Применение алгоритмов ГОСТ в cryptsetup

Введение

Современные защищённые операционные системы, такие как **НАЙС.ОС**, ориентированы на соответствие требованиям информационной безопасности, в том числе **ФСТЭК России**. Одним из ключевых элементов этой безопасности является **криптографическая защита данных**, реализуемая через проверенные стандартизованные алгоритмы.

В данной статье мы рассмотрим **применение отечественных криптографических алгоритмов ГОСТ** в инструментах `cryptsetup` и `dm-crypt` под управлением НАЙС.ОС. Особое внимание уделим форматам `LUKS1`, `LUKS2`, алгоритмам шифрования, режимам MAC, а также способам их настройки, использования и отладки.

🎯 **Цель статьи:** показать на практике, как применять алгоритмы ГОСТ для шифрования и контроля целостности (MAC) с помощью `cryptsetup` в системе **НАЙС.ОС**.

🔐 **Почему ГОСТ?**

- ✅ **Соответствие требованиям ФСТЭК и регуляторов** — ГОСТ-алгоритмы обязательны для сертифицированных ИС.
- ✅ **Поддержка в ядре Linux** — в НАЙС.ОС уже включены соответствующие модули и патчи.
- ✅ **Интеграция с dm-crypt и LUKS** — возможно полноценное использование как для шифрования, так и для MAC.

Все алгоритмы **ГОСТ Р 34.12-2015**, **ГОСТ 28147-89**, **ГОСТ Р 34.11-94** и **ГОСТ Р 34.11-2012** уже доступны в ядре **НАЙС.ОС** и зарегистрированы в `/proc/crypto`, а также распознаются утилитой `cryptsetup` в виде стандартных имён: `kuznyechik`, `magma`, `gost28147-cpa`, `streebog256` и др.

✍ В этой статье вы найдёте примеры **проверки доступности алгоритмов**, **создания зашифрованных томов** с ГОСТ-алгоритмами, **использования integrity** через

HMAC/CMAC, а также **оптимизации и отладки**.

1. Обзор криптографических ГОСТ-алгоритмов, доступных в НАЙС.ОС

В дистрибутиве **НАЙС.ОС** на уровне ядра реализована поддержка криптографических алгоритмов из российских стандартов **ГОСТ Р 34.12-2015** и **ГОСТ 28147-89**, что делает возможным использование этих алгоритмов в системах с повышенными требованиями к сертификации и безопасности.

1.1. Блочные шифры

Ниже представлены основные блочные шифры, поддерживаемые в ядре и доступные через `/proc/crypto` и `cryptsetup`:

kuznyechik

Современный 128-битный блочный шифр, стандартизованный как **ГОСТ Р 34.12-2015**.

- Размер блока: **128 бит**
- Поддерживает режимы: `CBC`, `CTR`, `XTS`, `ESSIV` и др.
- Пример использования: `kuznyechik-xts-plain64`

magma

Классический 64-битный блочный шифр, также стандартизованный **в ГОСТ Р 34.12-2015**. Является преемником ГОСТ 28147-89, но в новой редакции.

- Размер блока: **64 бита**
- Поддерживает: `CBC`, `ESSIV`, `CTR` и другие режимы
- Пример: `magma-cbc-essiv:streebog512`

gost28147-89

Исторически первый отечественный блочный шифр. Несмотря на возраст, до сих пор используется в некоторых сертифицированных системах. Поддерживает различные варианты S-box, соответствующие криптопрофилям:

- `gost28147-tc26z` — современный набор S-блоков от ТК26
- `gost28147-cpa`, `cpb`, `cpc`, `cpd` — CryptoPro-профили A–D

Примеры использования:

- `cbc(gost28147-cpa)`
- `cnt(gost28147-tc26z)` (режим аналогичный CTR)

💡 Все перечисленные алгоритмы уже загружены в ядро НАЙС.ОС и отображаются в `/proc/crypto`. Их можно применять напрямую в параметрах `cryptsetup --cipher` или `--integrity`.

