

**Общество с ограниченной ответственностью
«Цифровые технологии»**



**Руководство пользователя по установке и настройке
программного обеспечения «КриптоАРМ Gate»**

Листов 11

г. Йошкар-Ола
2026 г.

Оглавление

- Назначение
 - Условия применения
 - Требования для установки и настройки
 - Используемый стек технологий
 - Установка
 - Установка Docker и Docker Compose
 - Установка и первый запуск
 - Сборка на базе Astra Linux
 - Настройка
 - Сертификаты
 - Режим mTLS и OIDC
 - Переменные mTLS и заголовки для бэкенда
 - Проксирование на бэкенд
 - Проксирование на HTTPS с клиентским сертификатом
 - Проверка работы
-

Назначение

КриптоАРМ GATE — шлюз, который:

- Работает в **режиме прокси**: принимает HTTPS-трафик и проксирует запросы на внутренний бэкенд (в т.ч. OIDC-провайдер).
- Обеспечивает **аутентификацию по ГОСТ-сертификату** ([ГОСТ Р 34.10–2012](#)): взаимная аутентификация (mTLS) и аутентификация через OIDC выполняются на основе ГОСТ-сертификата клиента.
- Поддерживает **двойное серверное TLS (ГОСТ + RSA)** при необходимости: один и тот же веб-ресурс может быть опубликован с двумя серверными сертификатами — ГОСТ и RSA — на одном порту. Это нужно, когда к шлюзу подключаются разные категории клиентов: те, кто поддерживает только ГОСТ (типовой случай для квалифицированной ЭП), и те, кому нужен TLS по RSA (например, устаревшие браузеры или системы без поддержки ГОСТ). В таком режиме RSA используется **только для**

установления защищённого канала к одному и тому же ресурсу; аутентификация пользователя по-прежнему выполняется по ГОСТ-сертификату при mTLS/OIDC.

- Интегрируется с модулем **КриптоАРМ OIDC-mTLS** для проверки **квалифицированности** предъявленного ГОСТ-сертификата перед допуском пользователя.

Типовой сценарий: пользователь подключается к GATE по HTTPS (по ГОСТ или, при включённом двойном TLS, по RSA), для входа предъявляет **квалифицированный ГОСТ-сертификат**; GATE проверяет его (при необходимости через модуль КриптоАРМ OIDC-mTLS) и передаёт атрибуты сертификата бэкенду для OIDC-входа по сертификату.

Условия применения

Требования для установки и настройки

Характеристика	Минимальное значение	Дополнение
ОЗУ	4 ГБ	
HDD	10 ГБ	
Процессор	x86_64 (amd64)	
Поддерживаемые ОС	Ubuntu 18.04/20.04, Debian 11, Astra Linux SE 1.8 Опел	Поддерживаются другие Linux-дистрибутивы с поддержкой Docker
Дополнительно	Docker	Средство автоматизации развертывания и контейнеризации

Используемый стек технологий

Apache 2.4, Docker; КриптоПро CSP 5.0 (mod_ssl) устанавливается отдельным слоем при развёртывании. Опционально: модуль КриптоАРМ OIDC-mTLS (для проверки квалифицированности сертификата).

Установка

Установка Docker и Docker Compose

Для запуска проекта необходимы **Docker** и **Docker Compose**.

Инструкции: [официальная документация Docker](#).

Минимальные версии:

- Docker Engine: **20.10** или выше (для работы Docker Compose v2)
- Docker Compose: **2.x** (плагин `docker compose` или отдельная установка)

Установка и первый запуск

1. Создайте и перейдите в рабочую директорию:

```
mkdir cryptoarm-gate && cd cryptoarm-gate
```

2. Загрузите файлы конфигурации:

```
curl -O https://git.digtlab.ru/trusted/cryptoarm/gate/-/raw/main/docker-com
curl -O https://git.digtlab.ru/trusted/cryptoarm/gate/-/raw/main/Dockerfile
curl -O https://git.digtlab.ru/trusted/cryptoarm/gate/-/raw/main/.env
```

3. Создайте каталоги и конфигурацию SSL:

```
mkdir -p cryptopro certs/gost certs/rsa certs/root conf html logs cgi-bin
curl -o conf/ssl.conf https://git.digtlab.ru/trusted/cryptoarm/gate/-/raw/m
```

4. Скачайте актуальную версию [КриптоПро CSP 5.0 для Linux](#) (x64, deb) и поместите архив `linux-amd64_deb.tgz` в каталог `cryptopro/`. Для скачивания требуется авторизация на сайте КриптоПро.

