### Последовательность настройки двусторонней проверки между микросервисами и кластером Kafka

1. **Создание папки secrets в корневой директории проекта**:
   * Это сделает доступ к сертификатам удобным и организованным, особенно при использовании Docker и Docker Compose.
2. **Структура папки secrets**:
   * Для каждого брокера создайте отдельную подпапку, например, kafka1, kafka2, kafka3, а также для каждого микросервиса.
   * В каждой подпапке храните соответствующие \*.crt, \*.key, keystore.jks и truststore.jks для этого брокера или микросервиса (для некоторых микросервисов достаточно \*.crt, \*.key).

Если Вы настраиваете безопасность для продакшена, важно правильно управлять ключами и сертификатами, особенно корневым сертификатом (CA) и его приватным ключом. Использование CA в продакшене требует дополнительных мер безопасности, чтобы предотвратить утечку ключей и защитить всю инфраструктуру.

### Рекомендации по настройке безопасности на продакшене:

1. **Не храните приватный ключ CA (ca.key) в проекте**:
   * Приватный ключ CA — это самый важный компонент безопасности всей Вашей системы. Если он будет скомпрометирован, злоумышленники смогут создать любые сертификаты, которые будут восприниматься как доверенные, что приведет к серьезным угрозам безопасности.
   * **Используйте приватный ключ CA только в изолированной и защищенной среде** для создания и подписания сертификатов перед их загрузкой в проект.
2. **Создайте CA и подпишите все сертификаты в безопасной изолированной среде**:
   * Настройка CA и генерация всех сертификатов должна выполняться на отдельной, изолированной машине, которая не подключена к интернету или к основной инфраструктуре. Это снижает риск утечки приватного ключа CA.
   * После создания и подписания сертификатов с помощью CA, в проект переносятся ключи и сертификаты только микросервисов, а сам корневой сертификат (ca.crt) и промежуточные сертификаты (если Вы их создаете) не должен попадать в рабочую инфраструктуру.
3. **Настройка CA на выделенной машине**:
   * Если возможно, используйте специальное программное обеспечение для управления сертификатами, такое как HashiCorp Vault, или аппаратные средства, такие как HSM (Hardware Security Module), для создания и защиты CA.
4. **Использование промежуточных CA**:
   * Создайте промежуточный сертификат, подписанный Вашим корневым CA. Промежуточный сертификат, это сертификат, который будет использоваться для подписи всех сертификатов брокеров и микросервисов. Таким образом, если промежуточный сертификат скомпрометирован, корневой CA останется безопасным.
   * В продакшене используйте только промежуточный CA для подписания сертификатов, а корневой CA держите полностью изолированным.
5. **Ротация и отзыв сертификатов**:
   * В продакшене важно уметь оперативно заменять скомпрометированные сертификаты. Настройте механизмы ротации и отзыва сертификатов.
   * Используйте такие инструменты, как OCSP (Online Certificate Status Protocol) и CRL (Certificate Revocation List), чтобы клиенты могли проверять, не отозваны ли сертификаты.

### Структура папки secrets для продакшена

/test\_kafka  
├── docker-compose.yaml  
├── secrets│ │  
│ ├── kafka1 *# Сертификаты и ключи для брокера kafka1*│ │ ├── kafka.keystore.jks *# Keystore для kafka1 (содержит kafka1.key и kafka1.crt)*│ │ └── kafka.truststore.jks *# Truststore для kafka1 (содержит ca.crt и intermediate.crt)*│ │  
│ ├── kafka2 *# Сертификаты и ключи для брокера kafka2*  
│ │ ├── kafka.keystore.jks  
│ │ └── kafka.truststore.jks  
│ │  
│ ├── kafka3 *# Сертификаты и ключи для брокера kafka3*  
│ │ ├── kafka.keystore.jks  
│ │ └── kafka.truststore.jks  
│ │  
│ │  
│ ├── microservice1 *# Сертификаты и ключи для микросервиса 1*

│ │ ├── chain.crt *# Если он необходим*│ │ ├── microservice1.key *# Если он необходим*  
│ │ ├── microservice1.crt *# Если он необходим*  
│ │ ├── microservice1.keystore.jks *# Если он необходим*  
│ │ └── microservice1.truststore.jks *# Если он необходим*  
│ │  
│ ├── microservice2 *# Сертификаты и ключи для микросервиса 2*

