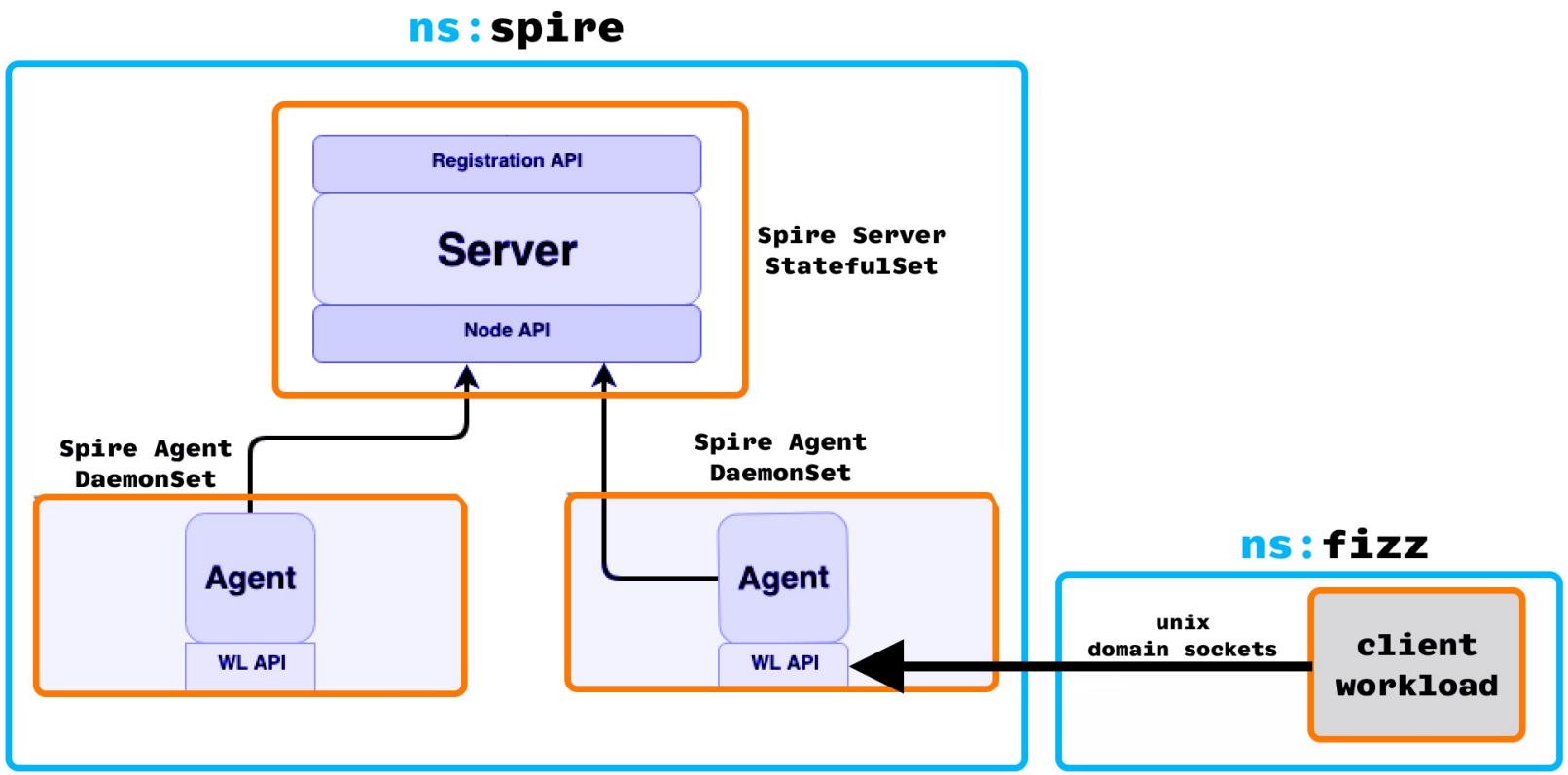
Envoy

Ghostunnel

* Not a conclusive list.

Volkan Özçelik
@linkibol

Workload Attestation



Demo

- Download and Build SPIRE
- Start SPIRE Server
- Register SPIRE Agent
- Register Workloads
- Launch Workloads
- Establish mTLS Connectivity
- Code Walk

Code Walk

Volkan Özçelik
@linkibol

```go
func main() {
    e := *env.New()  1

    configureApp(e)
    listenAndServeApp(e)
    listenAndServeMtls(e)  2
}
```


Volkan Özçelik
@linkibol

```go
func listenAndServeMtls(e env.FizzEnv) {
    mtls.ListenAndServe(service.Args{
        // … snip …
        IsDevelopment:     e.Deployment.Type == env.Development,
        MtlsServerAddress: e.Crypto.MtlsServerAddress,
        MtlsSocketPath:    e.Spire.SocketPath,
        MtlsAppName:       e.Crypto.ServiceName,
    },
        ext.SpireArgs{
            AppTrustDomain: e.Spire.AppTrustDomainFizz,
            AppPrefix:      e.Spire.AppPrefixFizz,
            AppNameDefault: e.Spire.AppNameFizzDefault,
            AppName:        e.Crypto.ServiceName,
            AppNameIdm:     e.Idm.ServiceName,
            AppNameMailer:  e.Mailer.ServiceName,
        },
    )
}
```

