Create Your Own Serverless PKI with .NET & Azure Key Vault

Eran Stiller

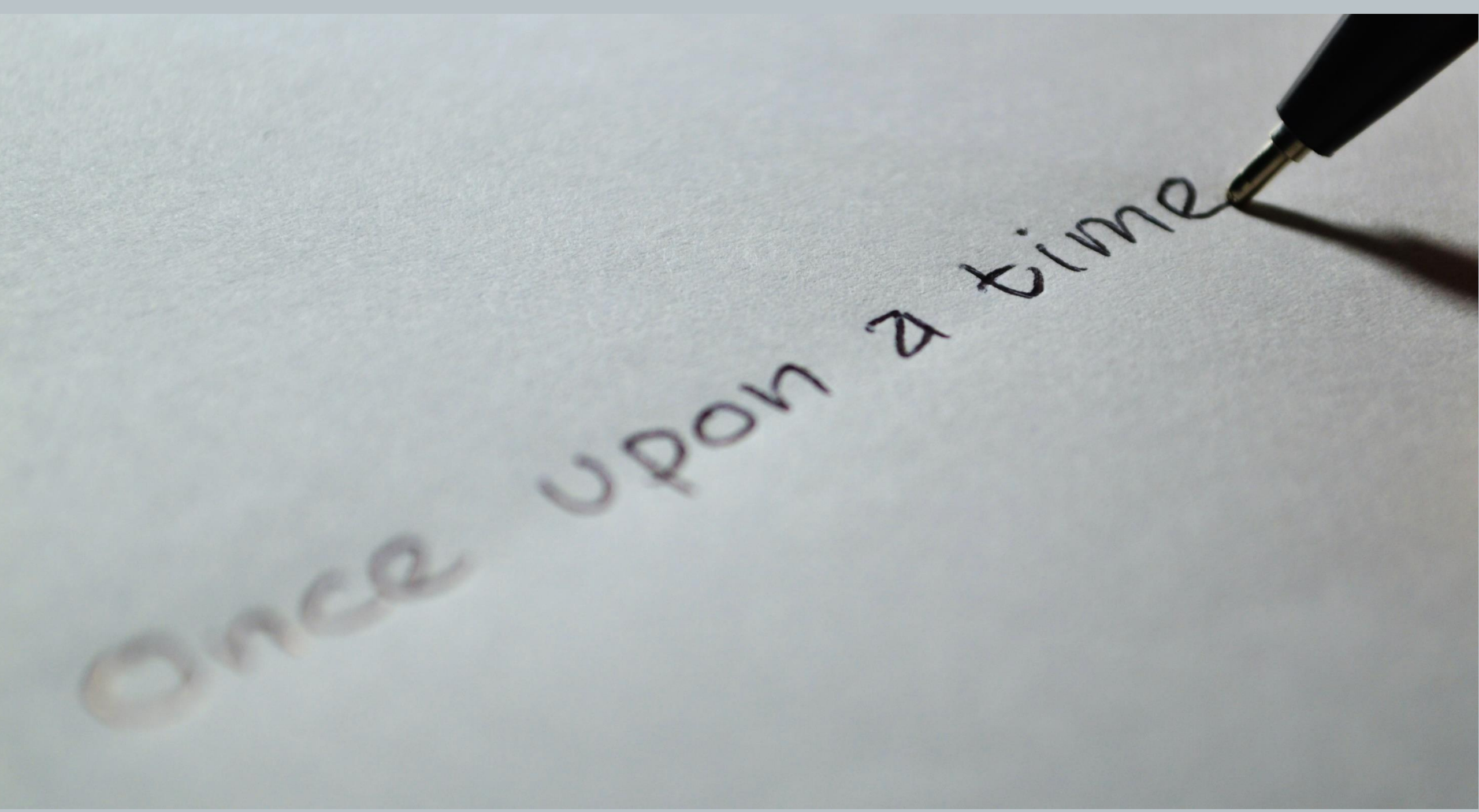
Chief Technology Officer

erans@codevalue.net

@eranstiller

https://stiller.blog

https://codevalue.net


Once upon a time


SECURITY

Agenda

- IoT Data Ingestion
- PKIs, CAs & Certificates
- Building a Serverless CA using .NET
- Authoring Certificate Subscribers using .NET
- Using the Generated Certificates
  - Device Provisioning Service & IoT Hub

