# Módulo fabric-ca-client del SDK de Fabric para gestionar identidades criptográficas con C#

Autor: Gabriela B. Martínez Giraldo

Tutores: Camilo Denis González
Daniel Mena Frias
Miguel Katrib Mora

# Hyperledger Fabric (HLF)

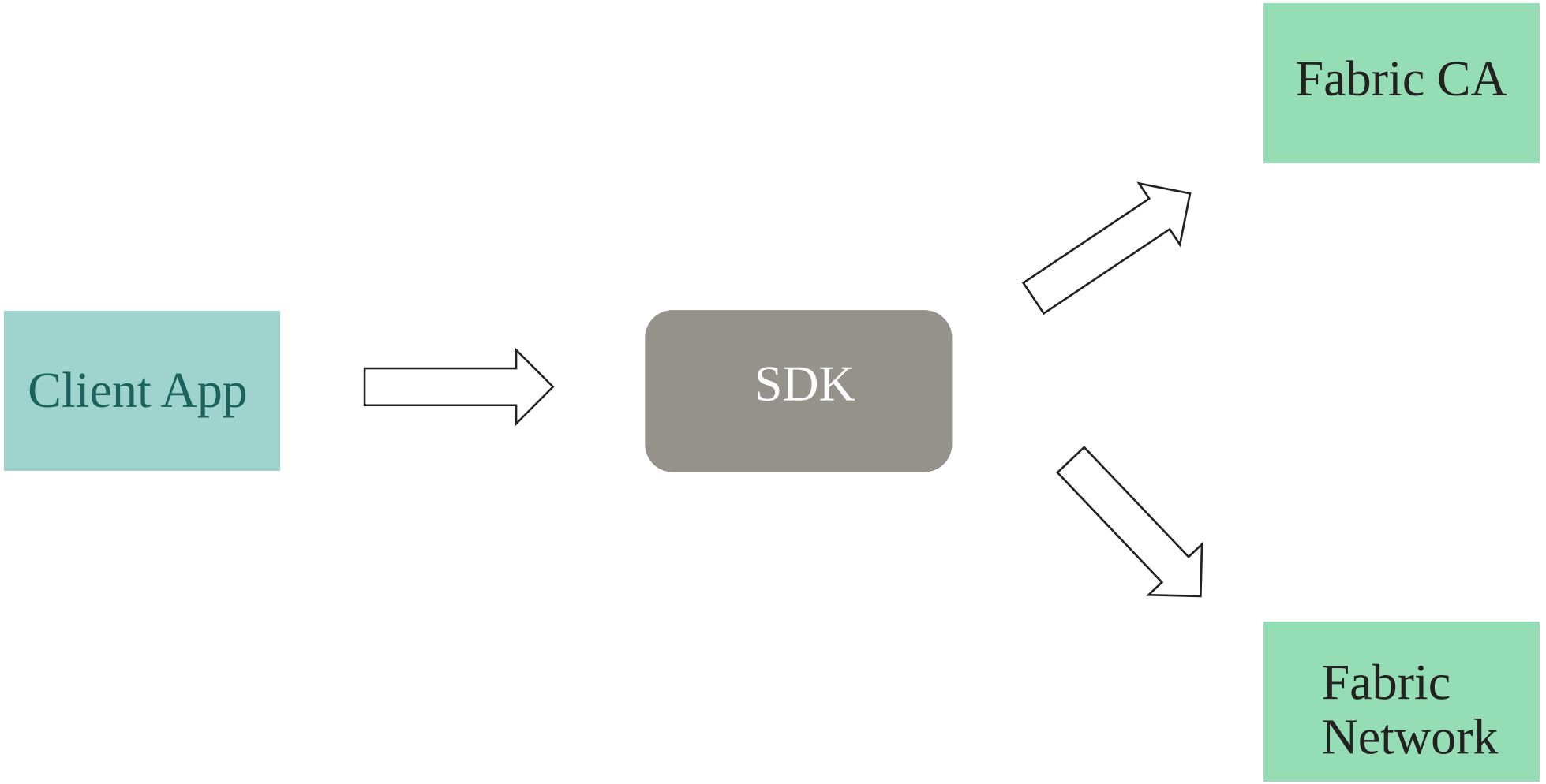
- Tecnología de redes distribuidas (Domain Ledger Technology DLT)
- Naturaleza permisionada
- Privacidad y confidencialidad
- Arquitectura modular y altamente configurable
- Alto rendimiento en cuanto al procesamiento de transacciones
- Baja latencia de confirmación de transacciones

