Kafka Security

By default, Kafka communicates in **PLAINTEXT**, which means that all data is sent in the clear.

SSL in Kafka works for encrypting data between brokers and clients.

```
Kafka Client      ->  Encrypted data -> Kafka Brokers
(ssl-port:9093) (producer/consumer)
```

- ❑ SSL can be configured for encryption or authentication.

- ❑ We may configure just SSL encryption and independently choose a separate mechanism for client authentication, e.g. SSL, SASL, etc.

* Simple Authentication and Security Layer (SASL)

- SSL uses private-key/certificates pairs which are used during the SSL handshake process.

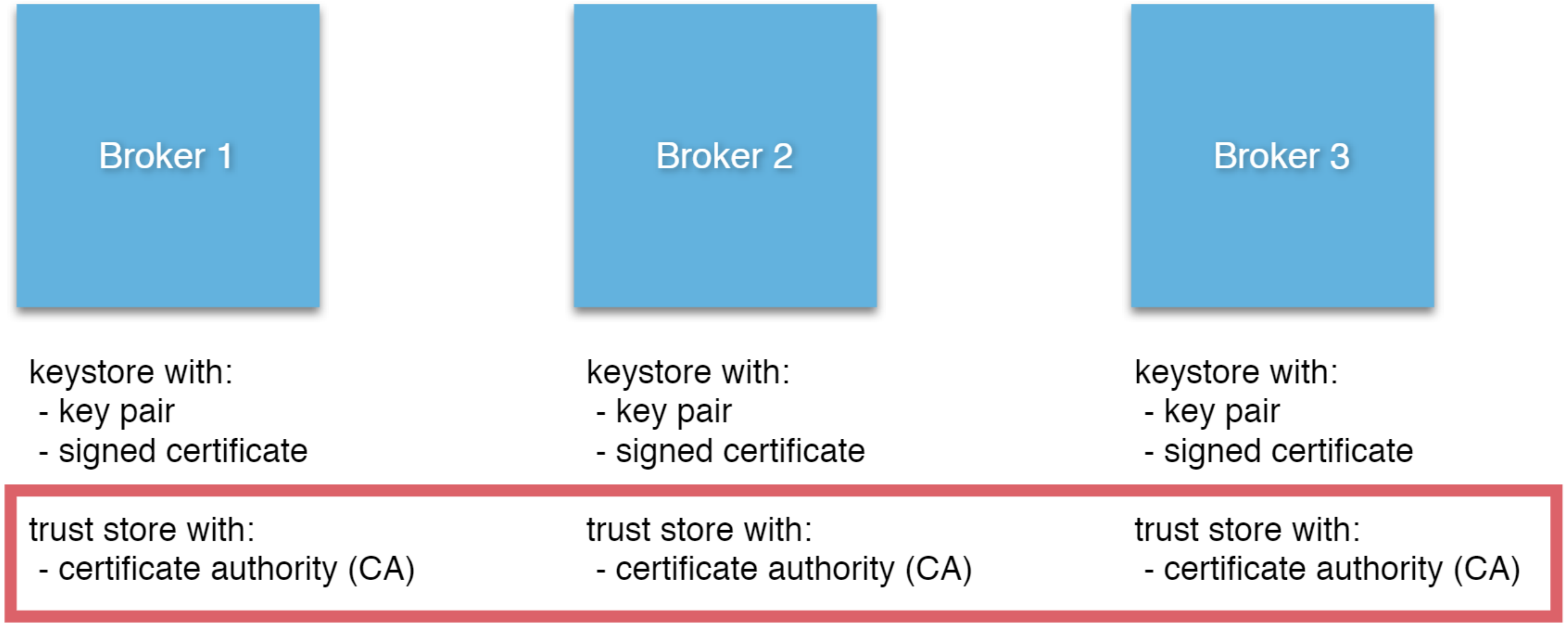
- each broker needs its own private-key/certificate pair, and the client uses the certificate to authenticate the broker.

- each logical client needs a private-key/certificate pair if client authentication is enabled, and the broker uses the certificate to authenticate the client.