About Eran

Eran Stiller

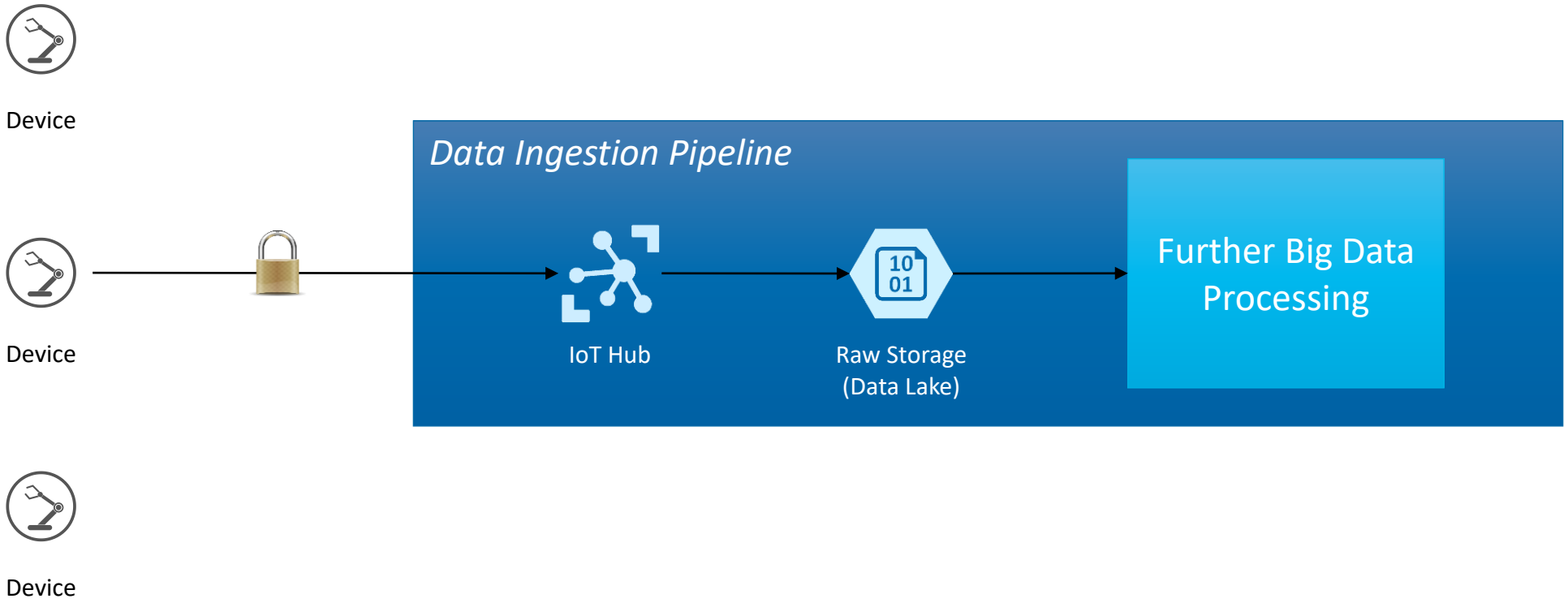
- @eranstiller
- CTO & Founder at CodeValue
- Software architect, consultant and instructor
- Microsoft Regional Director & Azure MVP
- Founder of Azure Israel Meetup