5. Отредактируйте настройки в файле `.env`:

- `LICENSE` — лицензия КриптоАРМ GATE (обязательна; при неверной лицензии контейнер не стартует).
- `CRYPTO_PRO_LICENSE` — серверная лицензия КриптоПро CSP (при необходимости).
- При использовании бэкенда укажите его адрес в конфигурации Apache (см. [Проксирование на бэкенд](#)).

6. Разместите сертификаты сервера в соответствующих каталогах (см. [Сертификаты](#); для теста можно скачать примеры из репозитория командами `curl`, указанными в этом разделе).

7. Запустите сборку и сервис:

```
docker compose build --pull
docker compose up -d
```

Дополнительно:

- Просмотр логов: `docker compose logs -f`
- Остановка: `docker compose stop`
- Обновление: скачайте актуальные файлы из `docker/`, затем `docker compose build --pull` и `docker compose up -d`

Сборка на базе Astra Linux

1. В каталоге развёртывания загрузите Dockerfile для Astra Linux:

```
curl -O https://git.digtlab.ru/trusted/cryptoarm/gate/-/raw/main/Dockerfile
mv Dockerfile.astralinux Dockerfile
```

2. Далее — как в разделе [Установка и первый запуск](#) (пункт с `cryptopro/` и `docker compose build`).

Базовый образ: `registry.digtlab.ru/trusted/cryptoarm/gate:astralinux` (без CSP; CSP ставится слоем из `Dockerfile.astralinux`).

Настройка

Сертификаты

- **ГОСТ (обязательно):** серверные сертификаты в `certs/gost/` (например, `localhost.cer`, `localhost.pfx`). Используются для HTTPS и для приёма клиентских ГОСТ-сертификатов при mTLS. Сюда же кладут PFX сертификата, которым GATE сам ходит на внешний HTTPS (исходящий mTLS через `gost-proxy.cgi`); в `SSLCertificateFile` указывайте только серверный сертификат GATE.
- **RSA (опционально):** серверные сертификаты в `certs/rsa/` (например, `localhost.cer`, `localhost.pfx`). Подключаются только если нужно поднять один веб-ресурс одновременно по ГОСТ и по RSA (совместимость с клиентами без поддержки ГОСТ). Вход по сертификату и mTLS при этом выполняются только по ГОСТ.
- **Корневые и доверенные СА:** в `certs/root/` (файлы `.cer`) — для проверки клиентских ГОСТ-сертификатов при mTLS. При старте контейнера устанавливаются в хранилище КриптоПро.

В качестве примеров для тестирования можно скачать сертификаты из репозитория:

```
mkdir -p certs/gost certs/rsa certs/root
curl -o certs/gost/localhost.cer https://git.digtlab.ru/trusted/cryptoarm/g
curl -o certs/gost/localhost.pfx https://git.digtlab.ru/trusted/cryptoarm/g
curl -o certs/rsa/localhost.cer https://git.digtlab.ru/trusted/cryptoarm/ga
curl -o certs/rsa/localhost.pfx https://git.digtlab.ru/trusted/cryptoarm/ga
curl -o certs/root/cryptopro_test.cer https://git.digtlab.ru/trusted/crypto
```

Внимание: примеры предназначены только для проверки развёртывания и тестовых сценариев. В продуктивной среде используйте собственные сертификаты и корневые СА.

Конфигурация SSL задаётся в `conf/ssl.conf`, монтируемом в контейнер.

