НАЙС.ОС — Криптография с ГОСТ

Предупреждение по применению (регуляторика). Данный документ описывает техническую возможность использования алгоритмов ГОСТ в открытом криптостеке НАЙС.ОС (для разработки, тестирования, валидации, интеграционных стендов, CI/CD и эксплуатационных задач вне режимов, где требуется применение сертифицированного СКЗИ). В сценариях, где нормативно требуется сертифицированное средство криптографической защиты информации, необходимо использовать сертифицированный криптопровайдер/СКЗИ и соответствующую организационно-техническую модель применения.

1. Назначение и область применения

Настоящий документ предназначен для системных администраторов, инженеров эксплуатации, разработчиков и интеграторов, использующих НАЙС.ОС в качестве базовой платформы для серверов, виртуальных машин и контейнеров, где требуется техническая поддержка алгоритмов ГОСТ на уровне системных криптографических библиотек и утилит.

Основная цель стека ГОСТ в НАЙС.ОС: обеспечить использование российских криптоалгоритмов «из коробки» для задач разработки и тестирования, ускорить интеграцию и деплой, снизить стоимость стендов и пилотов за счёт отсутствия необходимости немедленного приобретения криптопровайдера в сценариях, где это не является юридическим требованием. В сценариях, где применение ГОСТ-криптографии регламентировано и требует сертифицированного СКЗИ, НАЙС.ОС рассматривается как инфраструктурная ОС, а криптопровайдер подключается отдельно.

1.1. Объект документирования

Описываются следующие подсистемы:

- **OpenSSL 3.5.x** с поддержкой ГОСТ через **gost-engine** (используется «из коробки», без установки дополнительных пакетов).
- **GnuTLS + Nettle** (TLS/криптография в user-space, включая ГОСТ-наборы шифров, Магма/Кузнечик в составе Nettle в НАЙС.ОС).

- **GnuPG + libgcrypt + libksba** (подпись/проверка, OpenPGP и CMS/X.509 для тестирования ГОСТ-объектов).
- **Nginx** в сценариях TLS-терминации с ГОСТ-сертификатами (как компонент инфраструктуры).
- **cryptsetup/LUKS + dm-crypt** (шифрование данных на диске с использованием алгоритмов, доступных в крипто-API ядра Linux и/или пользовательских библиотек).
- **Пароли и секреты** (разграничение: хэширование паролей, KDF, хранение секретов и применимость ГОСТ).

2. Термины, сокращения, состав нормативных ссылок

2.1. Термины

- **ГОСТ-алгоритмы** — набор криптографических алгоритмов, стандартизованных в РФ (подпись, хэширование, блочные шифры и т.п.).
- **СКЗИ** — средство криптографической защиты информации (в т.ч. сертифицированное).
- **PKI** — инфраструктура открытых ключей (сертификаты, УЦ, CRL/OCSP).
- **CMS** — Cryptographic Message Syntax (PKCS#7), контейнер подписи/шифрования.
- **dm-crypt** — подсистема ядра Linux для шифрования блочных устройств.
- **LUKS** — формат контейнера dm-crypt (LUKS1/LUKS2) с метаданными и слотами ключей.

2.2. Сокращения

- **ЭП** — электронная подпись.
- **УЦ** — удостоверяющий центр.
- **KDF** — Key Derivation Function, функция выработки ключа из пароля/секрета.
- **PBKDF2/Argon2** — распространённые KDF для паролей/ключей.
- **TLS** — Transport Layer Security.

2.3. Нормативные ссылки

В эксплуатационных и юридически значимых сценариях необходимо руководствоваться действующими нормативными актами РФ, внутренними регламентами организации и требованиями к применению сертифицированных

СКЗИ. В данном документе акцент сделан на технических аспектах реализации и проверки работы криптостека.

3. Архитектура криптографического стека НАЙС.ОС

Поддержка ГОСТ в НАЙС.ОС реализуется как набор совместимых подсистем, каждая из которых закрывает свой класс задач: TLS-соединения, подписи/контейнеры, ключевая инфраструктура, шифрование на диске, прикладные операции.