│ │ ├── chain.crt *# Если он необходим*│ │ ├── microservice2.key *# Если он необходим*  
│ │ ├── microservice2.crt *# Если он необходим*  
│ │ ├── microservice2.keystore.jks *# Если он необходим*  
│ │ └── microservice2.truststore.jks *# Если он необходим*  
│ │  
│ └── ... *# Дополнительные микросервисы и компоненты по аналогии*

### Структура папки secrets для разработки (после проверки работоспособности проекта необходимо использовать сертификаты созданные по требованиям безопасности и использовать структуру проекта, описанную выше)

/test\_kafka  
├── docker-compose.yaml  
├── secrets  
│ ├── akhq│ │ └── akhq.keystore.jks

│ │ └── truststore.jks│ │

│ ├── ca │ │ └── ca.crt

│ │ └── ca.key│ │ └── ca.srl │ │ └── chain.crt│ │ └── intermediate.crt│ │ └── intermediate.csr│ │ └── intermediate.key│ │ └── intermediate.srl│ │ └── intermediate\_ext.cnf│ │ └── san1.cnf

│ │ └── san2.cnf│ │ └── san3.cnf│ │ └── san\_akhq.cnf│ │ └── san\_consumer.cnf│ │ └── san\_kafka\_exporter.cnf│ │ └── san\_producer.cnf│ │ └── san\_prometheus.cnf│ │

│ ├── consumer│ │ ├── chain.crt

│ │ ├── consumer.crt│ │ ├── consumer.key*)*

│ │   
│ ├── kafka1 │ │ ├── kafka.keystore.jks│ │ └── kafka.truststore.jks│ │  
│ ├── kafka2   
│ │ ├── kafka.keystore.jks  
│ │ └── kafka.truststore.jks  
│ │  
│ ├── kafka3   
│ │ ├── kafka.keystore.jks  
│ │ └── kafka.truststore.jks  
│ │  
│ ├── kafka\_exporter│ │ ├── chain.crt  
│ │ ├── kafka\_exporter.crt  
│ │ ├── kafka\_exporter.key  
│ │  
│ ├── producer│ │ ├── chain.crt

│ │ ├── producer.crt

│ │ ├── producer.key   
│ │  
│ ├── prometheus│ │ ├── chain.crt

│ │ ├── prometheus.crt  
│ │ ├── prometheus.key

Для настройки SSL/TLS в Вашем кластере Kafka и обеспечения защищенного подключения клиентов, таких как UI for Apache Kafka, Kafka Exporter, Prometheus и других микросервисов, необходимо выполнить несколько шагов по созданию сертификатов и конфигураций.

### Последовательность создания сертификатов и ключей для настройки SSL/TLS подключений:

1. **Создание корневого сертификата (ca)**.
2. **Создание промежуточного сертификата (intermediate)**
3. **Создание и подпись сертификатов для каждого брокера Kafka с использованием ca и intermediate**.
4. **Создание Truststore и Keystore для каждого брокера (микросервиса)**.

### Руководство по созданию необходимых сертификатов и их настройке (при разработке и тестировании проекта изолированной средой можно пренебречь):

### При разработке рекомендуется все (без исключения) сертификаты, ключи и конфигурационные файлы создавать в ./secrets/ca, после чего переместить сертификаты и ключи в соответствующие директории.

#### Шаг 1: **Создайте в изолированной среде корневой CA**:

Создайте корневой сертификат и ключ, который будет использоваться для подписи промежуточного сертификата и сертификатов брокеров и микросервисов:

*# Создаем приватный ключ для CA*openssl genpkey -algorithm RSA -out ca.key -pkeyopt rsa\_keygen\_bits:4096

*# Создаем самоподписанный корневой сертификат (CA)*openssl req -new -x509 -key ca.key -out ca.crt -days 365 -sha256 -subj "/CN=Production-CA"

#### Шаг 2: **Создайте в изолированной среде конфигурационный файл**:

В директории, в которой создаются ca и intermediate сертификаты создаем файл с именем intermediate\_ext.cnf. В файл прописываем: basicConstraints=critical,CA:TRUE,pathlen:0

#### Шаг 3: **Создайте в изолированной среде промежуточный CA для подписания сертификатов**:

*# Приватный ключ промежуточного CA* openssl genpkey -algorithm RSA -out intermediate.key -pkeyopt rsa\_keygen\_bits:4096