# Hyperledger Fabric (HLF)

- Contratos inteligentes en lenguajes de propósito general

1. Java
2. Node.js
3. Golang

# Fabric SDKs



# ¿Por qué C# ?

- Pequeña curva de aprendizaje
- Cercanía con C y C++
- Versatilidad
- Madurez
- Amplio soporte

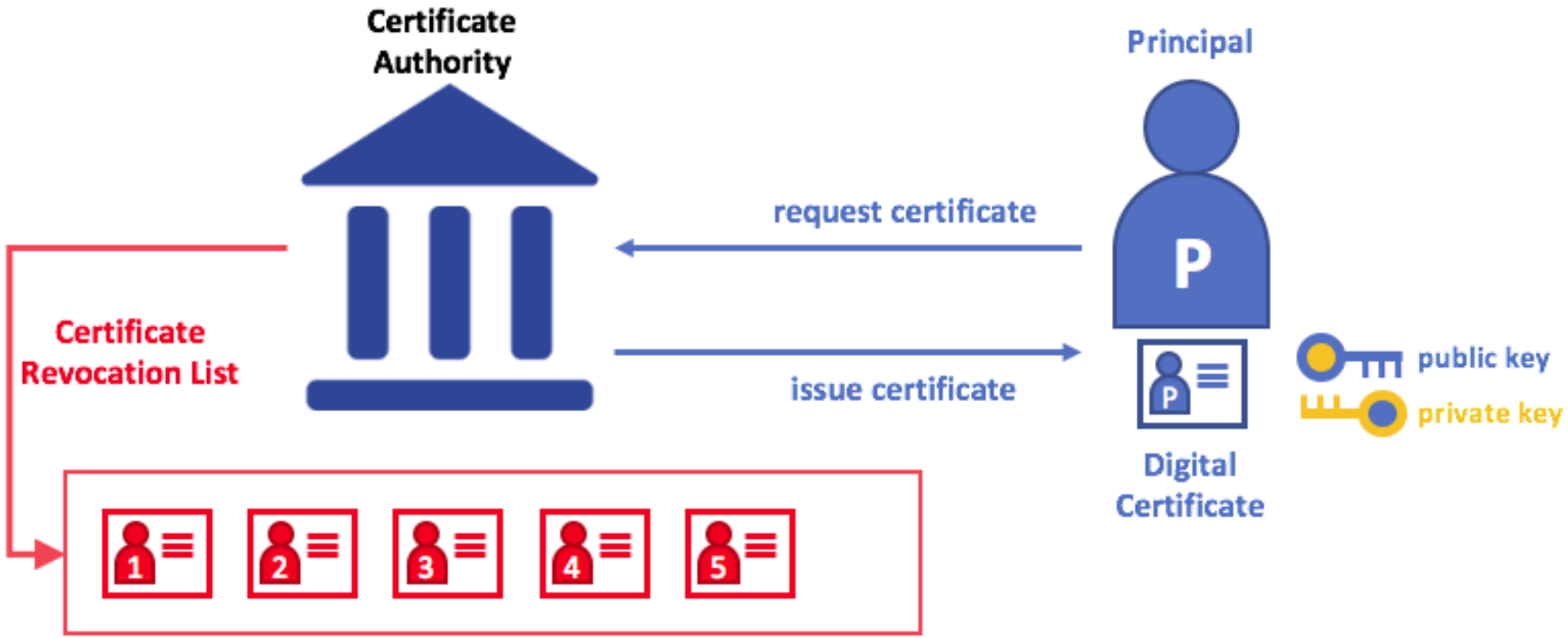
# Objetivo General

- Diseñar e implementar el módulo fabric-ca-client como módulo inicial de un Fabric-SDK escrito en C#, con el fin de poder interactuar con la Fabric CA de HLF utilizando el lenguaje mencionado.