1.2. Режимы шифрования

К ГОСТ-шифрам в НАЙС.ОС можно применять широкий спектр **режимов блочного шифрования**, поддерживаемых как Crypto API ядра, так и `cryptsetup`:

`ECB` — Electronic Codebook

Простейший режим, небезопасен для практического использования, т.к. не скрывает повторяющиеся блоки. Применяется только в тестах.

`CFB` — Cipher Feedback

Поддерживает потоковое шифрование, часто используется с ГОСТ 28147-89 в embedded-системах и телекоме.

`CBC` — Cipher Block Chaining

Популярный режим для блочных шифров, совместим с ESSIV, защищает от повторов.

`CTR` — Counter Mode

Потоковый режим, обеспечивающий параллелизм и высокую производительность. Вариант — CNT для ГОСТ 28147.

`XTS` — XEX Tweakable Block Cipher

Рекомендуется для дискового шифрования. Поддерживается для `kuznyechik`, `magma`.

`CNT` — ГОСТ-совместимый CTR

Вариант режима CTR, адаптированный для ГОСТ 28147, используется в `cnt(gost28147-...)`.

ESSIV — Encrypted Salt Sector IV

Защищённый IV-режим для CBC, используется совместно с хэш-функцией (например, essiv:streebog512).

📌 Примеры использования:

- kuznyechik-xts-plain64
- magma-cbc-essiv:streebog512
- cnt(gost28147-cpa)

1.3. Хэш-функции и имитовставки

В НАЙС.ОС доступны хэш-функции и MAC-алгоритмы (имитовставки), соответствующие ГОСТ-стандартам. Они используются в качестве:

- Хеш-функций для LUKS (--hash);
- Алгоритмов контроля целостности (--integrity);
- Генерации ключей и IV при использовании ESSIV и HMAC.

streebog256 / streebog512

Современный криптографический хеш из **ГОСТ Р 34.11-2012**. Поддерживается в ядре как streebog256 и streebog512, применяется во многих сценариях — от ESSIV до HMAC.

gosthash94

Старый ГОСТ-хеш из стандарта **ГОСТ Р 34.11-94**. Используется в унаследованных системах и сертифицированных решениях. Доступен как gosthash94.

gost28147imit-*

Классические **имитовставки** (MAC) на базе ГОСТ 28147-89. Доступны для всех вариантов S-box:

- gost28147imit-tc26z
- gost28147imit-cpa, cpb, cpc, cpd

Могут использоваться в integrity-настройках с LUKS2 (при наличии поддержки в dm-integrity).

💡 Некоторые алгоритмы могут быть недоступны без загрузки модулей ядра.

Используйте `modprobe` или проверьте `/proc/crypto`.

1.4. HMAC и CMAC

Для обеспечения **целостности данных** и защиты от подделки в `cryptsetup` и `LUKS2` используются MAC-алгоритмы — такие как **HMAC** и **CMAC**. В НАЙС.ОС доступны ГОСТ-совместимые реализации на базе сертифицированных алгоритмов.

hmac(streebog256)

Основан на современной хэш-функции **ГОСТ Р 34.11-2012**. Используется в качестве алгоритма контроля целостности в LUKS2:

```
--integrity hmac(streebog256)
```

Поддерживается dm-integrity и cryptsetup с sector-size от 4096 байт.

hmac(gosthash94)

Реализация HMAC на базе **ГОСТ Р 34.11-94**. Актуальна для систем, требующих обратной совместимости с ранними стандартами. Используется с:

- `--hash gosthash94`
- `--integrity hmac(gosthash94)`

cmac(kuznyechik)

Реализация **CMAC** (Cipher-based MAC) на основе шифра `kuznyechik` (ГОСТ Р 34.12-2015). Может применяться как альтернатива HMAC в LUKS2 (если поддерживается ядром и `dm-integrity`):

```
--integrity cmac(kuznyechik)
```

cmac(magma)

Аналогичная реализация CMAC на основе шифра `magma`. Используется в системах с сертифицированным стеком ГОСТ:

```
--integrity cmac(magma)
```

💡 **Примечание:** использование CMAC требует совместимой версии `cryptsetup` и

наличия поддержки в `dm-integrity`.