*# Запрос на сертификат для промежуточного CA* openssl req -new -key intermediate.key -out intermediate.csr -subj "/CN=Intermediate-CA" -sha256

*# Подписываем промежуточный CA корневым CA* openssl x509 -req -in intermediate.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out intermediate.crt -days 365 -sha256 -extfile intermediate\_ext.cnf

*# Создаем цепочку сертификатов для truststore*

(Get-Content intermediate.crt) + (Get-Content ca.crt) | Set-Content chain.crt

Проверка файла chain.crt с помощью OpenSSL, чтобы убедиться, что сертификаты читаются корректно:

openssl x509 -in chain1.crt -noout -text

При подписании сертификатов с использованием корневого или промежуточного сертификата создается файл ca.srl (серийный файл), и он служит для отслеживания серийных номеров сертификатов, которые подписываются данным CA. Серийный номер — это уникальный идентификатор каждого подписанного сертификата, который используется для управления и проверки подлинности.

Файл ca.srl должен храниться вместе с Вашим CA (корневым или промежуточным) в той же изолированной и защищенной среде, где хранятся приватные ключи CA (ca.key или intermediate.key). Это важно, потому что утрата или повреждение ca.srl может привести к проблемам с подписанием новых сертификатов или дублированию серийных номеров, что нарушает работу всей PKI (инфраструктуры публичных ключей). Как и приватный ключ CA, ca.srl не должен покидать изолированную машину или среду, где происходит генерация и подпись сертификатов. Этот файл не должен попадать в проект, так как его наличие там не нужно и может представлять риск. Обязательно создайте резервную копию файла ca.srl вместе с ключами и другими важными файлами CA. Если файл будет потерян или поврежден, у Вас могут возникнуть сложности с управлением серийными номерами подписываемых сертификатов.

#### Шаг 4: Используйте промежуточный CA для подписания сертификатов брокеров и клиентов:

*Для каждого брокера (например, kafka1, kafka2, kafka3), выполните следующую последовательность:*

В папке, в которой создаются сертификаты создаем файл с именем san1.cnf и содержимым (для каждого брокера свой файл, если kafka1 то san1.cnf, если kafka2 то san2.cnf):

[req]

distinguished\_name = req\_distinguished\_name

req\_extensions = v3\_req

prompt = no

[req\_distinguished\_name]

CN = kafka1

[v3\_req]

keyUsage = critical, digitalSignature, keyEncipherment

extendedKeyUsage = serverAuth, clientAuth

subjectAltName = @alt\_names

[alt\_names]

DNS.1 = kafka1

DNS.2 = kafka2

DNS.3 = kafka3

CN должен совпадать с «/CN=\*\*\*\*\*\*» который прописывается в командах openssl при создании сертификатов.

*Для каждого микросервиса (например, producer, consumer, kafka\_exporter и т.д.), выполните следующую последовательность:*

В папке, в которой создаются сертификаты создаем файл с именем san\_producer.cnf и содержимым (для каждого микросервиса свой файл, если consumer, то san\_consumer.cnf, если kafka\_exporter, то san\_kafka\_exporter.cnf и т.д.):

[req]

distinguished\_name = req\_distinguished\_name

req\_extensions = v3\_req

prompt = no

[req\_distinguished\_name]

CN = producer

[v3\_req]

keyUsage = critical, digitalSignature, keyEncipherment

extendedKeyUsage = serverAuth, clientAuth

subjectAltName = @alt\_names

[alt\_names]

DNS.1 = producer

CN должен совпадать с «/CN=\*\*\*\*\*\*» который прописывается в командах openssl при создании сертификатов.