Режим mTLS и OIDC

Вход по сертификату и допуск к защищённым ресурсам выполняются по **ГОСТ-сертификату** клиента. В конфигурации Apache включают проверку клиентского сертификата и передачу его данных бэкенду:

- В `conf/ssl.conf` раскомментируйте `SSLVerifyClient require` при необходимости обязательной проверки клиентского сертификата (принимаются только ГОСТ-сертификаты, настроенные в КриптоПро).
- Раскомментируйте блоки `RequestHeader set X-SSL-Client-*`, чтобы передавать в бэкенд атрибуты предъявленного сертификата:
 - `X-SSL-Client-Verify`, `X-SSL-Client-DN`, `X-SSL-Client-Serial`, `X-SSL-Client-Issuer`, `X-SSL-Client-Cert`, `X-SSL-Client-S-DN-CN`, `X-SSL-Client-I-DN-CN`.

Проверка **квалифицированности** предъявленного ГОСТ-сертификата выполняется с привлечением **модуля КриптоАРМ OIDC-mTLS: GATE** передаёт данные подписи/сертификата в API модуля; по результату проверки принимается решение о допуске пользователя для OIDC-входа.

Переменные mTLS и заголовки для бэкенда

При успешной проверке клиентского ГОСТ-сертификата Apache (`mod_ssl`) устанавливает переменные окружения `SSL_CLIENT_*` и при включённых `RequestHeader` передаёт их бэкенду в HTTP-заголовках `X-SSL-Client-*`. Ниже — пример значений (из SSI, например `index.shtml`).

Переменные сертификата клиента (mTLS):

Переменная / заголовок	Описание	Пример значения
SSL_CLIENT_VERIFY / X-SSL-Client-Verify	Результат проверки клиентского сертификата	SUCCESS
SSL_CLIENT_S_DN_CN / X-SSL-Client-S-DN-CN	Common Name субъекта сертификата	TEST_КРИПТОАРМ
SSL_CLIENT_I_DN_CN / X-SSL-Client-I-DN-CN	Common Name издателя (УЦ)	Тестовый УЦ 000 "КРИПТО-ПРО"
SSL_CLIENT_S_DN / X-SSL-Client-DN	Distinguished Name субъекта	CN=TEST_КРИПТОАРМ
SSL_CLIENT_I_DN / X-SSL-Client-Issuer	DN издателя	CN=Тестовый УЦ 000 "КРИПТО-ПРО",O=000 "КРИПТО-ПРО",L=Москва,ST=г. Москва,C=RU,...
SSL_CLIENT_M_SERIAL / X-SSL-Client-Serial	Серийный номер сертификата	7C002E6AAA722B2E8F1A3D81930012002E6AAA
SSL_CLIENT_CERT / X-SSL-Client-Cert	Сертификат клиента (PEM)	-----BEGIN CERTIFICATE----- ... ----- END CERTIFICATE-----
SSL_CLIENT_A_KEY	Алгоритм ключа	GOST R 34.10-2012 with 256 bit modulus
SSL_CLIENT_A_SIG	Алгоритм подписи	GOST R 34.10-2012 with GOST R 34.11-2012 (256 bit)
SSL_CLIENT_V_START / SSL_CLIENT_V_END	Срок действия сертификата	Feb 13 08:58:40 2026 GMT ... Apr 13 09:08:40 2026 GMT

Дополнительные переменные SSL (только на стороне GATE, в SSI/скриптах):

```
HTTPS=on
SSL_PROTOCOL=TLSv1.2
SSL_CLIENT_I_DN_C=RU
SSL_CLIENT_I_DN_ST=г. Москва
SSL_CLIENT_I_DN_L=Москва
SSL_CLIENT_I_DN_O=ООО "КРИПТО-ПРО"
SSL_CLIENT_I_DN_CN=Тестовый УЦ ООО "КРИПТО-ПРО"
SSL_VERSION_INTERFACE=mod_ssl/2.4.41
SSL_VERSION_LIBRARY=OpenSSL/1.1.1f
```

Бэкенд (OIDC-провайдер и др.) при проксировании получает заголовки `HTTP_X_SSL_CLIENT_VERIFY`, `HTTP_X_SSL_CLIENT_DN`, `HTTP_X_SSL_CLIENT_SERIAL`, `HTTP_X_SSL_CLIENT_ISSUER`, `HTTP_X_SSL_CLIENT_CERT`, `HTTP_X_SSL_CLIENT_S_DN_CN`, `HTTP_X_SSL_CLIENT_I_DN_CN` (префикс `HTTP_` и замена `-` на `_` задаются правилами сервера/приложения).