Логическая схема

```
Приложения
  Nginx / web-сервисы (TLS)
  curl / gnutls-cli (диагностика TLS)
  GnuPG (подпись/проверка OpenPGP, CMS)
  cryptsetup (LUKS/диски)
  сервисы/скрипты (KDF, подпись, шифрование файлов)
     
Системные криптобиблиотеки
  OpenSSL 3.5.x + gost-engine (ГОСТ для OpenSSL-потребителей)
  GnuTLS + Nettle (TLS и симметричные ГОСТ-алгоритмы)
  libgcrypt + libksba (подпись/сертификаты/CMS, тестирование)
     
Нижний уровень
  crypto API ядра Linux (/proc/crypto), dm-crypt
  аппаратные устройства/PKCS#11 (при наличии и при настройке)
```

3.1. Принцип поставки «из коробки»

НАЙС.ОС поставляет ГОСТ-функциональность как часть базового стека: пользователю не требуется устанавливать дополнительные пакеты для включения ГОСТ в OpenSSL (используется проект gost-engine), а также доступна криптография ГОСТ в пользовательских инструментах (GnuTLS/Nettle, GnuPG/libgcrypt/libksba) согласно комплектации системы.

4. Перечень поддерживаемых алгоритмов ГОСТ

Фактический набор доступных алгоритмов определяется сборкой библиотек, конфигурацией OpenSSL и доступностью криптопримитивов в ядре Linux. Верификация выполняется командами, приведёнными в разделах 5–12.

Категория	Алгоритмы	Где используется	Ключевые потребители
Хэширование	ГОСТ Р 34.11-2012 (Стрибог 256/512) (в зависимости от сборки также могут присутствовать устаревшие варианты для совместимости)	Подпись, KDF, контроль целостности, PKI	OpenSSL, libgcrypt/GnuPG, Nettle/GnuTLS
Подпись / ЭЦП	ГОСТ Р 34.10-2012 (256/512), параметры/кривые ТК-26	PKI, CMS, тестирование ЭП, подпись артефактов	OpenSSL, GnuPG (OpenPGP и gpgsm/CMS)
Симметричное шифрование	ГОСТ Р 34.12-2015: Кузнечик, Магма ГОСТ 28147-89 (наследуемые контуры)	TLS, шифрование данных, совместимость с ГОСТ-сервисами	GnuTLS/Nettle; OpenSSL (через gost-engine); dm-crypt (если доступно в ядре)
Режимы и аутентификация	CTR-ACPKM, OMAC/IMIT и др. (по доступности в реализации)	TLS (ГОСТ-наборы), протоколы, контейнеры	GnuTLS/Nettle, прикладные библиотеки
Обмен ключами	VKO/ECGOST-производные механизмы (в составе стека)	TLS, CMS, протоколы взаимодействия	GnuTLS/Nettle, OpenSSL/GnuPG (в зависимости от сборки)

Примечание. Для целей данного документа «поддержка алгоритма» трактуется как возможность перечисления, выбора и практического применения в штатных утилитах системы. Юридическая значимость и соответствие требованиям регуляторов зависят от режима применения и наличия сертифицированных компонентов в требуемых случаях.

5. OpenSSL + ГОСТ (gost-engine): назначение, проверка, базовые операции

В НАЙС.ОС ГОСТ-криптография для OpenSSL доступна «из коробки» и подключается через проект gost-engine (engine-модуль OpenSSL). Это позволяет использовать ГОСТ-алгоритмы в инструментах и сервисах, которые ориентируются на OpenSSL как на

криптографический backend.

5.1. Проверка версии и активной конфигурации

```
# Версия OpenSSL
openssl version -a

# Проверка доступных ENGINE (при наличии команды engine)
openssl engine -t -c 2>/dev/null | sed -n '1,200p'

# Поиск ГОСТ-алгоритмов в списках (фактические имена зависят от сборки)
openssl list -digest-algorithms | grep -i gost || true
openssl list -public-key-algorithms | grep -i gost || true
openssl list -cipher-algorithms | grep -i -E 'gost|kuz|magma|grasshopper' || true
```

Требование к интерпретации результатов. В OpenSSL имена алгоритмов и провайдеров/engine-плагинов зависят от конкретной сборки. В практических сценариях следует: (1) сначала получить фактические имена алгоритмов командой openssl list ...; (2) использовать эти имена в командах подписи/шифрования/генерации ключей; (3) фиксировать полученные списки в эксплуатационной документации конкретного контура.