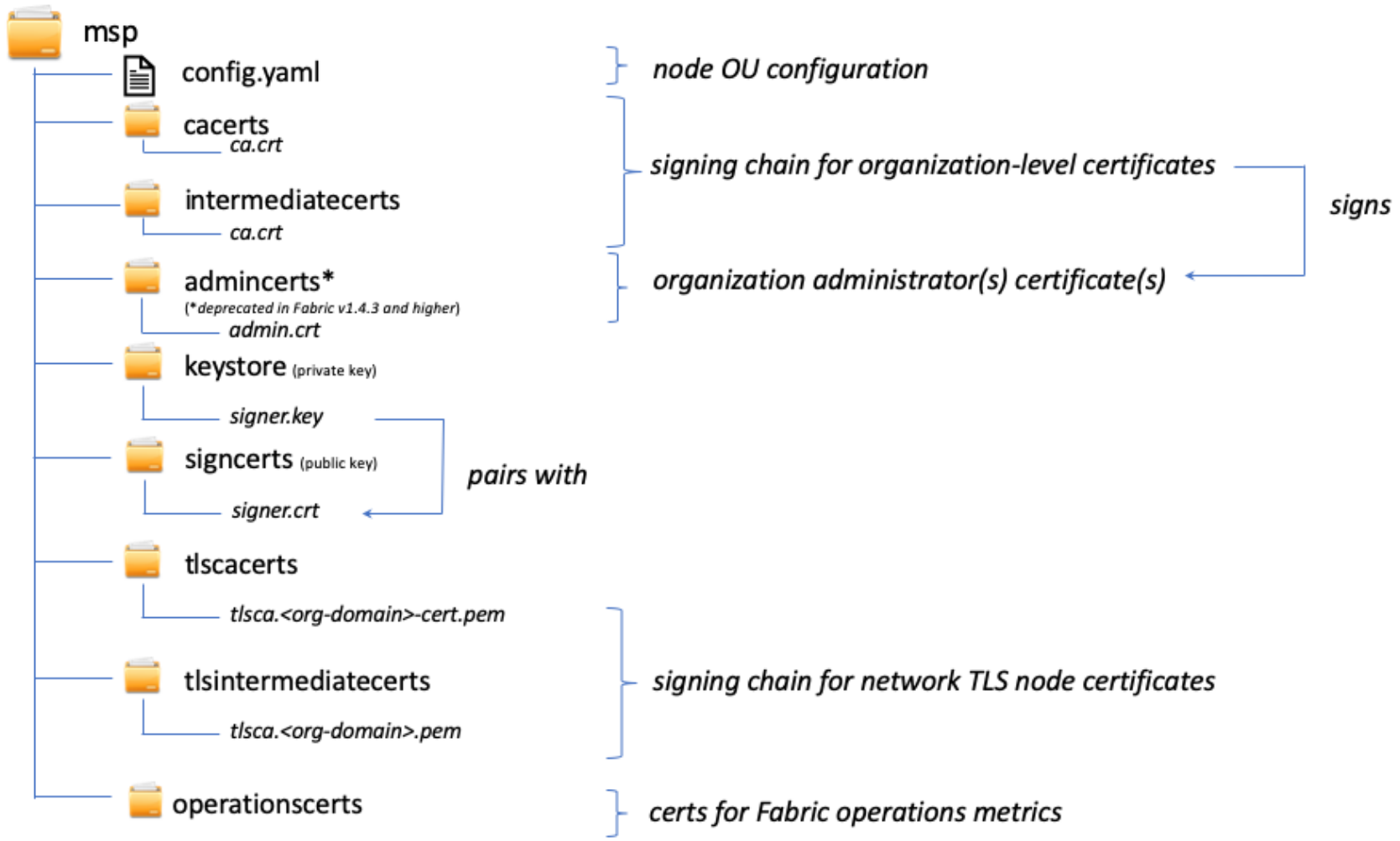
# Objetivos Específicos

- Estudio de las bibliotecas fabric-sdk existentes
- Análisis de aspectos a reutilizar, traducir, agregar
- Diseño e implementación del módulo fabric-ca-client (funciones de registro, inscripción, renovación y revocación de certificados)
- Prueba y validación de resultados