Microsoft® Most Valuable Professional

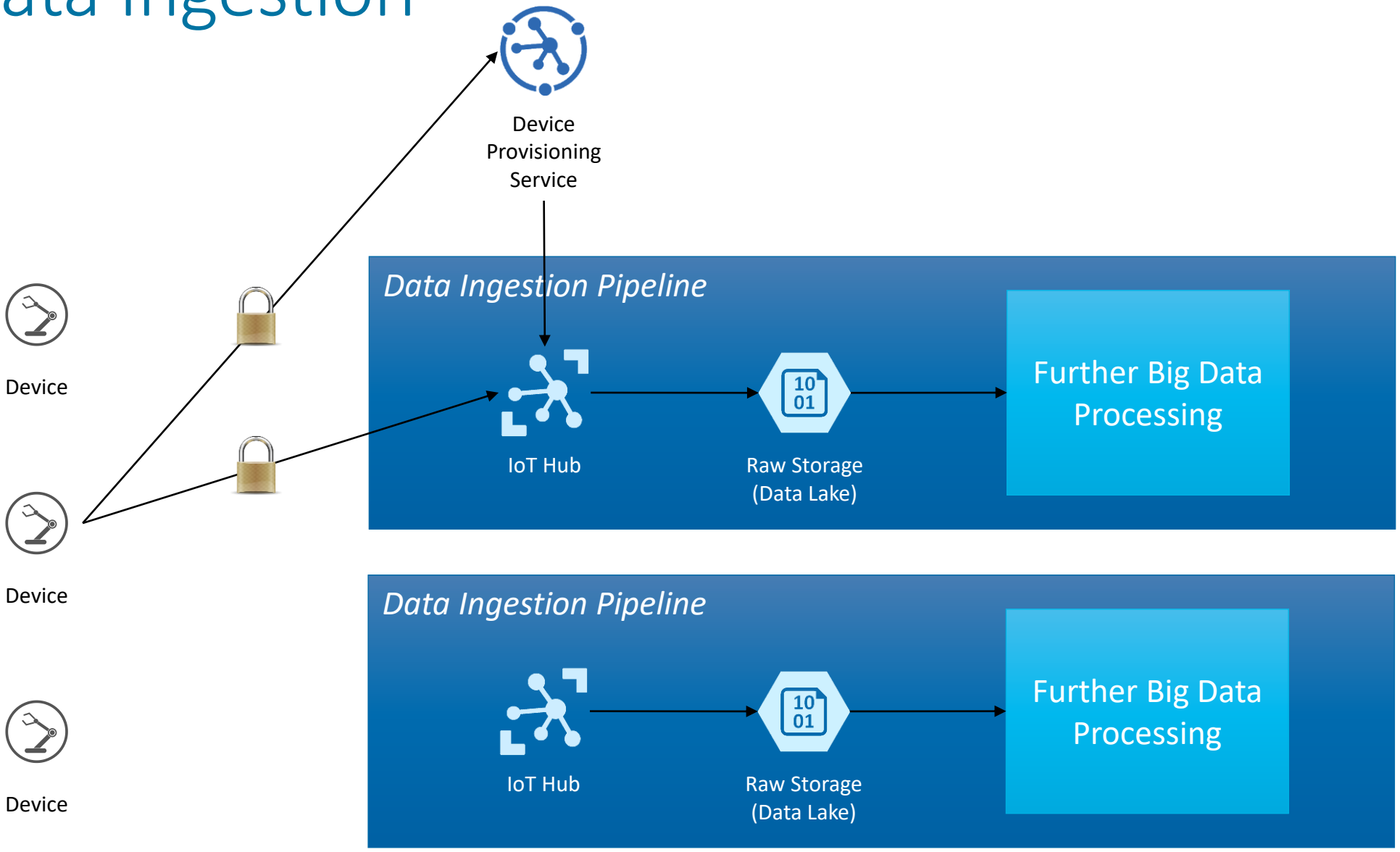
Microsoft Regional Director

IoT Data Ingestion

IoT Data Ingestion

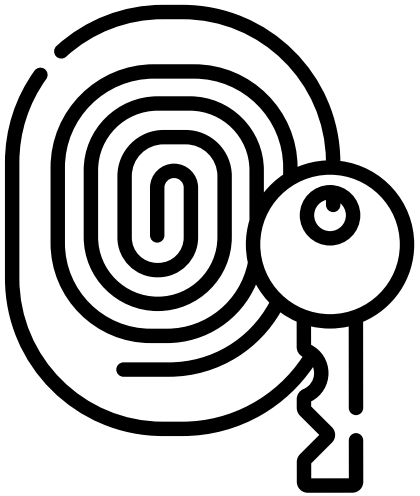


IoT Data Ingestion

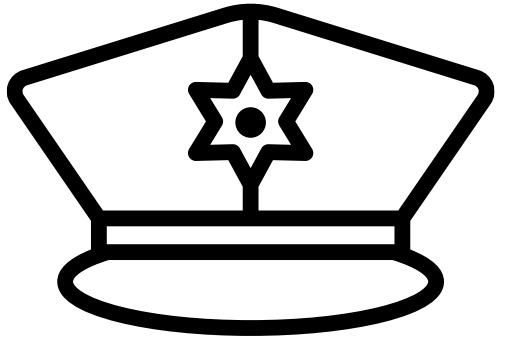


Securely Communicating with Devices

Confidentiality



Authentication



Authorization

Icons made by Freepik and Eucalyp for www.flaticon.com

Keep It Simple

PKIs, CAs & Certs

Public Key Infrastructure (PKI)

- An umbrella term for the stuff we need in order to:
  - Issue
  - Distribute
  - Store
  - Use
  - Verify
  - Revoke
  - and otherwise manage and interact with certificates and keys
- Don’t build from scratch
  - Can use an off-the-shelf solution
  - Can build some parts and rely on others

Certificates

- A driver's license for **computers and code**
- Basically, a binding between an identifier (name) and a public key
- Usually encoded as X.509

Icon made by Becris for www.flaticon.com

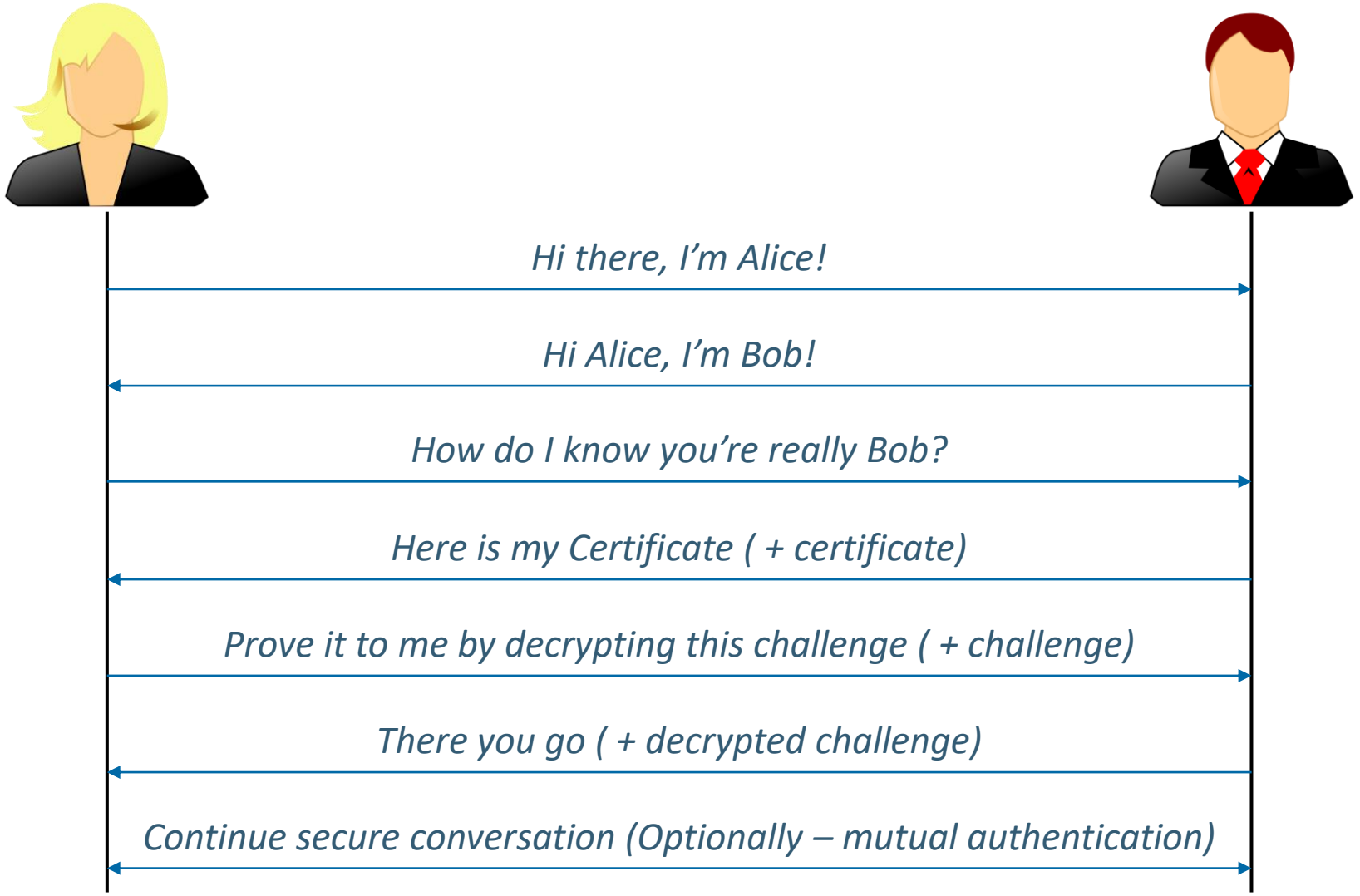
Certificate Authority

- The entity which issues certificates
- Trusted by the Relying Parties
  - Public trusted root certificates are pre-populated
- Various methods to verify identity