После создания конфигурационного файла выполняем следующие команды:

*# Создаем приватный ключ для брокера kafka1* openssl genpkey -algorithm RSA -out kafka1.key -pkeyopt rsa\_keygen\_bits:4096

*# Создаем запрос на сертификат (CSR) для брокера kafka1* openssl req -new -key kafka1.key -out kafka1.csr -config san1.cnf -sha256

*# Подписываем сертификат брокера с использованием CA* openssl x509 -req -in kafka1.csr -CA intermediate.crt -CAkey intermediate.key -CAcreateserial -out kafka1.crt -days 365 -sha256 -extensions v3\_req -extfile san1.cnf

Проверка цепочки сертификатов

openssl verify -CAfile ../ca/ca.crt -untrusted ../ca/intermediate.crt kafka1.crt

*Создание Keystore для брокера kafka1:*

Необходимо установить более сложный пароль чем «changeit»

*# Конвертируем приватный ключ и сертификат брокера в P12 формат* openssl pkcs12 -export -in kafka1.crt -inkey kafka1.key -out kafka1.p12 -name kafka1 -CAfile chain.crt -caname chain -password pass:changeit

**Пароль для P12 (PKCS12) формата**:

* Параметр -password pass:changeit используется при создании P12 файла с помощью openssl pkcs12 -export.
* Этот пароль защищает P12 файл, который содержит приватный ключ и сертификат. Он используется для предотвращения несанкционированного доступа к ключу в этом контейнере.
* Вы можете установить любой пароль на этом шаге, но важно его запомнить, так как он потребуется при импорте в Java Keystore (JKS).

*# Импортируем P12 в JKS (Java Keystore)* keytool -importkeystore -deststorepass changeit -destkeypass changeit -destkeystore kafka1.keystore.jks -srckeystore kafka1.p12 -srcstoretype PKCS12 -srcstorepass changeit -alias kafka1

**Пароли для Java Keystore (JKS)**:

* -deststorepass changeit: пароль для всего Keystore (JKS), который защищает контейнер JKS.
* -destkeypass changeit: пароль для конкретного ключа внутри JKS.
* -srcstorepass changeit: пароль, который был установлен на предыдущем шаге для P12 файла.

После создания kafka1.keystore.jks необходимо переименовать файл в kafka.keystore.jks и переместить в соответствующую директорию (обязательное условие при использовании образа kafka от bitnami!!!!

Повторите эти шаги для kafka2 и kafka3, заменяя соответствующие имена (например, kafka1 на kafka2).

Создание truststore один раз для всех брокеров и микросервисов (для микросервисов нужно скопировать этот файл и удалить из названия “kafka”):

*# Создаем truststore и импортируем корневой и промежуточный сертификаты* keytool -keystore kafka.truststore.jks -alias CARoot -import -file ca.crt -storepass changeit -noprompt  
   
keytool -keystore kafka.truststore.jks -alias IntermediateCA -import -file intermediate.crt -storepass changeit -noprompt

**Пароль для Truststore**:

* Параметр -storepass changeit используется для защиты Truststore (JKS), который хранит сертификаты доверенных CA.
* Этот пароль защищает Truststore и его содержимое от несанкционированного доступа.

После создания сертификатов файлы \*.p12 и \*.csr удаляем.

Повторите эти шаги для всех микросервисов, заменяя соответствующие имена (например, kafka1 на producer).

**Прочее:**

Используется для проверки сертификата брокера с клиентской стороны. Она позволяет убедиться, что клиент (продюсер) может установить соединение с брокером и что сертификат брокера валидируется с использованием цепочки сертификатов, указанной в chain.crt

openssl s\_client -connect host.docker.internal:9092 -CAfile C:/Users/Lucky/RustroverProjects/test\_kafka/secrets/producer/chain.crt -showcerts

Для проверки, что сертификат клиента корректно воспринимается брокером, можно использовать следующую команду:

openssl s\_client -connect host.docker.internal:9092 -cert C:/Users/Lucky/RustroverProjects/test\_kafka/secrets/producer/producer.crt -key C:/Users/Lucky/RustroverProjects/test\_kafka/secrets/producer/producer.key -CAfile C:/Users/Lucky/RustroverProjects/test\_kafka/secrets/producer/chain.crt -showcerts

Настройка при включении фичи ssl в rdkafka:

1. переходим в корень диска С
2. git clone <https://github.com/microsoft/vcpkg.git>
3. cd vcpkg
4. .\bootstrap-vcpkg.bat
5. vcpkg install openssl:x64-windows-static
6. vcpkg integrate install
7. setx OPENSSL\_ROOT\_DIR "C:\vcpkg\installed\x64-windows-static"

setx OPENSSL\_LIB\_DIR "C:\vcpkg\installed\x64-windows-static\lib"

setx OPENSSL\_INCLUDE\_DIR "C:\vcpkg\installed\x64-windows-static\include"