1

2

```go
func listenAndServeMtls(e env.FizzEnv) {
    mtls.ListenAndServe(service.Args{
        // … snip …
        IsDevelopment:     e.Deployment.Type == env.Development,
        MtlsServerAddress: e.Crypto.MtlsServerAddress,
        MtlsSocketPath:    e.Spire.SocketPath,
    }
        ext.SpireArgs{
            AppTrustDomain: e.Spire.AppTrustDomainFizz,
            AppPrefix:      e.Spire.AppPrefixFizz,
            AppNameDefault: e.Spire.AppNameFizzDefault,
            AppName:        e.Crypto.ServiceName,
            AppNameIdm:     e.Idm.ServiceName,
            AppNameMailer:  e.Mailer.ServiceName,
        },
    )
}
```

"spiffe://fizzbuzz.pro/app/fizz-default"

fizzbuzz.pro

app

fizz-default

Volkan Özçelik
@linkibol

```go
func ListenAndServe(cryptoArgs service.Args, spireArgs ext.SpireArgs) {
    ctx, cancel := context.WithCancel(context.Background())
    defer cancel()
    svc := service.New(cryptoArgs, ctx)

    mux := func(conn net.Conn) {
        handleConnection(conn, svc)
    }

    allowedIds, err := ext.AllowList(spireArgs, cryptoArgs.IsDevelopment)
    if err ≠ nil {
        log.Err("ListenAndServe: Unable to acquire SVIDs: %v", err.Error())
    }

    mtls.ListenAndServe(
        cryptoArgs.MtlsServerAddress,
        cryptoArgs.MtlsSocketPath,
        cryptoArgs.MtlsAppName,
        allowedIds,
        mux, handleError,
    )
}
```

1 2 3 4

Volkan Özçelik
@linkibol

```go
func ListenAndServe(
    serverAddress, socketPath, appName string, allowedIds []spiffeid.ID,
    handlerFn func(net.Conn), errFn func(error),
) {
    ctx, cancel := context.WithCancel(context.Background())
    defer cancel()

    log.Info("SPIRE mTLS server will try listening… (%s)", serverAddress)

    listener, err := spiffetls.ListenWithMode(ctx, "tcp", serverAddress,
        spiffetls.MTLSServerWithSourceOptions(
            tlsconfig.AuthorizeOneOf(allowedIds...),
            workloadapi.WithClientOptions(workloadapi.WithAddr(socketPath)),
        ))

    // … snip …

    defer func() {
        err := listener.Close()
        if err ≠ nil {
            log.Err("SPIRE: Possibly leaking a listener: '%s'", err.Error())
        }
    }()

    for {
        conn, err := listener.Accept()
        if err ≠ nil {
            go errFn(err)
            continue
        }
        go handlerFn(conn)
    }
}
```

1

2

3

Volkan Özçelik
@linkibol

```go
svc := service.New(cryptoArgs, ctx)

mux := func(conn net.Conn) {
    handleConnection(conn, svc)
```

1

```go
func handleConnection(conn net.Conn, svc servi
    defer func() {
        err := conn.Close()
        if err ≠ nil {
            log.Err("Unable to create TLS listener: %v", err.Error())
        }
    }()

    req, err := bufio.NewReader(conn).ReadString('\n')
    if err ≠ nil {
        log.Info("Error reading incoming data %v", err)
        return
    }

    apiRequest := &reqres.MtlsApiRequest{}

    _ = json.Unmarshal([]byte(req), apiRequest)

    multiplex(apiRequest.Endpoint, apiRequest.Method, req, conn, svc)
}
```

2

3

4

Volkan Özçelik
@linkibol

```go
func multiplex(
    apiEndpoint string,
    apiMethod method.Method,
    body string,
    conn net.Conn, svc service.Service,
) {
    switch {
    case apiEndpoint == endpoint.Crypto.SecureHashVerify && apiMethod ==
method.Post:
        handleSecureHashVerify(conn, svc, body)
    case apiEndpoint == endpoint.Crypto.Jwt && apiMethod == method.Post:
        handleJwt(conn, svc, body)
    case apiEndpoint == endpoint.Crypto.JwtVerify && apiMethod ==
method.Post:
        handleJwtVerify(conn, svc, body)
    case apiEndpoint == endpoint.Crypto.SecureHash && apiMethod ==
method.Post:
        handleSecureHash(conn, svc, body)
    case apiEndpoint == endpoint.Crypto.SecureToken && apiMethod ==
method.Get:
        handleSecureToken(conn, svc, body)
    default:
        handleUnknown(conn, svc, body)
    }
}
```