Demo

Certificate Content

Certificates 101



Building a Serverless CA using .NET

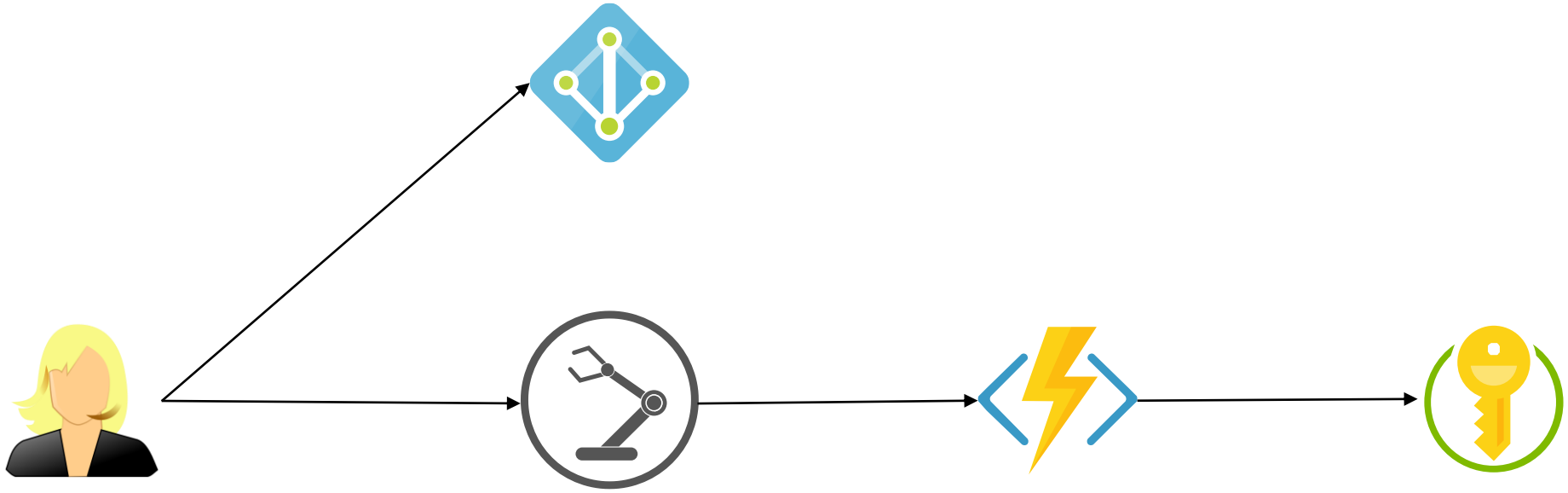
The Root Certificate

Azure Key Vault

- Safeguard cryptographic keys and other secrets
- Hardware Security Modules (HSM) as a service
- Can be replaced with AWS Certificate Manager
  - Principles will remain unchanged
  - Implementation details will defer

Scenario

Generating a new certificate on a new device

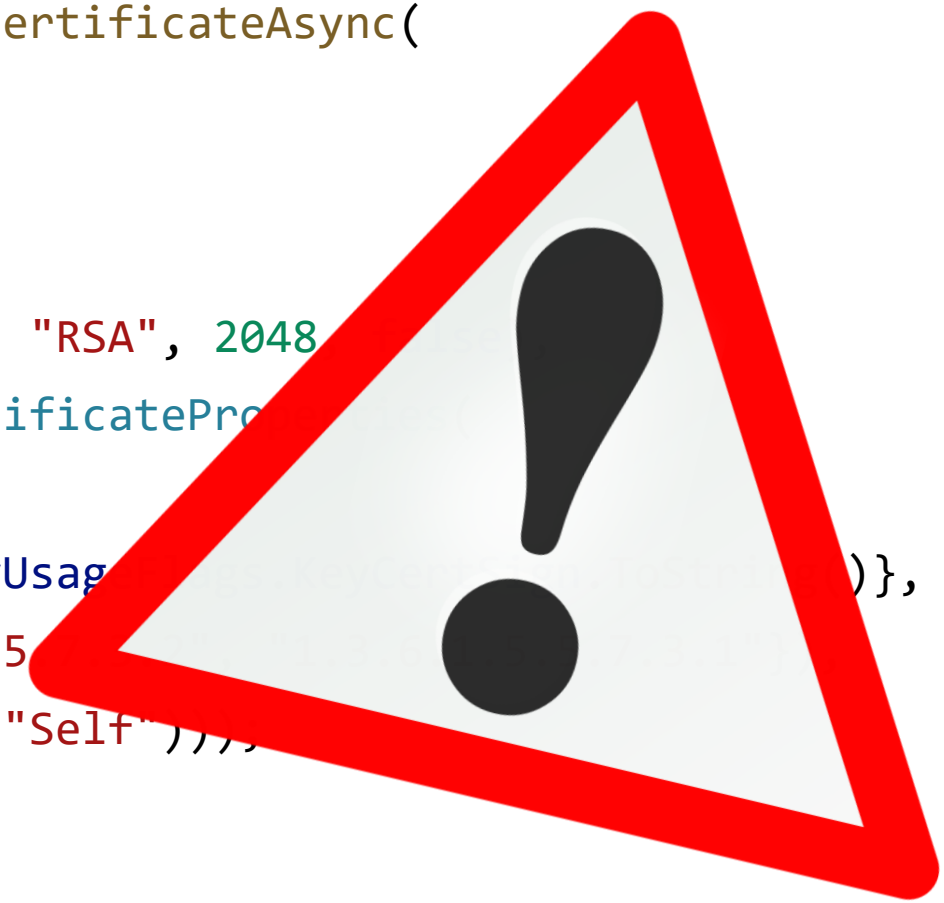


Demo

Setup Azure
Key Vault & Other
Resources

Generating the Root Certificate on Key Vault

```
var certificateOperation = await client.CreateCertificateAsync(
    _vaultBaseUrl,
    _certificateName,
    new CertificatePolicy(
        keyProperties: new KeyProperties(false, "RSA", 2048
        x509CertificateProperties: new X509CertificatePr
            "CN=" + RootSubjectName,
            keyUsage: new List<string> {X509KeyUsag
            ekus: new List<string> {"1.3.6.1.5.5
        issuerParameters: new IssuerParameters("Self")));
```