2. Проверка доступности алгоритмов в системе

Перед использованием алгоритмов ГОСТ в `cryptsetup`, важно убедиться, что соответствующие модули ядра загружены и криптографические примитивы доступны через **Crypto API**. Один из простейших способов — посмотреть содержимое `/proc/crypto`.

2.1. Через `/proc/crypto`

Для просмотра зарегистрированных алгоритмов используйте следующую команду:

```
grep -E 'kuznyechik|magma|gost|streebog' /proc/crypto
```

Вывод покажет зарегистрированные шифры, хэши и MAC-алгоритмы, например:

```
name        : kuznyechik
driver      : kuznyechik-generic
module      : kernel
...
name        : streebog256
driver      : streebog256-generic
...
name        : cmac(kuznyechik)
...
name        : hmac(streebog256)
...
name        : cnt(gost28147-cpa)
```

📌 **Совет:** если нужный алгоритм не отображается, проверьте, загружен ли соответствующий модуль ядра с помощью `lsmod` или загрузите его вручную:

```
sudo modprobe kuznyechik
```

⚠️ Некоторые алгоритмы (например, `gost28147-cpb` или `gosthash94`) могут быть доступны только в модифицированных ядрах или включаемы при установке спецпакетов в НАЙС.ОС.

2.2. Через `cryptsetup benchmark`

Утилита `cryptsetup` предоставляет встроенный инструмент **benchmark** для оценки скорости доступных алгоритмов шифрования и хеширования. Это также полезно для проверки наличия и работоспособности ГОСТ-алгоритмов в системе.

Основная команда:

```
sudo cryptsetup benchmark
```

Пример части вывода:

```
# Tests are approximate using memory only (no storage IO).
PBKDF2-sha1       1061350 iterations per second
PBKDF2-sha256     1309275 iterations per second
...
kuznyechik-xts   256b  152 MB/s
magma-cbc        128b   98 MB/s
cnt(gost28147-cpa)256b 104 MB/s
```

Проверка конкретного алгоритма:

Можно указать нужный шифр явно, чтобы проверить только его:

```
sudo cryptsetup benchmark --cipher kuznyechik-xts-plain64
```

Если алгоритм не поддерживается, вы получите ошибку:

```
No such cipher: kuznyechik-xts-plain64
```

✅ **Если шифр отображается и проходит тестирование — значит он доступен в ядре и может использоваться для `luksFormat`.**
⚠️ **Важно:** скорость шифрования и хэширования при benchmark-тесте не учитывает реальный диск. Это синтетический тест в памяти.
💡 Для нестандартных алгоритмов также можно протестировать, например:

- `cryptsetup benchmark --cipher cnt(gost28147-cpb)`
- `cryptsetup benchmark --cipher magma-cbc-essiv:streebog512`

3. Использование ГОСТ в LUKS1/LUKS2

3.1. LUKS2 с Кузнечиком (режим XTS)

Алгоритм **Кузнечик** (`kuznyechik`) из ГОСТ Р 34.12-2015 может использоваться в режиме XTS как основной шифр для шифрования томов в формате **LUKS2**.

Пример команды для форматирования раздела с использованием `kuznyechik-xts-plain64` и хэш-функции `streebog256`:

```
sudo cryptsetup luksFormat /dev/sdb1 \
  --type luks2 \
  --cipher kuznyechik-xts-plain64 \
  --key-size 512 \
  --hash streebog256
```

Здесь:

- `--type luks2` — используем формат LUKS2, необходимый для большинства современных функций.
- `--cipher kuznyechik-xts-plain64` — блочный шифр ГОСТ в режиме XTS (рекомендуемый для дисков).
- `--key-size 512` — XTS требует двойного ключа (по 256 бит на каждый блок).
- `--hash streebog256` — используется при derivation ключа (KDF) и для подписи заголовка.