5.2. Подготовка окружения: конфигурация OpenSSL

В типовой схеме загрузка ГОСТ-engine осуществляется через системный файл конфигурации OpenSSL (например, /etc/ssl/openssl.cnf или эквивалентный путь в НАЙС.ОС). На уровне инфраструктуры допускаются два подхода:

1. **Системный** — ГОСТ-engine подключён в системном openssl.cnf (рекомендуется для единообразия).
2. **Локальный** — задаётся переменная окружения OPENSSL_CONF на сервис/юнит (точечное включение).

Пример минимальной схемы конфигурации (референс-шаблон):

```
# Пример. Адаптируйте пути под НАЙС.ОС.
# Файл: /etc/ssl/openssl.cnf (или аналог)

openssl_conf = openssl_init

[openssl_init]
engines = engine_section
```

```
[engine_section]
gost = gost_section

[gost_section]
engine_id = gost
# dynamic_path = /usr/lib64/engines-3/gost.so
default_algorithms = ALL
```

Техническое примечание. OpenSSL 3.x развивается в сторону provider-модели, при этом ENGINE-подсистема используется для обратной совместимости. В инфраструктуре допускается эксплуатация engine-подхода при условии документирования конфигурации и контроля обновлений.

5.3. Генерация ключей ГОСТ и выпуск сертификатов (практическая схема)

Ниже приведён рекомендуемый порядок действий, который не опирается на «предполагаемые» имена алгоритмов, а сначала извлекает фактические значения из OpenSSL.

Шаг 1. Определить доступные алгоритмы ключей и хэшей

```
openssl list -public-key-algorithms | grep -i gost || true
openssl list -digest-algorithms | grep -i gost || true
```

Шаг 2. Сгенерировать ключ (примерная форма)

Команда генерации зависит от того, как именно engine экспортирует алгоритмы (названия могут отличаться). Типовая форма для OpenSSL 3.x:

```
# Пример (адаптируйте алгоритм и параметры по выводу openssl list)
# Возможные имена: gost2012_256, gost2012_512 и т.п.
openssl genpkey -algorithm gost2012_256 -out gost2012_256.key.pem

# Получить открытый ключ
openssl pkey -in gost2012_256.key.pem -pubout -out gost2012_256.pub.pem
```

Шаг 3. Сформировать CSR и самоподписанный сертификат

```
# CSR (PKCS#10)
openssl req -new \
```

```
 -key gost2012_256.key.pem \
 -subj "/C=RU/ST=Moscow/L=Moscow/O=NiceOS/OU=Lab/CN=localhost" \
 -out gost2012_256.csr.pem

# Самоподписанный сертификат (для стенда)
openssl req -x509 -new \
 -key gost2012_256.key.pem \
 -subj "/C=RU/ST=Moscow/L=Moscow/O=NiceOS/OU=Lab/CN=localhost" \
 -days 365 \
 -out gost2012_256.crt.pem
```

Шаг 4. Проверка сертификата и OID

```
openssl x509 -in gost2012_256.crt.pem -noout -text | sed -n '1,200p'
openssl x509 -in gost2012_256.crt.pem -noout -text | grep -E "Signature Algorithm|Public Key
Algorithm|1\.2\.643" -n
```

5.4. Подпись и проверка файлов (тестовый контур)

Для тестирования корректности стека удобно использовать CMS/PKCS#7 подписи, так как они близки к реальным контурам ЭДО.

```
# Подписать файл в CMS (DER)
openssl cms -sign \
 -binary -in document.bin \
 -signer gost2012_256.crt.pem \
 -inkey gost2012_256.key.pem \
 -outform DER -out document.p7s \
 -nodetach

# Проверить подпись (для самоподписанного стенда используйте -CAfile тем же сертификатом)
openssl cms -verify \
 -binary -in document.p7s -inform DER \
 -CAfile gost2012_256.crt.pem \
 -out /dev/null
```

Примечание по диагностике. При ошибках «unsupported algorithm / unknown OID» первично проверяются: (1) активность загрузки gost-engine; (2) наличие алгоритмов в openssl list; (3) корректность конфигурации OPENSSL_CONF для контекста выполнения (systemd/контейнер).