Generating the Root Certificate on Our Machine

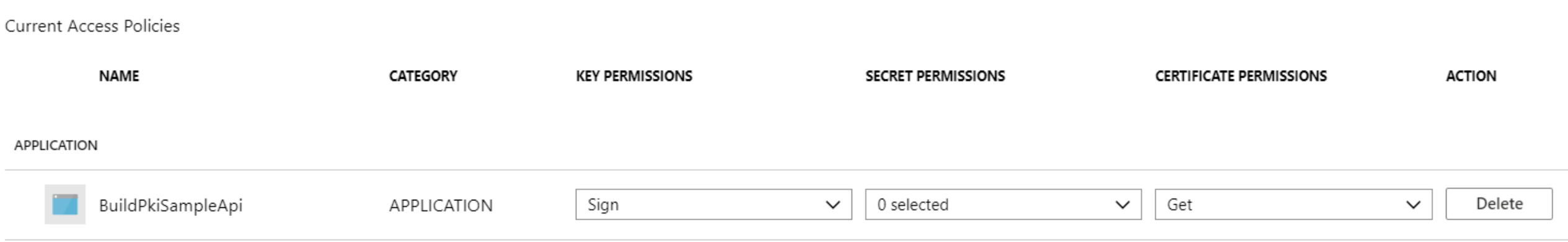
```
using var certificateKey = RSA.Create();
var subjectDistinguishedName = new X500DistinguishedName("CN=" + RootSubjectName);
var request = new CertificateRequest(subjectDistinguishedName, certificateKey,
    HashAlgorithmName.SHA256, RSASignaturePadding.Pkcs1);

request.CertificateExtensions.Add(
    new X509KeyUsageExtension(X509KeyUsageFlags.KeyCertSign, true));
request.CertificateExtensions.Add(
    new X509BasicConstraintsExtension(true, true, 1, true));
// Additional X509 extensions not shown for brevity

var certificate =
    request.CreateSelfSigned(DateTimeOffset.UtcNow, DateTimeOffset.UtcNow.AddYears(1));
```

Demo

Generating the
Root Certificate

Limiting Root Certificate Access

- Use Azure Key Vault Access Policies



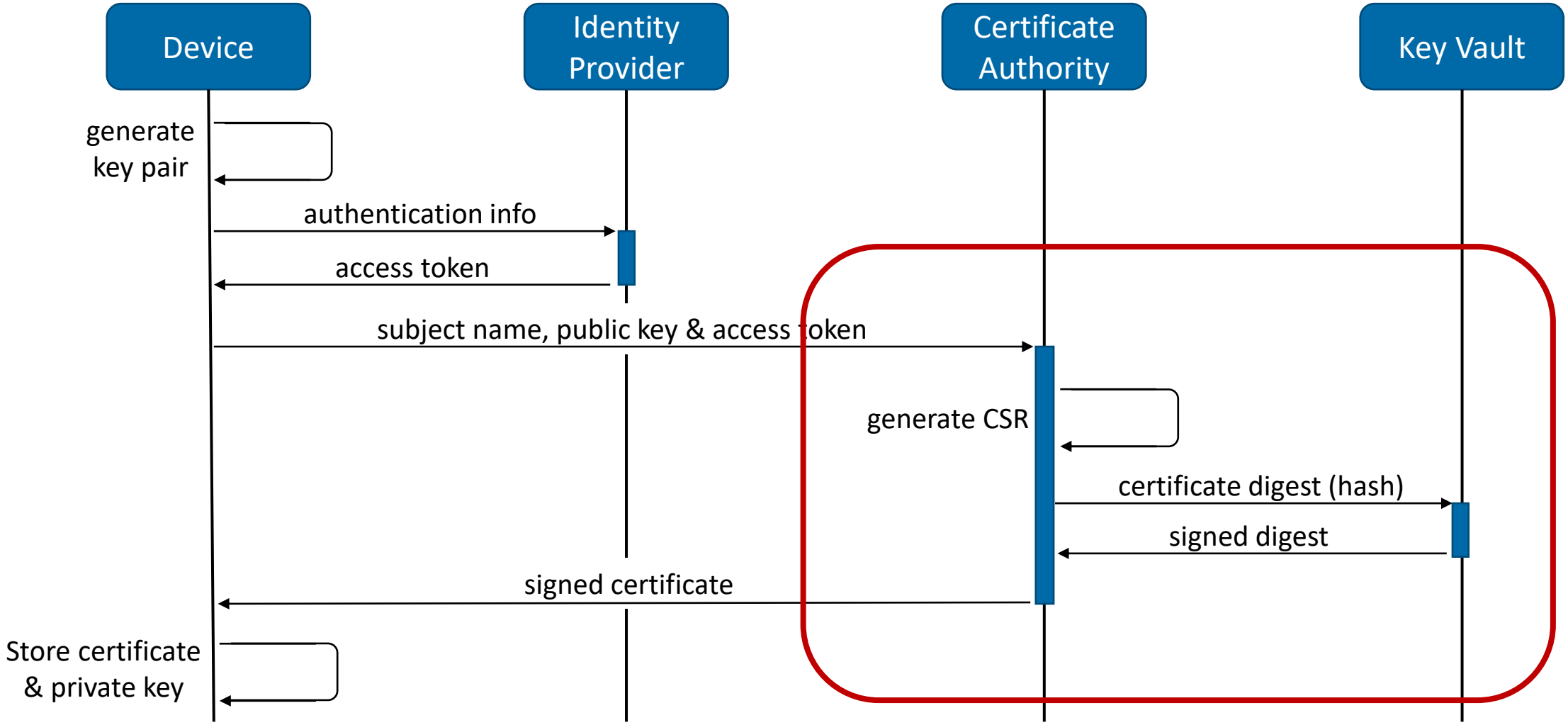
Auditing Root Certificate Access


```
{
    "time": "2016-01-05T01:32:01.2691226Z",
    "resourceId": "/SUBSCRIPTIONS/361DA5D4-A47A-4C79-AFDD-
XXXXXXXXXXXX/RESOURCEGROUPS/CONTOSOGROUP/PROVIDERS/MICROSOFT.KEYVAULT/VAULTS/CONTOSOKEYVAU
LT",
    "operationName": "VaultGet",
    "operationVersion": "2015-06-01",
    "category": "AuditEvent",
    "resultType": "Success",
    "resultSignature": "OK",
    "resultDescription": "",
    "durationMs": "78",
    "callerIpAddress": "104.40.82.76",
    "correlationId": "",
    "identity": {
        "claim": {
            "http://schemas.microsoft.com/identity/claims/objectidentifier":
                "d9da5048-2737-4770-bd64-XXXXXXXXXXXX"
```

```
    "durationMs": "78",
    "callerIpAddress": "104.40.82.76",
    "correlationId": "",
    "identity": {
        "claim": {
            "http://schemas.microsoft.com/identity/claims/objectidentifier":
                "d9da5048-2737-4770-bd64-XXXXXXXXXXXX",
            "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn":
                "live.com#username@outlook.com",
            "appid": "1950a258-227b-4e31-a9cf-XXXXXXXXXXXX"
        }
    },
    "properties": {
        "clientInfo": "azure-resource-manager/2.0",
        "requestUri": "https://control-prod-
wus.vaultcore.azure.net/subscriptions/361da5d4-a47a-4c79-afdd-
XXXXXXXXXXXX/resourcegroups/contosoresourcegroup/providers/Microsoft.KeyVault/vaults/conto
sokeyvault?api-version=2015-06-01",
        "id": "https://contosokeyvault.vault.azure.net/",
        "httpStatusCode": 200
    }
}
```