Each broker and logical client can be configured with a truststore, which is used to determine which certificates (broker or logical client identities) to trust (authenticate). The truststore can be configured in many ways, consider the following two examples:

- the truststore contains one or many certificates: the broker or logical client will trust any certificate listed in the truststore
- the truststore contains a Certificate Authority (CA): the broker or logical client will trust any certificate that was signed by the CA in the truststore.

Single Trust Store Configuration



keystore with:
- key pair
- signed certificate

keystore with:
- key pair
- signed certificate

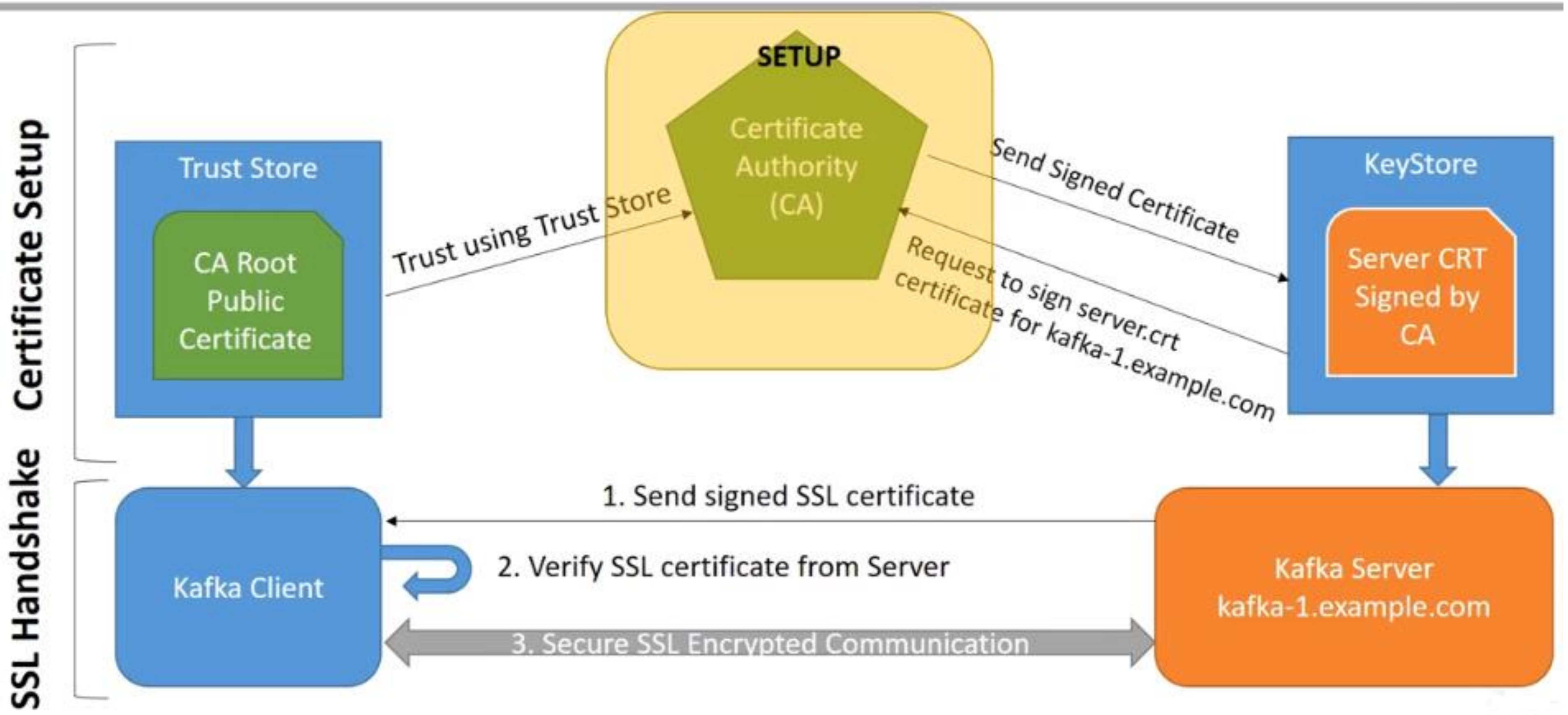
keystore with:
- key pair
- signed certificate

trust store with:
- certificate authority (CA)

trust store with:
- certificate authority (CA)

