Использование ГОСТ-сертификатов в Nginx

Использование ГОСТ-сертификатов в Nginx (НАЙС.ОС)

Документ устанавливает порядок генерации ГОСТ-ключей/сертификатов, настройки Nginx для TLS на ГОСТ-алгоритмах, а также порядок применения корневых сертификатов Минцифры/ЕСИА-цепочек в доверенных хранилищах.

1. Область применения

Настоящий документ применяется для серверных узлов НАЙС.ОС, где требуется:

- TLS-терминация в Nginx с использованием ГОСТ-алгоритмов (ГОСТ Р 34.10-2012, ГОСТ Р 34.11-2012, ГОСТ 34.12-2015 и др. по профилю).
- Тестирование TLS-сеансов ГОСТ-клиентами на базе OpenSSL.
- Установка/использование корневых сертификатов, распространяемых через инфраструктуру государственных сервисов.

Примечание: поддержка ГОСТ-cipher suites в OpenSSL активируется при наличии ГОСТ-совместимого движка (engine). Исторически ГОСТ-engine удалён из состава OpenSSL и поддерживается внешним проектом.

2. Нормативные и справочные ссылки

- RFC 9189 — GOST cipher suites для TLS (актуально для профилей TLS с ГОСТ-алгоритмами).
- RFC 9367 — дополнительные определения/профили GOST cipher suites для TLS (см. конкретную редакцию RFC).
- Документация Nginx: директивы TLS (ssl_certificate, ssl_certificate_key, ssl_protocols, ssl_ciphers, ssl_conf_command).

- Инструкция по корневым сертификатам (ссылки на источники/файлы russian_trusted_root_ca* и др.).

3. Термины и сокращения

- **ГОСТ-TLS** — набор алгоритмов и cipher suites, использующих ГОСТ-криптографию в рамках TLS.
- **Engine** — подключаемый криптомодуль OpenSSL, добавляющий алгоритмы/примитивы и/или поддержку ГОСТ-cipher suites.
- **Streebog** — ГОСТ Р 34.11-2012 (256/512).
- **Параметр-сет (paramset)** — выбор набора параметров эллиптических кривых для ГОСТ Р 34.10-2012 (пример: A).

Справочно: в OpenSSL строковые селекторы cipher suites содержат ключевые слова `aGOST`, `kGOST` и др.; они требуют активированного ГОСТ-движка.

4. Предварительная проверка (НАЙС.ОС)

До настройки Nginx требуется подтвердить наличие ГОСТ-алгоритмов в OpenSSL, используемом системой. В НАЙС.ОС предполагается, что ГОСТ-шифрование работоспособно «из коробки» (engine установлен и включён системной конфигурацией OpenSSL).

4.1. Проверка OpenSSL и доступных алгоритмов

```
openssl version -a

# Список доступных хеш-алгоритмов и поиск ГОСТ-дайджестов
openssl list -digest-algorithms | grep -i gost || true

# Список доступных алгоритмов ключей и поиск ГОСТ-ключей
openssl list -public-key-algorithms | grep -i gost || true

# Проверка, что движки видны (ожидается наличие gost/аналогичного)
openssl engine -c
```

4.2. Проверка наличия ГОСТ-cipher suites

```
# Показать cipher suites, содержащие GOST (включая новые/старые наименования)
openssl ciphers -v | grep -i gost || true
```

```
# Для TLS 1.2: выборка по ключевым словам OpenSSL (при наличии движка)
openssl ciphers -v -tls1_2 'kGOST' || true
openssl ciphers -v -tls1_2 'aGOST' || true
```

Предупреждение: если команды `openssl ciphers` не показывают ГОСТ-наборы, сначала устраняется причина на уровне OpenSSL (engine/провайдер/openssl.cnf). Без этого настройка Nginx результата не даст. В OpenSSL поддержка ГОСТ-cipher suites не активируется «сама по себе».

5. Самоподписанный ГОСТ-сертификат для Nginx

Для стендов и первичной отладки допускается выпуск самоподписанного сертификата. Рекомендуемый профиль: ГОСТ Р 34.10-2012-256 + ГОСТ Р 34.11-2012-256 (Streebog-256).

Справочно по именам алгоритмов: в ГОСТ-engine/интеграциях OpenSSL широко используются идентификаторы `gost2012_256` и дайджест `md_gost12_256`, а также параметр-сет `A`.