6. TLS по ГОСТ: GnuTLS/Nettle и практическая диагностика соединений

Для сценариев TLS-диагностики и сервисов, использующих GnuTLS, в НАЙС.ОС применяется связка **GnuTLS + Nettle**. При наличии ГОСТ-алгоритмов в Nettle и включенной опции ГОСТ в GnuTLS становится доступна практическая работа с ГОСТ-набором шифров (как правило, в рамках TLS 1.2).

6.1. Проверка наличия ГОСТ в GnuTLS

```
# Список возможностей
gnutls-cli --version

# Поиск ГОСТ в доступных алгоритмах/приоритетах (варианты зависят от версии)
gnutls-cli --list 2>/dev/null | grep -i gost || true
```

6.2. Тестовый TLS-сервер и клиент (ГОСТ-приоритет)

Для локальной проверки используется gnutls-serv и gnutls-cli. Сертификат может быть выпущен утилитой certtool (GnuTLS-утилита) либо OpenSSL (при условии, что сертификат и ключ доступны в понятных форматах).

Пример: генерация ключа и самоподписанного сертификата через certtool

```
# Генерация закрытого ключа (пример; фактический тип ключа уточняйте по справке certtool)
certtool --generate-privkey --outfile gost.key

# Шаблон сертификата
cat > server.tmpl <<'EOF'
cn = "localhost"
expiration_days = 365
tls_www_server
dns_name = "localhost"
EOF

# Самоподписанный сертификат
certtool --generate-self-signed \
 --load-privkey gost.key \
 --template server.tmpl \
 --outfile gost.crt
```

Запуск сервера и подключение клиента с ГОСТ-приоритетом

```
# Сервер (TLS 1.2, ГОСТ-приоритет)
gnutls-serv --priority 'NONE:+VERS-TLS1.2:+GOST' \
  --x509certfile gost.crt --x509keyfile gost.key \
  --port 5555

# Клиент
gnutls-cli --priority 'NONE:+VERS-TLS1.2:+GOST' -p 5555 localhost
```

Ограничение. ГОСТ-наборы шифров, как правило, применяются в TLS 1.2. В эксплуатационной документации контура следует фиксировать политику протоколов (включая отключение TLS 1.3 при необходимости соответствия профилю).

7. GnuPG/libgcrypt/libksba: подпись и проверка (OpenPGP и CMS/X.509)

Поддержка ГОСТ в GnuPG (в составе НАЙС.ОС) используется как инженерный инструмент: генерация тестовых ключей, проверка подписи, анализ сертификатов и контейнеров, репликация ошибок интеграции без необходимости немедленного подключения проприетарного криптопровайдера.

7.1. Роли компонентов

- **gpg** — OpenPGP (ключи, подпись, шифрование) для сценариев CI/подписания артефактов и тестов.
- **gpgsm** — X.509/CMS (S/MIME, контейнеры подписи), близко к документообороту и PKI-контурам.
- **libgcrypt** — криптографические примитивы (в т.ч. ГОСТ-хэши/подписи при сборке с поддержкой).
- **libksba** — ASN.1/DER, X.509, CMS, OCSP/CRL (разбор/проверка структур и OID).

7.2. Минимальная проверка «ГОСТ включён»

```
# Общая информация о сборке
gpg --version
gpgsm --version

# Проверка импорта ГОСТ-сертификата (примерный контур)
# Файл gost256.crt должен быть в DER/PEM в соответствии с вашими тестовыми данными
```

```
gpgsm --debug-all --import gost256.crt
```

7.3. Практические сценарии использования в инженерных контурах

- **Тестирование совместимости** сертификатов/подписей от российских УЦ (анализ OID, EKU, политик).
- **Регрессионные тесты интеграции** (после обновления OpenSSL/GnuTLS/libgcrypt).
- **Подготовка CI/CD**: подпись артефактов, проверка цепочек в pipeline, проверка CMS-контейнеров.
- **Диагностика «на стенде»**: воспроизведение ошибок без закупки криптопровайдера на ранней стадии проекта.