Проксирование на бэкенд

В `conf/ssl.conf` задаётся проксирование на внутренний бэкенд (в т.ч. OIDC-провайдер):

```
ProxyPass          /api http://backend/
ProxyPassReverse   /api http://backend/
```

Измените `http://backend/` на фактический адрес бэкенда (в Docker Compose — имя сервиса и порт, например `http://cryptoarm-id-back:3005/`).

Пример бэкенда для проверки. В `docker-compose.yml` можно использовать тестовый бэкенд [kennethreitz/httpbin](https://kennethreitz.com/httpbin/): он возвращает в ответе все заголовки запроса, в том числе переданные GATE заголовки mTLS (`X-SSL-Client-*`). Сервис уже описан в составе проекта:

```
backend:
  image: kennethreitz/httpbin
  container_name: backend
```

После запуска `docker compose up -d` запрос к `https://SERVER:4443/api/headers` (с предъявлением ГОСТ-сертификата при включённом mTLS) вернёт JSON с заголовками, среди которых

будут `X-Ssl-Client-Verify` , `X-Ssl-Client-Dn` , `X-Ssl-Client-Serial` , `X-Ssl-Client-Issuer` , `X-Ssl-Client-Cert` , `X-Ssl-Client-S-Dn-Cn` , `X-Ssl-Client-I-Dn-Cn` и др.

Проксирование на HTTPS с клиентским сертификатом

Задача: GATE принимает обычный HTTP(S)-запрос и сам ходит на внешний HTTPS-сервер, предъявляя ГОСТ-сертификат из хранилища КриптоПро (аналог `curl --cert <отпечаток>`).

Входящий mTLS (`SSLVerifyClient`) и исходящий клиентский сертификат независимы: здесь речь только о сертификате, которым GATE аутентифицируется у внешнего сервиса.

Реализация: CGI `cgi-bin/gost-proxy.cgi` вызывает CryptoPro curl с `--cert` . Пример конфига Apache: `conf/ssl.gost-proxy.conf.example` .

Схема:

```
Клиент → GATE (/ext/...) → gost-proxy.cgi → cprocsp curl --cert <SHA1> → ht
```

1. Сертификат в хранилище `www-data`

CGI выполняется от `www-data` . Положите PFX клиентского сертификата в `certs/gost/` (без пароля на контейнер ключа). При старте `entrypoint.sh` устанавливает все `certs/gost/*.pfx` в `uMy` пользователя `www-data` .

Проверка внутри контейнера (подставьте свой отпечаток и URL):

```
su -s /bin/sh -c "/opt/cprocsp/bin/amd64/certmgr -list -store uMy" www-data

su -s /bin/sh -c "/opt/cprocsp/bin/amd64/curl --verbose \
--cert <SHA1_ОТПЕЧАТОК> \
--header 'Accept: application/json' \
'https://example.gost.ru/api/...'" www-data
```

Если curl от `www-data` не проходит, прокси через CGI тоже не сможет.

2. Переменные в `.env`

```
CSP_PROXY_UPSTREAM=https://example.gost.ru
CSP_PROXY_CERT_THUMBPRINT=<SHA1_ОТПЕЧАТОК>
# опционально:
# CSP_PROXY_CONNECT_TIMEOUT=15
# CSP_PROXY_MAX_TIME=30
```

Переменная	Назначение
<code>CSP_PROXY_UPSTREAM</code>	Базовый URL внешнего сервиса (без завершающего <code>/</code>)
<code>CSP_PROXY_CERT_THUMBPRINT</code>	SHA1 отпечаток, как у <code>curl --cert</code>
<code>CSP_PROXY_CONNECT_TIMEOUT</code>	Таймаут соединения, сек (по умолчанию 15)
<code>CSP_PROXY_MAX_TIME</code>	Общий таймаут запроса, сек (по умолчанию 30)