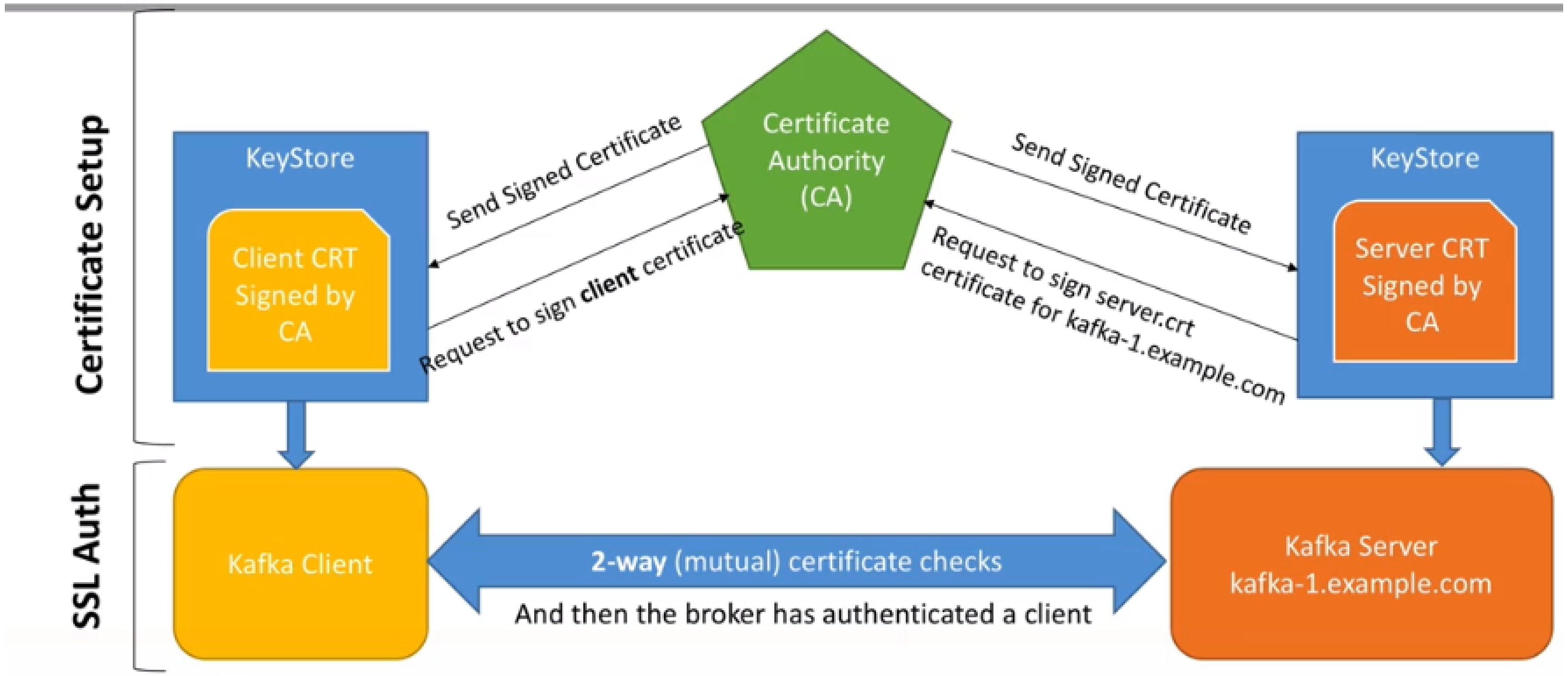
trust store with:
- certificate authority (CA)

Creating a CA (Certificate Authority)


Certificate Setup
SSL Handshake
SETUP
Certificate Authority (CA)
Trust Store
CA Root Public Certificate
Trust using Trust Store
Send Signed Certificate
Request to sign server.crt certificate for kafka-1.example.com
KeyStore
Server CRT Signed by CA
1. Send signed SSL certificate
2. Verify SSL certificate from Server
3. Secure SSL Encrypted Communication
Kafka Client
Kafka Server kafka-1.example.com

SSL Authentication

- ❑ Currently, only the servers have certificates.
- ❑ These certificates are used to enable encryption.
- ❑ With SSL, clients can also have certificates!!
- ❑ If the certificate is validated by the broker, the client is authenticated and has an identity.