5.1. Генерация ключа ГОСТ Р 34.10-2012-256

```
install -d -m 700 /etc/nginx/tls-gost
cd /etc/nginx/tls-gost

# Закрытый ключ (paramset A — типовой выбор для 2012-256)
openssl genpkey \
  -algorithm gost2012_256 \
  -pkeyopt paramset:A \
  -out server-gost.key

chmod 600 server-gost.key
```

5.2. Выпуск самоподписанного сертификата X.509

```
# Пример subject — замените на свой домен/организацию
SUBJ="/C=RU/ST=Moscow/L=Moscow/O=NiceSOFT/OU=IT/CN=gost.example.local"

openssl req -new -x509 \
  -key server-gost.key \
  -out server-gost.crt \
  -days 365 \
  -md_gost12_256 \
```

```
  -subj "$SUBJ"

# Контроль: поля сертификата
openssl x509 -in server-gost.crt -noout -text | sed -n '1,120p'
openssl x509 -in server-gost.crt -noout -subject -issuer -serial -fingerprint -sha256
```

Предупреждение: самоподписанный сертификат не формирует доверенную цепочку. Для продуктивного контура используются сертификаты от доверенного УЦ, а также корректное размещение корневых/промежуточных сертификатов в trust store.

6. Настройка Nginx для ГОСТ-TLS

Nginx использует библиотеку OpenSSL, с которой он собран/связан. Требуется, чтобы Nginx был собран с OpenSSL, где доступны ГОСТ-алгоритмы и ГОСТ-cipher suites, и чтобы OpenSSL-конфигурация на узле активировала движок.

6.1. Контроль сборки Nginx и версии OpenSSL

```
nginx -V 2>&1 | sed -n '1,200p'

# При необходимости: убедиться, что nginx линкуется с нужной libssl
ldd "$(command -v nginx)" | egrep 'libssl|libcrypto' || true
```

6.2. Базовый серверный блок (только ГОСТ, TLS 1.2)

Ниже приведён пример конфигурации для выделенного ГОСТ-порта. Это практичный подход: обычный HTTPS для браузеров остаётся на 443, ГОСТ-TLS публикуется на отдельном порту (например, 8443) для специализированных клиентов.

```
# /etc/nginx/conf.d/gost-tls.conf

server {
  listen 8443 ssl;
  server_name gost.example.local;

  # Сертификат и ключ ГОСТ
  ssl_certificate     /etc/nginx/tls-gost/server-gost.crt;
  ssl_certificate_key /etc/nginx/tls-gost/server-gost.key;

  # Практика: ограничить протокол для профиля ГОСТ (как минимум TLSv1.2)
  ssl_protocols TLSv1.2;
```

```
  # Выбор cipher suites: формируется по фактическому выводу openssl ciphers.
  # Минимально — использовать селекторы kGOST/aGOST (если они присутствуют),
  # и запретить слабые/нецелевые.
  ssl_ciphers 'kGOST:aGOST:!aNULL:!eNULL:!MD5:!RC4:!3DES:@STRENGTH';
  ssl_prefer_server_ciphers on;

  # Типовой контент для проверки
  location / {
    return 200 "GOST TLS endpoint OK\n";
    add_header Content-Type text/plain always;
  }
}
```

Основание по директивам: директивы `ssl_certificate`, `ssl_certificate_key`, `ssl_protocols`, `ssl_ciphers` определены в модуле Nginx SSL.{index=8}

6.3. Смешанный режим (ГОСТ + «обычный» TLS) — рекомендуемая эксплуатационная схема

Большинство массовых клиентов (включая стандартные браузеры) не поддерживают ГОСТ-cipher suites «по умолчанию». Поэтому эксплуатационно целесообразно: (1) держать стандартный HTTPS для широкой аудитории; (2) выделять ГОСТ-TLS как отдельную точку входа (отдельный порт/виртуальный хост/отдельный IP).

```
# Пример: обычный TLS на 443 (RSA/ECDSA) — отдельный server {}
# server { listen 443 ssl; ... }

# ГОСТ — отдельный server {}, как показано выше (8443)
```

Предупреждение: не следует «склеивать» требования регуляторного ГОСТ-профиля и публичного браузерного TLS в одну конфигурацию без формальной модели совместимости. На практике это приводит либо к недоступности для части клиентов, либо к «размыванию» криптопрофиля.

7. Тестирование ГОСТ-TLS

7.1. Проверка конфигурации Nginx

```
nginx -t
systemctl restart nginx
```

```
systemctl --no-pager --full status nginx
```

7.2. Проверка рукопожатия через OpenSSL

```
# 1) Проверить, какие GOST-ciphers доступны локально
openssl ciphers -v -tls1_2 | grep -i gost || true

# 2) Тест TLS 1.2 к ГОСТ-порту (пример)
# Выберите конкретный cipher из вывода openssl ciphers -v -tls1_2 | grep -i gost
openssl s_client -connect 127.0.0.1:8443 -tls1_2 -cipher 'kGOST' -brief

# 3) Подробный вывод при диагностике
openssl s_client -connect 127.0.0.1:8443 -tls1_2 -cipher 'kGOST' -state -msg
```

Справочно: в OpenSSL для выбора ГОСТ-cipher suites используются ключевые слова `kGOST`, `aGOST` и др., которые становятся рабочими при активном ГОСТ-engine.