# Infraestructura de Clave Pública (PKI)



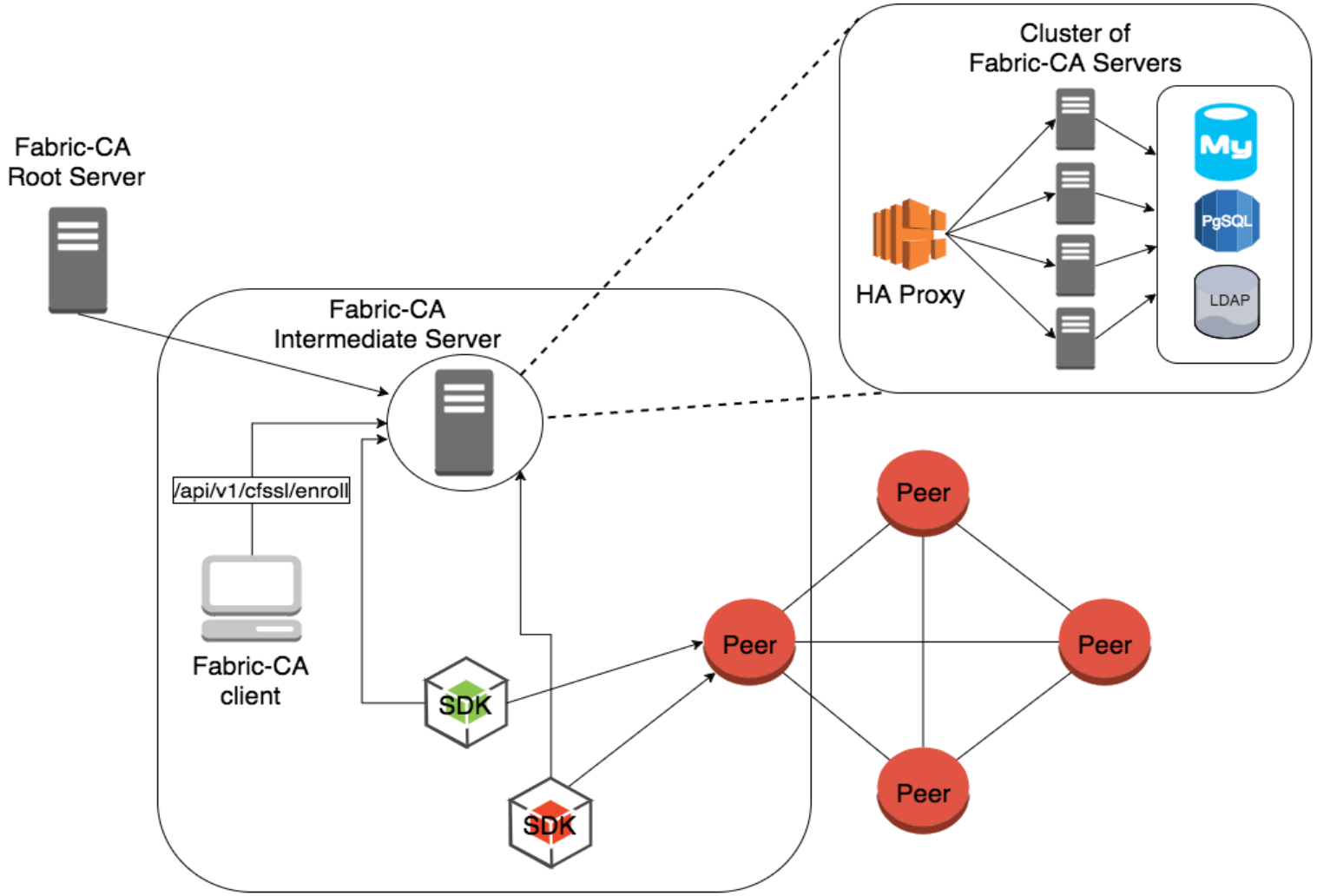
# Proveedor de Servicios de Membresía (MSP)