Практическая модель применения. НАЙС.ОС закрывает инженерный цикл «разработка тестирование интеграция деплой». На этапе перехода в юридически значимую эксплуатацию подключается сертифицированный криптопровайдер (если это требуется нормативно), при этом инфраструктурный контур уже подготовлен и протестирован.

8. Nginx и ГОСТ-сертификаты: место в архитектуре

Nginx в типовой инфраструктуре выполняет роль TLS-терминатора и reverse-proxy. Применение ГОСТ-сертификатов в Nginx зависит от того, каким криптографическим backend он собран/связан: чаще всего это OpenSSL. В НАЙС.ОС, при наличии OpenSSL+ГОСТ (через gost-engine), возможно формирование стендов и пилотных инсталляций с ГОСТ-сертификатами для тестирования.

8.1. Общие требования к контуру

- Nginx должен быть собран с поддержкой OpenSSL (типовая сборка).
- ГОСТ-engine должен быть загружен в контексте процесса Nginx (через системный openssl.cnf или OPENSSL_CONF).
- Выбор ciphersuites должен соответствовать требованиям совместимости клиента (обычно TLS 1.2 и ГОСТ-наборы).

8.2. Практический порядок подготовки

1. Сгенерировать ГОСТ-ключ/сертификат (раздел 5.3) либо использовать сертификат тестового УЦ.
2. Проверить, что openssl ciphers видит ГОСТ-наборы.
3. Настроить Nginx на TLS 1.2 и ограничить наборы шифров под ГОСТ-профиль (в рамках стенда).
4. Проверить соединение клиентом: gnutls-cli --priority 'NONE:+VERS-TLS1.2:+GOST'.

8.3. Референс-фрагмент конфигурации Nginx (шаблон)

```
server {
 listen 443 ssl;
 server_name example.local;

 # Сертификат и ключ (ГОСТ)
 ssl_certificate     /etc/nginx/tls/server.crt.pem;
 ssl_certificate_key /etc/nginx/tls/server.key.pem;

 # Для ГОСТ обычно применяется TLS 1.2 (политика протоколов фиксируется в документации контура)
 ssl_protocols TLSv1.2;

 # Наборы шифров указываются по фактическому выводу:
 #   openssl ciphers -v | grep -i gost
 # Примерная форма (ЗАПОЛНИТЬ ПО ФАКТУ):
 ssl_ciphers 'GOST-...:GOST-...';

 ssl_prefer_server_ciphers on;

 location / {
   proxy_pass http://127.0.0.1:8080;
   proxy_set_header Host $host;
   proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
   proxy_set_header X-Forwarded-Proto $scheme;
 }
}
```

Контрольный тест. После перезапуска Nginx выполните проверку: gnutls-cli --priority 'NONE:+VERS-TLS1.2:+GOST' example.local -p 443.

9. Шифрование данных на диске: cryptsetup/LUKS и ГОСТ-алгоритмы

Шифрование данных на диске в Linux реализуется через **dm-crypt** (подсистема ядра), а управление контейнерами выполняет **cryptsetup**. Выбор шифра/режима определяется доступностью алгоритмов в крипто-API ядра Linux, а параметры KDF и метаданных — возможностями cryptsetup и политикой безопасности контура.

Ключевой принцип. В LUKS-контейнере важно разделять: (1) алгоритм шифрования данных (cipher/mode для dm-crypt) и (2) алгоритм защиты ключа (KDF для парольной фразы, параметры слота ключа). В ГОСТ-контурах обычно нормируется именно криптографический профиль (алгоритмы/режимы/длины ключей), при этом парольная KDF может оставаться современной (Argon2id и т.п.), если это допустимо политикой и регламентом.