✅ **Поддержка:** Все параметры доступны в ядре **НАЙС.ОС**, начиная с версии с включённым Crypto API ГОСТ.
⚠️ **Важно:** всегда делайте `--dry-run` на тестовом разделе, прежде чем форматировать реальные данные.

3.2. LUKS2 с Магмой (режим CBC + ESSIV)

Шифр **Магма** (ГОСТ Р 34.12-2015, 64-битный блок) может использоваться с режимом CBC, защищённым через **ESSIV** для устранения уязвимостей с предсказуемыми IV.

Пример создания LUKS2-раздела с использованием Магмы и `streebog512` в качестве хеш-функции:

```
sudo cryptsetup luksFormat /dev/sdb2 \
  --type luks2 \
  --cipher magma-cbc-essiv:streebog512 \
```

```
  --key-size 256 \
  --hash streebog512
```

Разбор параметров:

- `--cipher magma-cbc-essiv:streebog512` — блочный шифр Магма в режиме CBC, защищённом ESSIV, где IV зависит от `streebog512`.
- `--key-size 256` — полный размер ключа, хотя Магма использует 256-битные ключи по умолчанию.
- `--hash streebog512` — для KDF и подписи заголовка.

📌 **Примечание:** Магма имеет 64-битный блок и потому **не рекомендуется** для больших томов или SSD/NVMe — лучше использовать Кузнечик.
⚠️ **Важно:** режим CBC требует ESSIV или аналогичной защиты — иначе возможны атаки на повторяющиеся блоки.

3.3. LUKS1 с GOST 28147-89 (CPA S-box, CBC)

Алгоритм **ГОСТ 28147-89** продолжает использоваться в сертифицированных решениях, особенно в формате LUKS1, где важна совместимость с проверенными средствами защиты.

Пример создания LUKS1-тома с использованием S-блока **CPA** и режима CBC:

```
sudo cryptsetup luksFormat /dev/sdb3 \
  --type luks1 \
  --cipher cbc(gost28147-cpa) \
  --key-size 256 \
  --hash gosthash94
```

Пояснение параметров:

- `--type luks1` — формат LUKS1 необходим для совместимости с некоторыми сертифицированными загрузчиками.
- `cbc(gost28147-cpa)` — режим CBC с блочным шифром ГОСТ 28147-89, S-блок **CryptoPro A**.
- `--key-size 256` — полный ключ ГОСТ 28147-89 (8 × 32-битных подблока).
- `--hash gosthash94` — старая хэш-функция ГОСТ Р 34.11-94, используется при генерации ключа из пароля.

CPA — это один из утверждённых CryptoPro S-блоков. Доступны также: `cpb`, `cpc`, `cpd`, `tc26z`.

⚠️ **Важно:** ГОСТ 28147-89 с CBC рекомендуется использовать только там, где это требуется нормативами или стандартами ФСТЭК.

4. Использование ГОСТ-HMAC и CMAC в LUKS2 (Integrity)

4.1. С `hmac(streebog256)`

LUKS2 поддерживает режимы с проверкой целостности с помощью подсистемы **`dm-integrity`**. Для ГОСТ можно использовать `HMAC` на базе **Streebog 256** (ГОСТ Р 34.11-2012).

Пример команды для создания тома с шифрованием и встроенной защитой целостности:

```
sudo cryptsetup luksFormat /dev/sdb4 \
  --type luks2 \
  --cipher kuznyechik-xts-plain64 \
  --integrity hmac(streebog256) \
  --sector-size 4096 \
  --key-size 512 \
  --hash streebog256
```

Пояснение параметров:

- `--integrity hmac(streebog256)` — включение проверки целостности с использованием HMAC на базе ГОСТ-хеша.
- `--sector-size 4096` — размер сектора, на который будет применяться MAC (рекомендуется 4K для современных дисков).
- `--cipher kuznyechik-xts-plain64` — блочное шифрование с Кузнечиком.
- `--key-size 512` — двойной 256-битный ключ для режима XTS.