Sign a Request Using the Root Certificate

Sign a Request Using the Root Certificate



Generate Certificate Signing Request (CSR)

```
var parameters = new RSAParameters
    { Modulus = publicKey.Modulus, Exponent = publicKey.Exponent };
var certificateKey = RSA.Create(parameters);

var subjectDistinguishedName = new X500DistinguishedName("CN=" + subjectName);
var request = new CertificateRequest(subjectDistinguishedName, certificateKey,
    HashAlgorithmName.SHA256, RSASignaturePadding.Pkcs1);
request.CertificateExtensions.Add(
    new X509KeyUsageExtension(
        X509KeyUsageFlags.DigitalSignature | X509KeyUsageFlags.KeyEncipherment, true));
request.CertificateExtensions.Add(
    new X509BasicConstraintsExtension(false, true, 0, true));
// Additional X509 not shown for brevity

return request;
```

Sign Certificate with Root Key

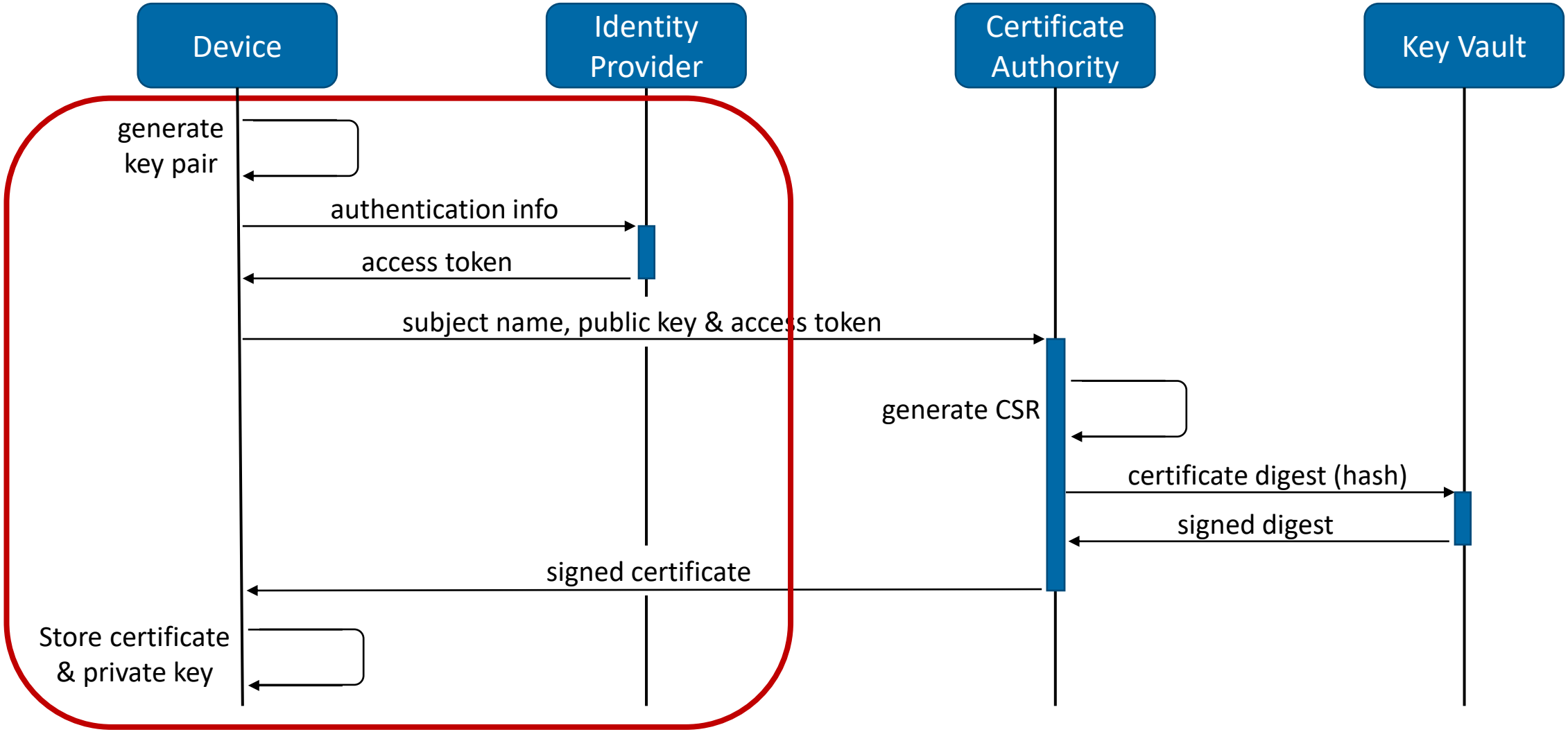
```
CertificateRequest request = CreateCertificateRequest(subjectName, certificateKey);
byte[] certificateSerialNumber = await _serialNumberGenerator.GenerateSerialAsync();

RSA rsaKeyVault = _keyVaultClient.ToRSA(
    certificateBundle.KeyIdentifier, issuerCertificate);
var generator = X509SignatureGenerator.CreateForRSA(
    rsaKeyVault, RSASignaturePadding.Pkcs1);

var certificate = request.Create(
    issuerCertificate.SubjectName, generator,
    DateTime.Today, DateTime.Today.AddYears(1),
    certificateSerialNumber);
```