Certificate Setup
SSL Auth
KeyStore
Client CRT Signed by CA
Certificate Authority (CA)
Send Signed Certificate
Request to sign **client** certificate
Send Signed Certificate
Request to sign server.crt certificate for kafka-1.example.com
KeyStore
Server CRT Signed by CA
Kafka Client
2-way (mutual) certificate checks
And then the broker has authenticated a client
Kafka Server kafka-1.example.com

In the encryption case:

Only the brokers has signed servers certificates.
The client were verifying the broker certificates to establish a SSL connection.
The client is "anonymous" to the broker (no identity)

In the authentication case:

The clients and the brokers have signed server certificates
The client and the brokers verify each other's certificates
The client now has an IDENTITY to the broker (we can apply ACL's)

Authorization in Kafka (using ACLs & SASL)

ACL (Access Control List) and SASL (Simple Authentication and Security Layer) are both security mechanisms in Apache Kafka, but they serve different purposes.

ACL is a mechanism that provides access control at the Kafka cluster and topic level. It allows Kafka administrators to define who can perform certain actions on a particular topic, such as read or write. ACLs are defined using the kafka-acls.sh script or by using the Kafka API.

SASL is a framework for authentication and security layer negotiation. It provides a way for clients and brokers to authenticate each other, and to negotiate the use of encryption for their communication. SASL is used to authenticate clients to brokers, and brokers to clients, using a variety of authentication mechanisms, such as Kerberos or OAuth.

In summary, ACL is used to control access to Kafka resources, while SASL is used to authenticate clients and brokers to each other.

What are ACL's?

Clients are authenticated to the brokers using either SSL or SASL.

ACLs (Access Control Lists) are used for authorization purpose.
It comes in three flavours:

Topics : restrict which client can read/write data
Consumer Groups : which client can use a specific consumer group
Cluster: which client can create/delete topics or apply settings

Note: "Super User" in Kafka can do everything without any special kind of ACL

ACLs are stored in Zookeeper, and added through the usage of the command line.

Therefore, we need to absolutely restrict who can access your Zookeeper Cluster (through security/network rules)

SASL

Simple Authentication and Security Layer

It's becoming a standard in big data applications for security (Hadoop, Kafka etc.,)

It does not change the application protocol and theoretically one SASL ticket can give you access to many systems

SASL is usually combined with SSL/TLS for adding an encryption layer

SASL in kafka currently implements the following protocols:

PLAIN (simple username/password)

SCRAM (modern username/password with challenge)

GSS-API (Kerberos authentication / Active Directory authentication)

What is SCRAM?

In cryptography, the Salted Challenge Response Authentication Mechanism (SCRAM) is a family of modern, password-based challenge–response authentication mechanisms providing authentication of a user to a server.

Kafka supports SCRAM-SHA-256 and SCRAM-SHA-512.

GSSAPI - Generic Security Service Application Program Interface also GSS-API).

SASL/GSSAPI is for organizations using Kerberos (for example, by using Active Directory)

Kerberos is a security protocol that provides an alternate mechanism for both client and server authentication. Kerberos authentication relies on a trusted third party called the KDC (Key Distribution Center). The Secure Shell protocol supports Kerberos authentication via GSSAPI (Generic Security Services Application Programming Interface).

Kerberos

It is a network authentication protocol. It is designed to provide strong authentication for client/server applications by using secret-key cryptography.

OAuth2

It is an authorization framework that enables applications to obtain limited access to user accounts on an HTTP service, such as Facebook, GitHub etc.,

confluent Identity and Access Management (IAM)

Description:

Use the confluent iam commands to manage RBAC and IAM permissions.

Manage Role Based Access (RBAC) and Identity and Access Management (IAM) permissions.

Usage:
confluent iam [command]