✅ **Преимущество:** Защита от аппаратных ошибок и атак на манипуляцию данными в диске без доступа к ключу.
⚠️ **Требование:** ядро должно поддерживать `dm-integrity` и HMAC на основе `streebog256`.
💡 Рекомендуется проверять статус после форматирования с помощью:

```
cryptsetup status luks_gost
```

4.2. С `cmac(kuznyechik)` (если `dm-integrity` поддерживает CMAC)

В некоторых сборках **НАЙС.ОС** поддерживается использование `CMAC` вместо HMAC для контроля целостности. CMAC обеспечивает более компактную и быструю

проверку, особенно при использовании сертифицированных шифров вроде Кузнечика.

Пример команды:

```
sudo cryptsetup luksFormat /dev/sdb5 \
  --type luks2 \
  --cipher kuznyechik-xts-plain64 \
  --integrity cmac(kuznyechik) \
  --sector-size 4096
```

Объяснение параметров:

- `--integrity cmac(kuznyechik)` — контроль целостности с использованием CMAC на шифре Кузнечик.
- `--sector-size 4096` — размер сектора, для которого рассчитывается CMAC.
- `--cipher kuznyechik-xts-plain64` — основной шифр данных (двойной ключ 256+256).

⚠️ **Поддержка CMAC в dm-integrity может отсутствовать** в ядрах по умолчанию. Убедитесь, что модуль включён, или пересоберите ядро с нужной опцией.
Для теста можно использовать `bash` + `dmsetup table` и `cryptsetup status` после открытия устройства.

5. Расширенные примеры с `gost28147-*`

5.1. CNT-режим (CTR-like) с `tc26z`

Алгоритм **ГОСТ 28147-89** с параметрическим S-блоком `tc26z` может применяться в режиме `CNT` — аналог режима CTR (счётчик). Такой режим хорошо подходит для потокового шифрования и устойчив к повторяющимся блокам.

Пример создания LUKS2-тома с CNT-режимом:

```
sudo cryptsetup luksFormat /dev/sdb6 \
  --type luks2 \
  --cipher cnt(gost28147-tc26z) \
  --key-size 256 \
  --hash gosthash94
```

Объяснение параметров:

- `cnt(gost28147-tc26z)` — режим счётчика (CTR-like) с использованием S-блока `tc26z`.
- `--key-size 256` — ГОСТ 28147 требует 256-битный ключ.

- `--hash gosthash94` — ГОСТ Р 34.11-94 в качестве KDF.

✅ **Плюс:** CNT-режим не требует обратной связи (как CBC), а значит устойчив к ошибкам в одном блоке.
⚠️ Не используйте CNT-режим без уникальных nonce/IV при каждом монтировании — повторные IV делают его уязвимым.

5.2. Использование S-блоков CPA, CPB, CPC, CPD

ГОСТ 28147-89 допускает использование разных **S-блоков (подстановочных таблиц)**. В `cryptsetup` через `dm-crypt` поддерживаются следующие варианты:

- **CPA** — CryptoPro A (наиболее распространённый в сертифицированных решениях)
- **CPB** — CryptoPro B
- **CPC** — CryptoPro C
- **CPD** — CryptoPro D

Эти S-блоки определяют конкретную таблицу замен в алгоритме ГОСТ 28147, что влияет на совместимость с внешними системами (например, с аппаратными криптопроцессорами или ГОСТ-сертифицированными библиотеками).