1

Volkan Özçelik
@linkibol

```go
func handleSecureToken(conn net.Co svc service.Service, body string) {
    token, _ := svc.TokenCreate()
    res := &reqres.TokenCreateResponse{
        Token: token,
    }

    sendErr := ext.Send(conn, res)
    if sendErr ≠ nil {
        log.Err("handleSecureToken: could not send: %s", sendErr.Error())
    }
}
```

1

2

3

Volkan Özçelik
@linkibol

```go
func Send(conn net.Conn, result interface{}) error {
    serialized, _ := json.Marshal(result)
    if _, err := conn.Write([]byte(string(serialized) + "\n")); err ≠ nil {
        return errors.Wrap(err, "Unable to send a response")
    }

    return nil
}
```

```go
func main() {
    e := *env.New()

    // … snip …

    mtls.Init(e)

    // … snip …

    // For demo only!
    go func() {
        value := "GopherCon"
        res, _ := mtls.CryptoHashCreate(reqres.HashCreateRequest{
            Value: value,
        })

        fmt.Println("Generated hash:", res.Hash)

        vr, _ := mtls.CryptoHashVerify(reqres.HashVerifyRequest{
            Value: value,
            Hash:  res.Hash,
        })

        fmt.Println("Verified:", vr.Verified)
        fmt.Println("Done.")
    }()

    app.ListenAndServe(e, svcName, appEnv.Port, r)
}
```

1

2

Volkan Özçelik
@linkibol

```go
var serverAddressCrypto string
var serverAddressMailer string
var spireSocketPath string
var spireAppNameCrypto string
var spireAppNameMailer string
var requestTimeout time.Duration
var initialized = false

func Init(e env.FizzEnv) {
    if initialized {
        return
    }

    isDevelopment := e.Deployment.Type == env.Development

    serverAddressCrypto = e.Idm.MtlsServerAddressCrypto
    serverAddressMailer = e.Idm.MtlsServerAddressMailer
    spireSocketPath = e.Spire.SocketPath
    requestTimeout = e.Spire.MtlsTimeout

    if isDevelopment {
        spireAppNameCrypto = e.Spire.AppNameFizzDefault
        spireAppNameMailer = e.Spire.AppNameFizzDefault
    } else {
        spireAppNameCrypto = e.Crypto.ServiceName
        spireAppNameMailer = e.Mailer.ServiceName
    }

    initialized = true
}
```


Volkan Özçelik
@linkibol

```go
func CryptoHashCreate(request reqres.HashCreateRequest) (
    *reqres.HashCreateResponse, error,
) {
    payload := build(request, method.Post, endpoint.Crypto.SecureHash)  1

    conn, cancel := connectCrypto()
    defer disconnect(conn, cancel)()  2

    _ := send(conn, payload)  3

    var hr reqres.HashCreateResponse
    _ = deserialize(conn, &hr)  4

    return &hr, nil
}
```

```go
func serialize(request interface{}) string {
    if !initialized {
        panic("serialize: mTLS service has not been initialized")
    }

    body, _ := json.Marshal(request)

    return string(body)
}

func deserialize(conn net.Conn, response interface{}) error {
    res, _ := bufio.NewReader(conn).ReadString('\n')

    _ = json.Unmarshal([]byte(res), response)

    return nil
}
```

```go
func build(request interface{}, m method.Method, e string) string {
    return serialize(reqres.MtlsApiRequest{
        Service:  reqres.CryptoService,
        Endpoint: e, Method: m,
        Body: serialize(request),
    })
}
```

```go
func connectCrypto() (net.Conn, context.CancelFunc) {
    return connect(spireAppNameCrypto, serverAddressCrypto)
}

func disconnect(conn net.Conn, cancel context.CancelFunc) func() {
    return func() {
        cancel()
        if conn == nil {
            log.Info("mTLS: Unable to close nil connection")
            return
        }
        _ := conn.Close()
    }
}
```

```go
func connect(appName, serverAddress string) (net.Conn, context.CancelFunc) {
    ctx, cancel := context.WithTimeout(context.Background(), requestTimeout)

    spiffeId := spiffeid.Must("fizzbuzz.pro", "app", appName)

    conn, err := spiffetls.DialWithMode(ctx, "tcp", serverAddress,
        spiffetls.MTLSClientWithSourceOptions(
            tlsconfig.AuthorizeID(spiffeId),
            workloadapi.WithClientOptions(
              workloadapi.WithAddr(spireSocketPath)),
        ),
    )

    return conn, cancel
}
```

1

2

3

Volkan Özçelik
@linkibol

```go
func send(conn net.Conn, payload string) error {
    if conn == nil {
        return errors.New("send: Failed to connect downstream")
    }

    _, err := fmt.Fprintf(conn, payload+"\n")
    if err ≠ nil {
        return errors.Wrap(err, "send: Problem sending payload")
    }

    return nil
}
```

🐢 Thank You 🤗

- Questions?

Volkan Özçelik
@linkibol