9.1. Верификация доступных алгоритмов ядра

```
# Список криптоалгоритмов ядра
cat /proc/crypto | sed -n '1,120p'

# Поиск ГОСТ/Кузнечик/Магма по именам (фактические имена зависят от ядра и модулей)
grep -i -E 'gost|kuz|magma|grasshopper|streebog' /proc/crypto || true
```

9.2. Диагностика cryptsetup: производительность и доступность

```
# Бенчмарки cryptsetup (помогают определить доступные комбинации и скорость)
cryptsetup benchmark

# При необходимости — указать предполагаемый cipher (зависит от доступности в ядре)
# cryptsetup benchmark --cipher <CIPHER-NAME>
```

9.3. Референс-схема создания LUKS2 (шаблон)

Поскольку имена ГОСТ-шифров для dm-crypt зависят от конкретного ядра и сборки, документируется шаблонная команда. Перед применением необходимо: (1) зафиксировать доступный cipher/mode по /proc/crypto и cryptsetup benchmark; (2)

подтвердить требованиями контура (ключи, режимы, параметры KDF).

```
# ВНИМАНИЕ: пример-шаблон. Подставьте фактические значения cipher/mode.
# Устройство: /dev/vdb (пример)
# Контейнер: LUKS2
# KDF: Argon2id (рекомендуется как парольная KDF для защиты ключевого слота)

cryptsetup luksFormat /dev/vdb \
 --type luks2 \
 --cipher <CIPHER> \
 --cipher-mode <MODE> \
 --key-size <BITS> \
 --pbkdf argon2id \
 --iter-time 3000

# Открытие контейнера
cryptsetup open /dev/vdb secure_data

# Создание ФС
mkfs.ext4 /dev/mapper/secure_data

# Монтирование
mount /dev/mapper/secure_data /mnt/secure_data
```

9.4. Эксплуатационные операции LUKS

```
# Информация о контейнере
cryptsetup luksDump /dev/vdb

# Резервное копирование заголовка (критично для эксплуатации)
cryptsetup luksHeaderBackup /dev/vdb --header-backup-file luks-header.vdb.bin

# Добавление дополнительной парольной фразы (новый слот)
cryptsetup luksAddKey /dev/vdb

# Удаление слота (осторожно: не потерять доступ)
cryptsetup luksKillSlot /dev/vdb <SLOT_NUMBER>
```

Ограничение и ответственность. Неправильный выбор cipher/mode или потеря заголовка LUKS ведут к утрате доступа к данным. Для каждого контура требуется регламент: резервирование заголовков, управление слотами, ротация ключей, контроль параметров KDF.

10. Пароли и секреты: где применим ГОСТ, а где требуются современные KDF

10.1. Разграничение задач

Задача	Правильная криптографическая модель	Роль ГОСТ
Хранение паролей пользователей (аутентификация)	Памятьёмкие KDF: yescrypt/argon2id (в зависимости от политики), соль, параметры стоимости	ГОСТ-хэш не является «по умолчанию» правильным заменителем парольной KDF; ГОСТ применим в иных слоях (PKI/ЭП), либо в рамках регламентов, где это отдельно определено
Хранение секретов приложений (API keys, токены)	Не «хэшировать», а шифровать: KMS/Vault, LUKS, HSM/PKCS#11, ACL, аудит	ГОСТ применим как алгоритм шифрования/подписи в хранилище секретов (в зависимости от выбранного механизма)
Выработка ключей из пароля для шифрования (KDF)	PBKDF2/Argon2id + соль + параметры; предпочтение Argon2id	ГОСТ-хэш может использоваться как PRF/хэш в PBKDF2 при наличии реализации; при этом требования контура должны быть явно зафиксированы
Юридически значимая подпись	Сертифицированное СКЗИ + регламент + PKI	ГОСТ обязателен по регламентам, но реализация должна быть сертифицированной при соответствующих требованиях

10.2. Практический инженерный подход в НАЙС.ОС

В инженерных контурах (разработка/тестирование) целесообразно:

- использовать ГОСТ-подписи и ГОСТ-сертификаты для воспроизведения интеграции (TLS, CMS, PKI);

- использовать современные парольные KDF для локальной аутентификации (даже если в контуре применяется ГОСТ в PKI);
- использовать LUKS/шифрование дисков для секретов, а не «самодельное шифрование паролей».