Примеры команд:

CBC с **CPB**:

```
cryptsetup luksFormat /dev/sdb7 \
  --type luks1 \
  --cipher cbc(gost28147-cpb) \
  --key-size 256 \
  --hash gosthash94
```

CBC с **CPC**:

```
cryptsetup luksFormat /dev/sdb8 \
  --type luks1 \
  --cipher cbc(gost28147-cpc) \
  --key-size 256 \
  --hash gosthash94
```

CNT с **CPD**:

```
cryptsetup luksFormat /dev/sdb9 \
  --type luks2 \
  --cipher cnt(gost28147-cpd) \
  --key-size 256 \
  --hash streebog256
```

Примечание: Выбор S-блока зависит от требований нормативных документов и взаимодействия с другими системами. Некоторые блоки (например, `tc26z`) считаются современными и рекомендованными.

6. Тестирование производительности ГОСТ

После настройки томов с ГОСТ-алгоритмами важно проверить производительность шифрования и фактическую скорость работы с диском. Ниже представлены основные инструменты для тестирования.

📊 `cryptsetup benchmark`

Оценивает производительность шифрования и хеширования на уровне CPU и Crypto API:

```
cryptsetup benchmark --cipher kuznyechik-xts-plain64
```

Результат покажет скорость в МБ/с для операций `encryption` и `decryption`.

🔍 `dmsetup table` и `dmsetup status`

Позволяет проверить, какие модули `device-mapper` задействованы и как сконфигурирован том:

```
sudo dmsetup table
sudo dmsetup status
```

В выводе можно увидеть активные цепочки: шифрование, целостность, base device.

🛰 Замер реальной скорости I/O

🔹 Быстрый тест чтения (hdparm):

```
sudo hdparm -t /dev/mapper/luks_gost
```

🔹 Простая запись (dd):

```
dd if=/dev/zero of=/mnt/testfile bs=1M count=512 status=progress
```

🔹 Гибкий тест с `fio`:

```
fio --name=test --filename=/mnt/testfile \
  --bs=4k --rw=randwrite --size=256M --ioengine=libaio --direct=1
```

⚠️ Для достоверности отключите кэширование ОС и убедитесь, что том не находится в idle.
✅ **Рекомендация:** сравнивайте скорость до и после включения шифрования — это покажет реальные накладные расходы от ГОСТ-алгоритма.

7. Практика: Создание тома, монтирование и использование

После форматирования устройства с использованием ГОСТ-алгоритмов (например, `kuznyechik-xts-plain64`), необходимо подключить и использовать зашифрованный том в системе.

🔓 Открытие зашифрованного тома

Открываем LUKS-том и назначаем ему имя:

```
sudo cryptsetup luksOpen /dev/sdb1 luks_gost
```

После ввода корректного пароля устройство будет доступно как `/dev/mapper/luks_gost`.

Форматирование файловой системы

Создаём файловую систему (например, ext4) на открытом устройстве:

```
sudo mkfs.ext4 /dev/mapper/luks_gost
```

💡 Вы можете использовать любую подходящую ФС: `ext4`, `xfs`, `btrfs` — в зависимости от

задачи.

📎 Монтирование зашифрованного тома

Монтируем файловую систему в нужную точку:

```
sudo mount /dev/mapper/luks_gost /mnt
```

Теперь том готов к использованию как обычный диск. Все операции будут прозрачно шифроваться в реальном времени.

✅ Готово! Вы можете использовать `/mnt` как обычное хранилище с ГОСТ-шифрованием.

8. Безопасное удаление и закрытие

После завершения работы с зашифрованным томом его необходимо корректно размонтировать и закрыть, чтобы исключить потерю данных и обеспечить защиту от несанкционированного доступа.

📤 Шаг 1: Размонтирование файловой системы

Прежде всего, нужно размонтировать точку монтирования:

```
sudo umount /mnt
```

⚠️ Убедитесь, что в каталоге `/mnt` больше не используются файлы или процессы — иначе размонтирование не удастся.