3. Маршрут в `conf/ssl.conf`

Префикс URL произвольный (`/ext/`, `/partner/`, ...). Путь после префикса дописывается к `CSP_PROXY_UPSTREAM`.

```

PassEnv CSP_PROXY_UPSTREAM CSP_PROXY_CERT_THUMBPRINT CSP_PROXY_CONNECT_TIME

<VirtualHost *:443>
    SSLEngine on
    SSLCertificateFile /certs/gost/localhost.cer
    SSLCertificateFile /certs/rsa/localhost.cer

    <Directory /var/www/cgi-bin>
        Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
        Require all granted
        SetHandler cgi-script
    </Directory>

    ProxyPreserveHost On
    ProxyPass          /api http://backend/
    ProxyPassReverse   /api http://backend/

    # https://GATE/ext/api/v1/foo → ${CSP_PROXY_UPSTREAM}/api/v1/foo
    ScriptAliasMatch   ^/ext(/.*)?$ /var/www/cgi-bin/gost-proxy.cgi$1
</VirtualHost>

```

Не убирайте `SSLEngine` и серверные `SSLCertificateFile` — это HTTPS самого GATE.

```

docker compose build
docker compose up -d

```

4. Проверка

```
curl -k -H 'Accept: application/json' 'https://SERVER:4443/ext/api/v1/foo'
```

Должен вернуться ответ внешнего сервиса — тот же, что у `curl --cert` от `www-data`.

Симптом	Что проверить
<code>500</code> и <code>CSP_PROXY_* is not set</code>	переменные в <code>.env</code> , есть <code>PassEnv</code> , контейнер перезапущен
<code>502</code> / пустой ответ	<code>logs/error.log</code> ; отпечаток; PFX у <code>www-data</code> ; доступность upstream
<code>curl</code> от <code>www-data</code> с <code>--cert</code> не работает	сначала починить установку сертификата, затем CGI

Проверка работы

- Доступ по HTTPS (порт 4443 по умолчанию): `https://SERVER:4443`
- Статическая раздача из `/var/www/html` (монтируется из `html/`).
- Проксирование запросов по путям, указанным в `ProxyPass` (например, `/api`), на бэкенд.
- Исходящий HTTPS с ГОСТ-сертификатом:
`https://SERVER:4443/ext/...` (см. [Проксирование на HTTPS с клиентским сертификатом](#)).

Проверка mTLS и передачи данных сертификата бэкенду.

Подключитесь к GATE с клиентским ГОСТ-сертификатом и выполните запрос к примеру бэкенда (httpbin):

```
https://SERVER:4443/api/headers
```

В ответе (JSON) в объекте `headers` будут переданные GATE заголовки, например:

```
{
  "headers": {
    "Host": "172.17.1.115:14443",
    "X-Forwarded-Host": "172.17.1.115:14443",
    "X-Ssl-Client-Verify": "SUCCESS",
    "X-Ssl-Client-Dn": "CN=TEST_КРИПТОАРМ",
    "X-Ssl-Client-S-Dn-Cn": "TEST_КРИПТОАРМ",
    "X-Ssl-Client-I-Dn-Cn": "Тестовый УЦ 000 \"КРИПТО-ПРО\"",
    "X-Ssl-Client-Serial": "7C002E6AAA722B2E8F1A3D81930012002E6AAA",
    "X-Ssl-Client-Issuer": "CN=Тестовый УЦ 000 \"КРИПТО-ПРО\",O=000 \"КРИПТО-ПРО\"",
    "X-Ssl-Client-Cert": "-----BEGIN CERTIFICATE----- MIIIE5DCCBJGgAwIBAgITf
  }
}
```

По наличию и значениям этих заголовков можно убедиться, что бэкенд получает данные клиентского сертификата для OIDC-входа по сертификату.