10.3. Референс-пример PBKDF2 с ГОСТ-хэшем (шаблон)

Если требуется получить ключ из пароля с применением ГОСТ-хэш-функции, сперва фиксируются доступные имена дайджестов:

```
openssl list -digest-algorithms | grep -i gost || true
```

Далее применяется KDF-утилита OpenSSL (если доступна в вашей сборке OpenSSL 3.x). Пример-шаблон:

```
# Пример: выработка 32-байтного ключа PBKDF2 с ГОСТ-дайджестом.
# Фактическое имя digest подставьте из `openssl list -digest-algorithms`.

openssl kdf -keylen 32 -kdfopt digest:md_gost12_256 \
 -kdfopt pass:'CorrectHorseBatteryStaple' \
 -kdfopt salt:0102030405060708 \
 -kdfopt iter:200000 PBKDF2 | xxd -p -c 32
```

Контроль качества. Для парольных KDF приоритетом является стойкость к перебору (памятьёмкость/стоимость). Если регламент не требует иного, для новых систем предпочтительнее Argon2id (в LUKS и в прикладных системах), а ГОСТ использовать на уровне PKI, TLS-профилей и регламентированных криптоопераций.

11. Типовые сценарии применения в инфраструктуре

11.1. Разработка и тестирование интеграции с ГОСТ-инфраструктурой

- Проверка сертификатов: цепочки, политики, EKU, OID.
- Проверка CMS-контейнеров: подпись/верификация на стенде.
- Эмуляция ГОСТ-TLS на тестовых эндпоинтах (GnuTLS/gnutls-serv, Nginx).
- Регрессионное тестирование после обновлений криптобиблиотек (OpenSSL/GnuTLS/libgcrypt).

11.2. Быстрый деплой сервисов с требованиями по ГОСТ (не СКЗИ-режим)

- Развертывание reverse-proxy (Nginx) на стендах интеграции.
- Настройка шифрования дисков на ВМ/узлах (LUKS2) с документированными параметрами.
- Ведение воспроизводимой конфигурации (IaC) с проверками «алгоритм доступен».

11.3. Переход в регламентированный режим

В момент, когда проект переходит в контур, требующий сертифицированного СКЗИ:

- подключается сертифицированный криптопровайдер (PKCS#11/CSP/агент/аппаратный модуль),
- фиксируются профили TLS/PKI по требованиям,
- включаются организационные меры: регламенты ключей, аудит, допуск администраторов, журналы, контроль обновлений.

12. Единый чек-лист проверки «ГОСТ работает»

1. **OpenSSL**:

```
openssl version
openssl list -digest-algorithms | grep -i gost
openssl list -public-key-algorithms | grep -i gost
openssl list -cipher-algorithms | grep -i -E 'gost|kuz|magma|grasshopper'
```

2. **GnuTLS/Nettle (TLS диагностика)**:

```
gnutls-cli --version
gnutls-cli --list 2>/dev/null | grep -i gost || true
gnutls-cli --priority 'NONE:+VERS-TLS1.2:+GOST' example.local -p 443
```

3. **GnuPG/libksba (сертификаты и CMS)**:

```
gpg --version
gpgsm --version
```

```
gpgsm --debug-all --import gost256.crt
```

4. **cryptsetup/dm-crypt**:

```
cryptsetup benchmark
grep -i -E 'gost|kuz|magma|grasshopper|streebog' /proc/crypto || true
```

Критерий прохождения. ГОСТ-алгоритмы должны быть перечислены хотя бы в одном из криптобэкендов (OpenSSL/GnuTLS/libgcrypt), а тестовый сценарий (подпись или TLS-рукопожатие) должен быть воспроизводим на стенде.
Рекомендация по документации контуров. В эксплуатационных документах фиксируйте: версии пакетов, конфигурацию openssl.cnf (или OPENSSL_CONF), наборы TLS-алгоритмов, параметры LUKS/KDF, а также контрольные команды и ожидаемые фрагменты вывода.