# Manejo de identidades

MSP + PKI

# Comunicación Fabric CA – Red de Fabric

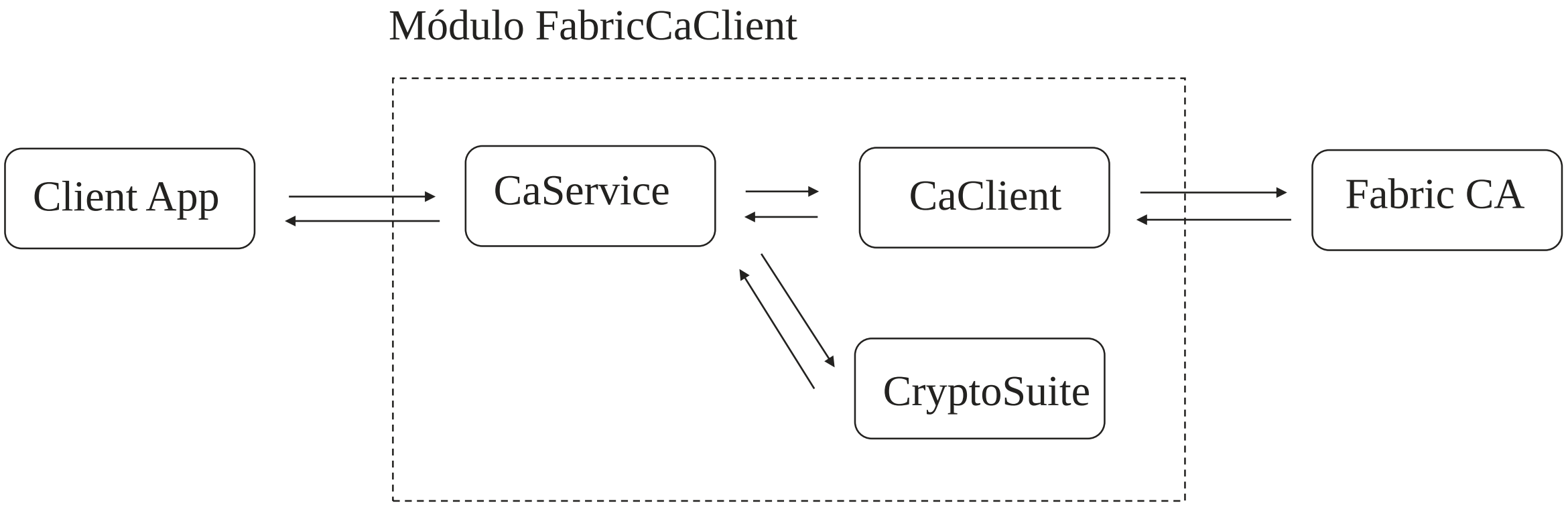


# Módulo FabricCAClient

## CAService y CaClient

- Registro
- Inscripción
- Reinscripción
- Revocación de certificado o identidad

# Fabric SDK C#


Módulo FabricCaClient
Client App
CaService
CaClient
Fabric CA
CryptoSuite

# Módulo FabricCAClient

## CAService. Registro

```
/// <summary> Registers an identity.
5 references
public async Task<string> Register(string enrollmentId, string enrollmentSecret,
                                   int maxEnrollments, Tuple<string, string, bool>[] attrs,
                                   Enrollment registrar, string role = "", string affiliatiton = "") {
    if (string.IsNullOrEmpty(enrollmentId))
        throw new ArgumentException("EntrollmentID cannot be null or empty." +
            "                       Please provide an unique id to identify the identity for enrollment.");
    if (registrar == null)
        throw new ArgumentException("Registrar should be a valid Enrollment instance.");

    return await _caClient.Register(enrollmentId, enrollmentSecret,
                                    maxEnrollments, attrs, registrar, role, affiliatiton);
}
```

# Módulo FabricCAClient

## CaClient. Registro

```
/// <summary> Registers an identity.
1 reference
internal async Task<string> Register(string enrollmentId, string enrollmentSecret,
                                     int maxEnrollments, Tuple<string, string, bool>[] attrs,
                                     Enrollment registrar, string role = "", string affiliatiton = "") {
    JObject jsonBody = new JObject {
        new JProperty("id", enrollmentId),
        new JProperty("affiliation", affiliatiton),
        new JProperty("max_enrollments", maxEnrollments),
    };
    if (caName != "")
        jsonBody.Add(new JProperty("caname", caName));
    if (role != "")
        jsonBody.Add(new JProperty("type", role));
    if (enrollmentSecret != "")
        jsonBody.Add(new JProperty("secret", enrollmentSecret));
    if (attrs != null) {
        // converting attrs to JArray of JObjects
        JArray attrsArray = new JArray();
        foreach (var attrTuple in attrs) {
            JObject attrObj = new JObject {
                new JProperty("name", attrTuple.Item1),
                new JProperty("value", attrTuple.Item2),
                new JProperty("ecert", attrTuple.Item3)
            };
            attrsArray.Add(attrObj);
        }
```