Demo

Serverless CA using
.NET & Azure Functions

Authoring Certificate Subscribers using .NET

Sign a Request Using the Root Certificate



Getting an Access Token for Azure Functions

```
var auth = await authHelper.AcquireTokenAsync();  // AAD Token

var client = new HttpClient
    { BaseAddress = new Uri(configuration.BaseUrl) };
var request = new {access_token = auth.AccessToken};
var httpContent =  ... // JSON serialization
var responseMessage = await client.PostAsync(".auth/login/aad", httpContent);

var serializedResponse = await responseMessage.Content.ReadAsStringAsync();
dynamic response = JsonConvert.DeserializeObject<dynamic>(serializedResponse);
return response.authenticationToken;
```

Issue Certificate

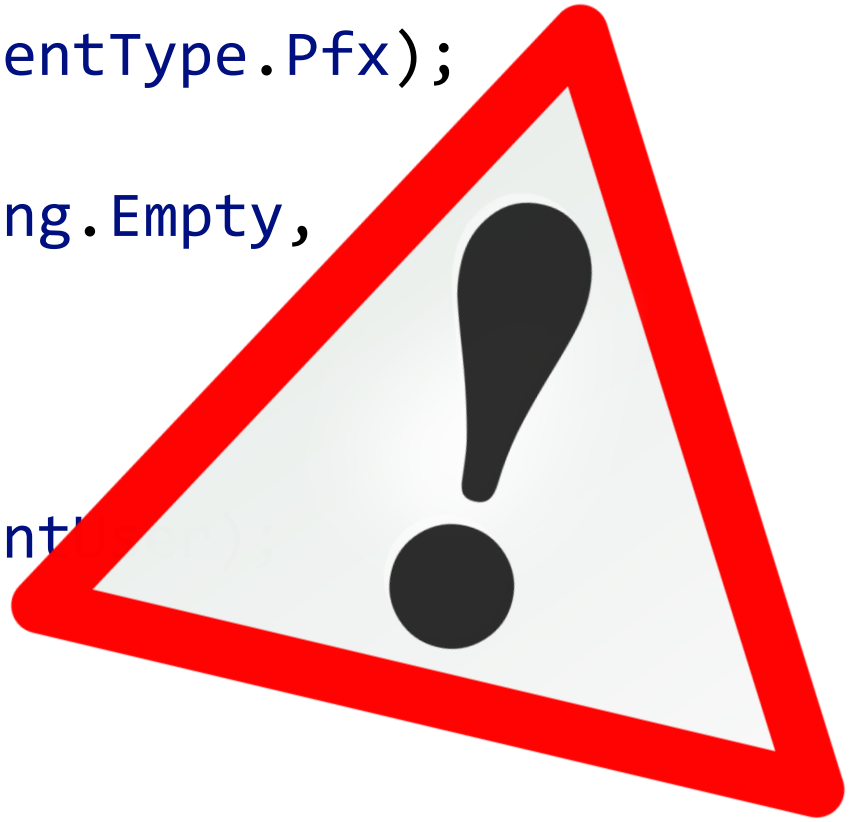
```
var key = RSA.Create();
var publicParameters = key.ExportParameters(false);
var request = new IssueCertificateRequest(
    subjectName, publicParameters);

var httpContent = new StringContent(
    JsonConvert.SerializeObject(request),
    Encoding.UTF8,
    MediaTypeNames.Application.Json);
```

Issue Certificate (Cont.)

```
var client = new HttpClient {BaseAddress = ...};
client.DefaultRequestHeaders.Add("X-ZUMO-AUTH", accessToken);
var responseMessage =
    await client.PostAsync("api/issueCertificate", httpContent);

var serializedResponse =
    await responseMessage.Content.ReadAsStringAsync();
var response = JsonConvert.DeserializeObject<IssueCertificateResponse>(
    serializedResponse);
var certificate = new X509Certificate2(
    Convert.FromBase64String(response.Certificate));
return certificate;
```

Store Certificate With Private Key

```
var certificateWithPrivateKey =
    certificate.CopyWithPrivateKey(key);
var rawCertificate =
    certificateWithPrivateKey.Export(X509ContentType.Pfx);
var persistableCertificate =
    new X509Certificate2(rawCertificate, string.Empty,
        X509KeyStorageFlags.PersistKeySet |
        X509KeyStorageFlags.UserKeySet);

var store = new X509Store(StoreLocation.Current
store.Open(OpenFlags.ReadWrite);
store.Add(persistableCertificate);
```