🔐 Шаг 2: Закрытие тома LUKS

После успешного размонтирования закройте зашифрованный том:

```
sudo cryptsetup luksClose luks_gost
```

Устройство `/dev/mapper/luks_gost` исчезнет и станет недоступным до следующего подключения.

✅ Том закрыт — шифрованные данные снова находятся под защитой.

9. Интеграция в систему

Для удобства и автоматизации можно настроить постоянное подключение зашифрованного тома через `/etc/crypttab` и `/etc/fstab`, а также встроить поддержку в загрузку через initramfs.

🔄 Автоматическое открытие через `/etc/crypttab`

Добавьте строку в `/etc/crypttab`:

```
luks_gost  UUID=xxx-yyy-zzz  none  luks
```

Где `luks_gost` — имя маппинга (`/dev/mapper/luks_gost`), а UUID можно получить через `blkid`. Параметр `none` означает интерактивный ввод пароля.

📄 Автомонтирование через `/etc/fstab`

Добавьте строку в `/etc/fstab` для монтирования после открытия:

```
/dev/mapper/luks_gost  /mnt/gostdata  ext4  defaults,noatime  0 2
```

Создайте каталог `/mnt/gostdata`, если он ещё не существует:

```
sudo mkdir -p /mnt/gostdata
```

Интеграция с initramfs

Для автоматического открытия при загрузке необходимо обновить initramfs:

- **dracut (RHEL/ALT/НАЙС.ОС):** `sudo dracut -f`
- **mkinitcpio (Arch):** `sudo mkinitcpio -P`

Убедитесь, что в конфиге initramfs включён модуль `crypt` и прописан нужный `crypttab`.

🖥️ Сценарии для серверов

- Использование ключевых файлов в `/etc/keys` с ограничениями доступа
- Открытие тома через systemd unit с `RequiresMountsFor`
- Удалённый unlock через dropbear или Tang (системы удалённой разблокировки)

💡 Для серверов лучше избегать ручного ввода паролей при загрузке — используйте

ключи, TPM или сеть.

10. Частые ошибки и отладка

- ❗ **No such cipher** — ядро не поддерживает указанный алгоритм. Проверьте через `/proc/crypto` и наличие модулей (`modprobe`).
- ❗ **device-mapper: cipher key size incorrect** — неверная длина ключа или несовместимость режима с указанным алгоритмом. Проверьте параметры `--key-size`.
- 🛠 Используйте `dmesg` для чтения сообщений ядра при ошибках монтирования или открытия.
- Применяйте `strace cryptsetup ...` для выявления проблем при запуске.
- 🎁 Убедитесь, что нужные модули загружены: `modprobe kuznyechik`, `modprobe streebog512` и др.

📚 Приложение A: Список поддерживаемых имён алгоритмов

Имя	Тип	Пример использования	Модуль ядра
kuznyechik-xts-plain64	Шифр (128-бит)	--cipher kuznyechik-xts-plain64	kuznyechik
magma-cbc-essiv:streebog512	Шифр (64-бит)	--cipher magma-cbc-essiv:streebog512	magma, streebog512
cbc(gost28147-cpa)	Шифр (S-box CPA)	--cipher cbc(gost28147-cpa)	gost28147-cpa
cnt(gost28147-tc26z)	CTR-like режим	--cipher cnt(gost28147-tc26z)	gost28147-tc26z
hmac(streebog256)	MAC	--integrity hmac(streebog256)	streebog256
cmac(kuznyechik)	MAC	--integrity cmac(kuznyechik)	kuznyechik

Расшифровка режимов:

- `cnt()` — аналог CTR (счётчик);
- `cbc()` — шифрование с цепочкой блоков;
- `xts` — стандартный режим для дисковых шифров;
- `essiv:` — расширенный CBC с защитой IV (salted hash).