# Módulo FabricCAClient

## CaClient. Registro

```
    try {
        // get the result field which is Base64-encoded PEM
        var jsonResponse = await PostAsync(caUrlRegister, jsonBody.ToString(Formatting.None), registrar);

        JObject jsonst = JObject.Parse(jsonResponse);

        JObject result = jsonst["result"] as JObject;

        if (result == null)
            throw new Exception("Error in HTTP call, result not found.");

        string secret = result["secret"]?.Value<string>();

        return secret;
    }
    catch (Exception exc) {
        throw (new RegisterException("Error in register request.", exc));
    }
}
```

# Módulo FabricCAClient

## CAService, CaClient. Incripción

- Solicitud de certificados de inscripción o TLS
- Presenta ID y secret
- Creación de llaves y CSR (opcional)
- Especificación de atributos para el certificado
- Resultado: Enrollment con Cert, PrivateKey y CAChain
- Token de autenticación: ID: Clave Secreta

# Módulo FabricCAClient

## CAService, CaClient. Reinscripción

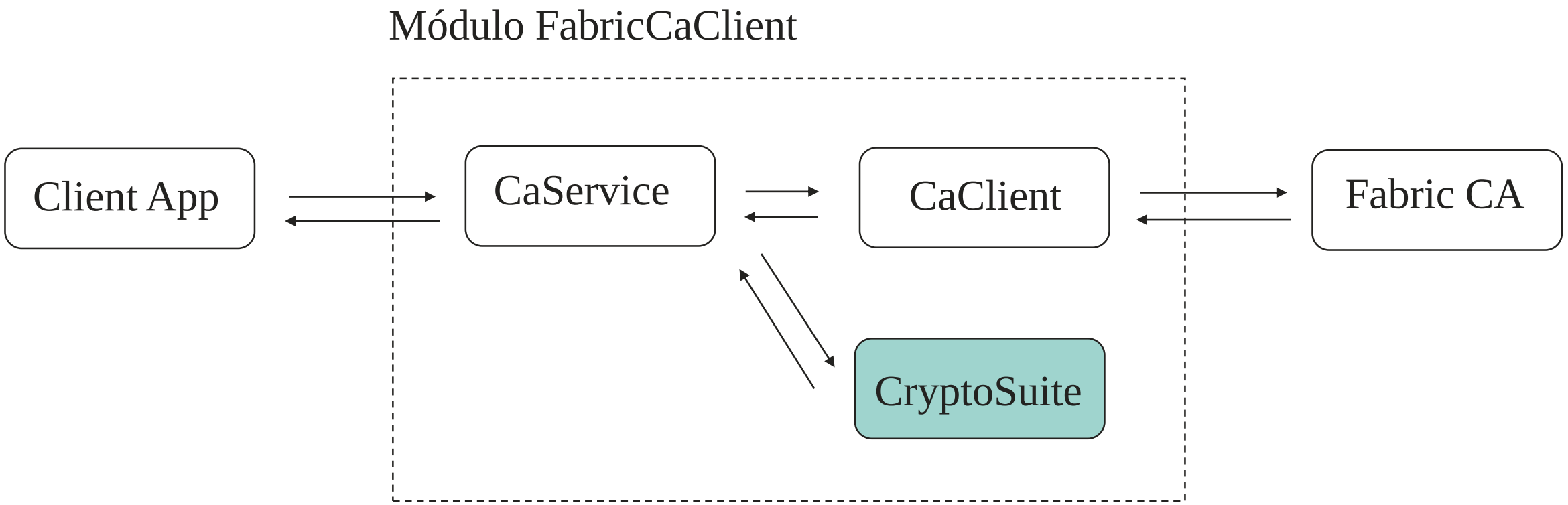
- Renovación de certificados
- X509Certificate2 (SystemSequrityCryptography) para acceder al Subject del previo Enrollment.
- Creación de llaves y CSR
- Especificación de atributos para el certificado
- Resultado: Enrollment con Cert, PrivateKey y CAChain
- Token de autenticación: Ecert + Firma sobre mensaje y certificado

# Módulo FabricCAClient

## CAService, CaClient. Revocación

- Revocación de:
  - Certificado: AKI + Serial Number
  - Identidad ( y certs relacionados): ID
- Justificación de la operación: Reason
- Indicación para generar o no CRL
- Token de autenticación: Ecert + Firma sobre mensaje y certificado