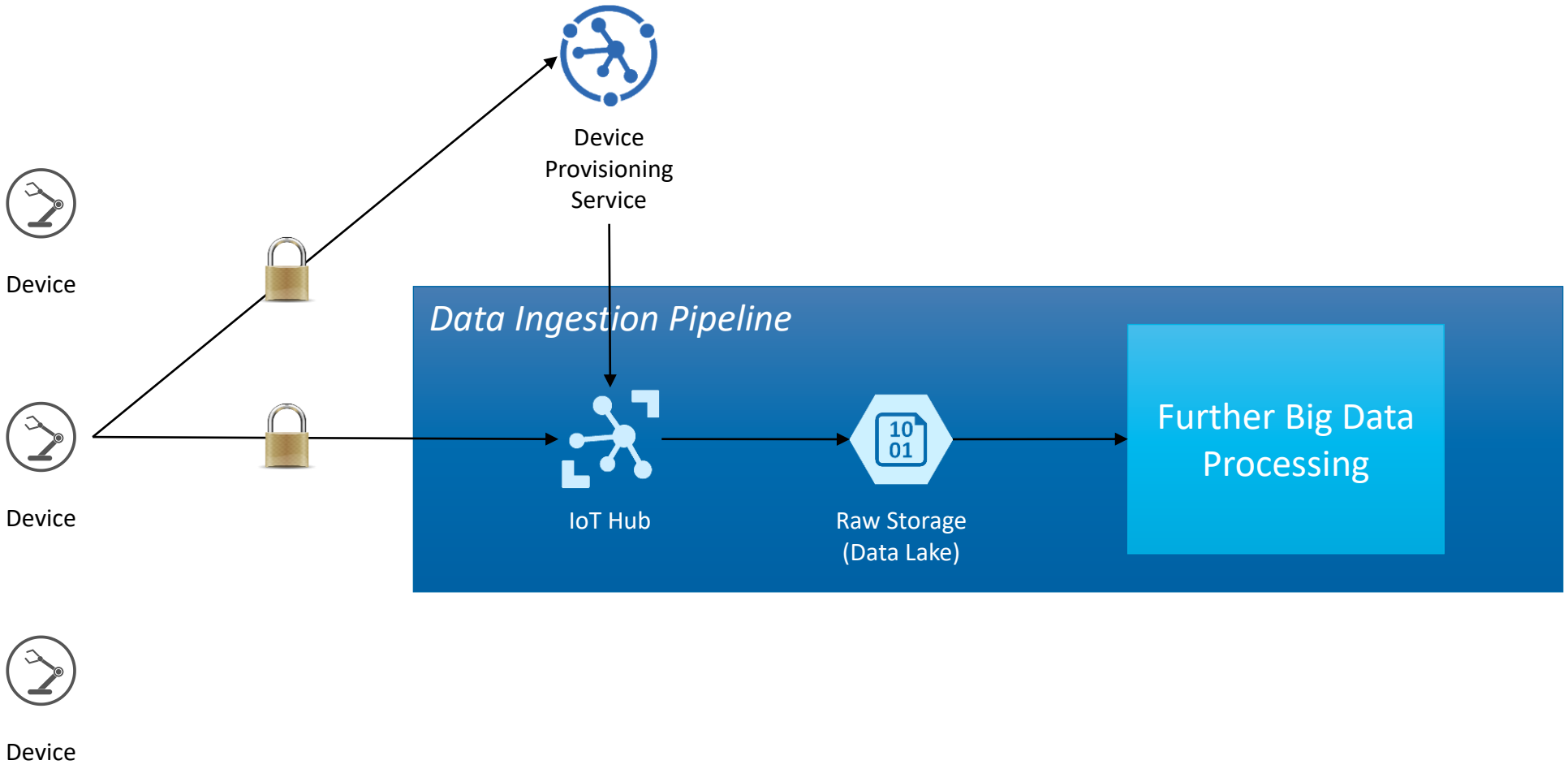


Demo

Certificate Subscribers using .NET

Using the Generated Certificates

IoT Data Ingestion (Reminder)



Sending Data Using the Generated Certificate

```
X509Certificate2 certificate =
    LoadCertificate(configuration.DeviceName);
var security =
    new SecurityProviderX509Certificate(certificate);
var transport = new ProvisioningTransportHandlerAmqp(
    TransportFallbackType.TcpOnly);

var provClient = ProvisioningDeviceClient.Create(
    GlobalDeviceEndpoint, configuration.DpsIdScope,
    security, transport);
DeviceRegistrationResult registrationResult =
    await provClient.RegisterAsync();
```

Demo

End-to-End Operation

Takeaways

Takeaways

- Certificates are hard, but crucial, to get right
- Don’t author an entire PKI from scratch
  - Customize an existing solution where appropriate
- IoT is one scenario where I encountered a need for a custom PKI
- Handling certificates with .NET is surprisingly undocumented
  - With and without Azure Key Vault
- Azure Key Vault is a great platform to base a CA on
- The sample is just a sample, but is derived from a production system
  - Can base on it and form your own solution

Resources

- Source Code
  - https://github.com/estiller/build-pki-net-azure-sample
- NuGet package to integrate Key Vault with .NET Cryptographic Keys
  - https://github.com/onovotny/RSAKeyVaultProvider
- Additional resources
  - https://smallstep.com/blog/everything-pki.html
  - https://docs.microsoft.com/en-in/azure/key-vault/certificate-scenarios
  - https://docs.microsoft.com/en-us/azure/key-vault/key-vault-logging

Thank you!

Eran Stiller

Chief Technology Officer

erans@codevalue.net

@eranstiller

https://stiller.blog

https://codevalue.net

Microsoft MVP Most Valuable Professional

Microsoft Regional Director

CODEVALUE