8. Пример: корневые/промежуточные сертификаты, распространяемые через гос-инфраструктуру (Минцифры/ЕСИА-цепочки)

Для проверок цепочек, TLS-клиентов, прокси и внутренних сервисов может потребоваться установка корневых/промежуточных сертификатов, публикуемых для доверия к государственным сервисам. На практике встречаются файлы вида `russian_trusted_root_ca_pem.crt` и `russian_trusted_sub_ca_pem.crt` (и варианты для отдельных ОС/пакетов).

Источник: пример процедуры и ссылок на файлы/архивы приведён в технической инструкции (в т.ч. с указанием домена публикации и путей установки).

8.1. Загрузка и проверка сертификатов (примерный порядок)

```
mkdir -p /tmp/gov-certs
cd /tmp/gov-certs

# Пример (URL берите из официального источника/инструкции вашей организации):
# wget https://gu-st.ru/content/lending/russian_trusted_root_ca_pem.crt
# wget https://gu-st.ru/content/lending/russian_trusted_sub_ca_pem.crt
```

```
# Контроль: субъект, издатель, отпечаток, срок
openssl x509 -in russian_trusted_root_ca_pem.crt -noout -subject -issuer -dates -fingerprint -sha256
openssl x509 -in russian_trusted_sub_ca_pem.crt  -noout -subject -issuer -dates -fingerprint -sha256
```

Предупреждение: отпечатки (fingerprint) должны сверяться с доверенным источником вашей организации/регуляторным пакетом. Не используйте сертификаты, полученные из неофициальных каналов.

8.2. Установка в доверенное хранилище (RPM-подобный профиль)

```
# Пути и утилиты зависят от вашей реализации trust store.
# Для RPM-профиля часто используется ca-trust (update-ca-trust).

install -d -m 755 /etc/pki/ca-trust/source/anchors

cp -f russian_trusted_root_ca_pem.crt /etc/pki/ca-trust/source/anchors/
cp -f russian_trusted_sub_ca_pem.crt  /etc/pki/ca-trust/source/anchors/

update-ca-trust extract || update-ca-trust
```

9. Продуктивный сертификат (примерная схема)

Для продуктивной эксплуатации вместо самоподписанного сертификата используется сертификат от УЦ (в т.ч. корпоративного или аккредитованного), соответствующий вашему профилю ГОСТ.

9.1. Что требуется от сертификата для серверного TLS

- Наличие корректного Subject/SAN для доменного имени.
- Назначение ключа и расширения (KeyUsage/EKU) должны включать сценарий serverAuth по принятой политике УЦ.
- Цепочка должна быть полной: серверный сертификат + промежуточные, а корневой — в доверенном хранилище клиента.

9.2. Размещение цепочки в Nginx

Как правило, в `ssl_certificate` помещается файл «серверный + промежуточные» (fullchain), а в `ssl_certificate_key` — закрытый ключ.

```
server {
  listen 8443 ssl;
  server_name gost.example.local;

  # fullchain: leaf + intermediate(s)
  ssl_certificate     /etc/nginx/tls-gost/fullchain-gost.crt;
  ssl_certificate_key /etc/nginx/tls-gost/server-gost.key;

  ssl_protocols TLSv1.2;
  ssl_ciphers 'kGOST:aGOST:!aNULL:!eNULL:!MD5:!RC4:!3DES:@STRENGTH';
}
```

Примечание: Nginx допускает настройку TLS-директив согласно официальной документации, включая работу с сертификатами/ключами и наборами шифров.

10. Типовые неисправности и порядок устранения

10.1. Ошибка рукопожатия: «no shared cipher»

```
# 1) Проверить, что ГОСТ-ciphers реально доступны на сервере
openssl ciphers -v -tls1_2 | grep -i gost || true

# 2) Проверить, что Nginx применяет нужный набор (диагностика через s_client)
openssl s_client -connect 127.0.0.1:8443 -tls1_2 -cipher 'kGOST' -brief
```

10.2. Ошибка подписи/дайджеста при выпуске сертификата

```
# Проверка доступности md_gost12_256
openssl list -digest-algorithms | grep -i gost || true

# Прямая проверка дайджеста (пример для файла)
echo test > /tmp/t.txt
openssl dgst -md_gost12_256 /tmp/t.txt
```

Справочно: использование `md_gost12_256` и `gost2012_256` является типовым для OpenSSL-интеграций ГОСТ-движков.
Предупреждение: не рекомендуется «лечить» проблему отключением проверок/ослаблением профиля. Правильное устранение выполняется на уровне: (1) корректная активация ГОСТ-engine, (2) корректный выбор cipher suites, (3)

корректный сертификат и цепочка.

11. Заключительные положения

В НАЙС.ОС настройка ГОСТ-TLS для Nginx сводится к (1) подтверждению работоспособности ГОСТ-алгоритмов в OpenSSL, (2) выпуску/установке ГОСТ-сертификата и ключа, (3) заданию в Nginx протокола и набора cipher suites, соответствующих вашему криптопрофилю, (4) контролю цепочки доверия (в т.ч. через установку корневых/промежуточных сертификатов в trust store при необходимости).