📚 Приложение B: Ссылки и стандарты

- 📖 **ГОСТ Р 34.12–2015** — блочные шифры: Магма, Кузнечик
- 📖 **ГОСТ Р 34.11–2012** — хэш-функции (Стрибог)
- 📖 **ГОСТ 28147–89** — классический блочный шифр с S-блоками
- 🌐 **RFC 7801** — GOST block cipher in TLS

📖 Приложение C: Список поддерживаемых имён алгоритмов ГОСТ в `/proc/crypto`

Категория	Имя алгоритма	Пример использования	Комментарий
Блочные шифры	`kuznyechik`	`kuznyechik-xts-plain64`	ГОСТ Р 34.12-2015, 128-бит
	`magma`	`magma-cbc-essiv:streebog512`	ГОСТ Р 34.12-2015, 64-бит
	`gost28147-{tc26z,cpa,cpb,cpc,cpd}`	`cbc(gost28147-cpa)`	ГОСТ 28147-89 с различными S-box
Режимы шифрования	`cbc()`, `xts()`	`kuznyechik-xts-plain64`	Рекомендуемые режимы для дисков
	`cfb()`, `ecb()`	`cfb(gost28147-cpb)`	Не все режимы безопасны (ecb — редко)
	`cnt()`	`cnt(gost28147-tc26z)`	Аналог CTR-режима
	`essiv:`	`magma-cbc-essiv:streebog512`	Защита от повторов IV

Категория	Имя алгоритма	Пример использования	Комментарий
MAC / имитовставки	`cmac(kuznyechik)`	`--integrity cmac(kuznyechik)`	На базе Кузнечика
	`cmac(magma)`	`--integrity cmac(magma)`	На базе Магмы
	`gost28147imit-{tc26z,cpa,...}`	—	Классическая имитовставка (MAC)
	`hmac(streebog256)`	`--integrity hmac(streebog256)`	На базе ГОСТ Р 34.11-2012
	`hmac(gosthash94)`	`--integrity hmac(gosthash94)`	ГОСТ Р 34.11-94 (устаревший)
Хеш-функции	`streebog256`, `streebog512`	`--hash streebog256`	ГОСТ Р 34.11-2012
	`gosthash94`	`--hash gosthash94`	ГОСТ Р 34.11-94

Все алгоритмы перечислены в `/proc/crypto` и могут использоваться в `cryptsetup` через параметры `--cipher`, `--hash`, `--integrity`.

🖊 Приложение D: Комбинации ГОСТ-алгоритмов — стойкость, производительность и баланс

Категория	Параметры	Описание	Когда использовать

🔒 Максимальная стойкость	`--cipher kuznyechik-xts-plain64`
	`--key-size 512`
	`--hash streebog512`
	`--integrity hmac(streebog256)`
		Современный ГОСТ-шифр (Кузнечик), стойкий хеш (Стрибог), защита целостности через HMAC.	Конфиденциальные серверы, сертифицируемые ОС, защищённые станции

⚡ Самая высокая производительность	`--cipher cbc(gost28147-cpa)`
	`--key-size 256`
	`--hash gosthash94`
		Быстрый шифр и хеш, минимальные накладные расходы. Без контроля целостности.	Старые системы, низкопроизводительные устройства, тестовые среды

Категория	Параметры	Описание	Когда использовать

⚖️ **Баланс стойкости и скорости**	`--cipher kuznyechik-xts-plain64`
	`--key-size 512`
	`--hash streebog256`
		Кузнечик в XTS без integrity-защиты. Хороший компромисс между безопасностью и скоростью.	Рабочие станции, ноутбуки, серверы без жёстких требований по MAC

Наивысшая совместимость	`--cipher magma-cbc-essiv:streebog512`
	`--key-size 256`
	`--hash streebog512`
		Магма (ГОСТ 64-бит) + ESSIV. Поддерживается большинством дистрибутивов с ГОСТ-ядерными патчами.	Старое оборудование, системы с ограниченной поддержкой, LUKS2/GRUB

💡 **Примечание:** В комбинациях с `--integrity` требуется поддержка `dm-integrity` в ядре и initramfs.