# Fabric SDK C#


Módulo FabricCaClient
Client App
CaService
CaClient
Fabric CA
CryptoSuite

# Módulo FabricCAClient

## ICryptoSuite y CryptoPrimitives

- GenrateKeyPair: Genera par de llaves (ECDSA)
- Sign: Firma un mensaje (algoritmo SHA256withECDSA)
- GenCSR: Genera una solicitud de certificación

- Implementación a partir de la biblioteca BouncyCastle

# Módulo FabricNetwork

## Wallet, WalletStore y FSWalletStore

- Get: Recuperar información sobre entidad
- Put: Almacenar
- Remove: Remover
- List: Listar entidades existentes

# Fabric SDK C#

FabricCaClient

CaService, CaClient → Comunicación con la CA para registro, incripción y revocación de identidades.

ICryptoSuite, CryptoPrimitives → Firmas digitales, generación de llaves criptográficas.

FabricNetwork

Wallet, WalletStore, FSWalletStore → Persistencia de datos. Acceso a identidades inscritas.

# Experimentación

```
[TestMethod()]
[DoNotParallelize]
0 references
public async Task CaClientFlowWithWalletTest() {
    string userId = "appUser";
    string userSecret = "usrPass";
    int maxEnrollment = 5;

    // initializing caService
    caService = new CAService(null, caEndpoint: caEndpoint, caName: caName, caCertsPath: caCertsPath);
    adminEnr = await caService.Enroll(registrarName, registrarSecret);

    // creating File System Wallet
    wallet = new Wallet(new FSWalletStore(storagePath));

    X509Identity identity = new X509Identity(adminEnr.Cert, adminEnr.PrivateKey, orgMSP);
    wallet.Put(registrarName, identity);
```

# Experimentación

```
// retrieve identity
var adminIdentity = wallet.Get(registrarName);
Enrollment newAdminEnr = new Enrollment(adminIdentity.GetPrivateKey(), adminIdentity.GetCertificate(),
                                                                                    null, caService);
// register user
string secret = await caService.Register(userId, userSecret, maxEnrollment, null, newAdminEnr);
// enroll user
Enrollment usrEnr = await caService.Enroll(userId, secret);
// revoke credentials
var result = await caService.Revoke(userId, "", "", "unspecified", true, newAdminEnr);
try {
    // check user is unable to enroll after its credentials are revoked
    Enrollment newUsrEnr = await caService.Enroll(userId, secret);
    wallet.Remove(registrarName);
}
catch (EnrollmentException exc) {
    StringAssert.Contains(exc.ToString(), unauthorizedMessage);
    wallet.Remove(registrarName);
    return;
}

Assert.Fail("Expected an enrollment denial.");
```

# Conclusiones

- Módulo FabricCAClient
  - CAService, CaClient
  - Funcionalidades de CryptoSuite
- Módulo FabricNetwork
  - Funcionalidades de Wallet
- Código robusto, extensible, bien comentado, fácil de usar
- Cumple normativas de Fabric para el desarrollo

# Recomendaciones

- Refactorizar código
- Extender clase CryptoPrimitives (métodos encrypt, decrypt, verify)
- Añadir soporte para Hadware Sequrity Module (HSM)
- Completar módulos restantes  (FabricNetwork, FabricCommon)

# Gracias

# Preguntas

# Preguntas

1. ¿Por qué en Hyperledger Fabric no tiene sentido usar algoritmos de consenso basados en recompensa como PoW o PoS?

# Preguntas

2. Al ser la CA de carácter jerárquico, que pasaría si se atacara la entidad principal (padre).

# Preguntas

3. ¿Por qué implementar un SDK para C# si al final Fabric CA tiene el binario  "fabric-ca-client" y con este cualquier desarrollador puede conectar su código C# e implementar una especie de CLI para interactuar con el servicio de la CA para gestionar los certificados, sin tener que usar el SDK? ¿Qué ventaja consideras  que brinda el SDK?

# Módulo fabric-ca-client del SDK de Fabric para gestionar identidades criptográficas con C#

Autor: Gabriela B. Martínez Giraldo

Tutores: Camilo Denis González
Daniel Mena Frias
Miguel Katrib Mora