Применение алгоритмов ГОСТ в cryptsetup

Применение алгоритмов ГОСТ в cryptsetup (LUKS/dm-crypt) в НАЙС.ОС

Документ: NICEOS-SEC-CRYPTSETUP-GOST · Редакция: 1.0 · Статус: техническая документация · Язык: RU
Примечание по реализации
cryptsetup является пользовательским интерфейсом к dm-crypt и использует криптографические реализации, предоставляемые ядром Linux (Crypto API). Параметры алгоритмов задаются при создании/открытии тома.

1 Область применения

Настоящий документ устанавливает порядок применения алгоритмов ГОСТ при шифровании блочных устройств и файловых контейнеров с использованием cryptsetup (форматы LUKS1/LUKS2) и подсистемы dm-crypt в ОС НАЙС.ОС.

Документ ориентирован на системных администраторов, инженеров эксплуатации и разработчиков, выполняющих тестирование, интеграцию и валидацию криптографических параметров в инфраструктуре. Для юридически значимых сценариев и объектов, подпадающих под обязательную сертификацию, требуется применение сертифицированных СКЗИ и выполнение регуляторных требований в соответствии с принятой моделью угроз и политикой безопасности организации.

2 Нормативные ссылки

- Документация cryptsetup (опции LUKS, cipher, PBKDF, служебные команды).
- Сведения о блочном шифровании данных на дисках и требуемых режимах работы (общие положения).

- Ограничения 64-битных блочных шифров при обработке больших объемов данных (эффект дня рождения, практические атаки).

3 Термины и сокращения

dm-crypt	Механизм ядра Linux для шифрования блочных устройств.
LUKS	Формат разметки шифрованных томов (ключевые слоты, метаданные, политика KDF).
PBKDF	Функция выработки ключа из пароля (PBKDF2, Argon2).
XTS	Режим шифрования блочных устройств, применяемый для дисков (обычно с 128-битным блоком).
IV/plain64	Способ формирования вектора инициализации для секторного шифрования (счетчик с 64-битным номером сектора).

4 Архитектура решения

cryptsetup управляет созданием и открытием LUKS-томов и передает параметры алгоритмов в dm-crypt. Реальное шифрование выполняется в ядре Linux через Crypto API. Наличие/отсутствие конкретного алгоритма (например, Кузнечик, Магма) определяется конфигурацией ядра и доступными криптомодулями.

Схема потоков данных (логическая)

1. Администратор вызывает cryptsetup luksFormat для инициализации LUKS и задания cipher-параметров.
2. cryptsetup формирует метаданные LUKS (ключевые слоты, PBKDF-политика, параметры шифрования).
3. При открытии тома cryptsetup создает device-mapper устройство /dev/mapper/<name> с параметрами dm-crypt.
4. Шифрование/расшифровка выполняется ядром, прозрачно для файловой системы и приложений.

5 Ограничения и выбор алгоритмов

5.1 Рекомендуемый профиль для шифрования дисков

Для шифрования блочных устройств (диски/разделы/тома) рекомендуется использовать режим XTS, поскольку он предназначен для секторного шифрования и является типовым профилем для dm-crypt/LUKS. Практика дискового шифрования предполагает применение режимов, адаптированных под повторяющуюся структуру данных на носителях.

5.2 Применимость Магмы и ГОСТ 28147 для больших томов

Магма и ГОСТ 28147 относятся к 64-битным блочным шифрам. Для больших объемов данных 64-битный блок приводит к росту вероятности коллизий на уровне блоков (эффект дня рождения) и ухудшению стойкости при больших объемах зашифрованного трафика/данных. Известные практические атаки на 64-битные блочные шифры (на примере 3DES) демонстрируют риски при больших объемах данных.

Требование эксплуатации
Для дискового шифрования томов общего назначения следует выбирать 128-битные блочные шифры (в контексте ГОСТ — Кузнечик), а применение 64-битных блочных шифров ограничивать узкими наследуемыми сценариями и малыми объемами данных с отдельной оценкой рисков.

5.3 LUKS2 как базовый формат

Для новых развертываний рекомендуется использовать LUKS2 (включая параметры PBKDF и управляемость ключевых слотов). Выбор формата задается опцией --type luks2.

6 Проверка доступности ГОСТ-алгоритмов

6.1 Проверка криптоалгоритмов, доступных ядру

Список зарегистрированных алгоритмов Crypto API доступен через /proc/crypto. Верификацию наличия Кузнечика/Магмы/ГОСТ 28147 следует выполнять до создания тома.

```
# Просмотр доступных алгоритмов ядра
grep -Ei 'kuznyechik|grasshopper|magma|gost28147|gost' /proc/crypto | head -n 80

# Для удобства: вывести имена "name" и тип "type"
awk '
  $1=="name" {name=$3}
  $1=="type" {type=$3; if (name ~ /(kuznyechik|grasshopper|magma|gost)/i) print name, type}
' /proc/crypto | sort -u
```

6.2 Проверка поддержки cipher на уровне cryptsetup

cryptsetup не «добавляет» алгоритмы, а использует то, что поддерживает ядро. Для тестирования доступности и производительности применяют cryptsetup benchmark и/или пробное открытие тома. Опции формирования тома и выбора cipher определены документацией cryptsetup.

```
cryptsetup --version
cryptsetup benchmark | head -n 120
```

Критерий готовности
Система считается готовой к применению ГОСТ в dm-crypt при выполнении одновременно: (1) наличие алгоритмов в /proc/crypto; (2) успешное создание и открытие LUKS-тома с выбранным cipher; (3) корректное отображение параметров cipher в cryptsetup status.

7 Правила именования cipher в cryptsetup

Значение --cipher задает строку cipher в формате, понятном dm-crypt. Типовой вид: <алгоритм>-<режим>-<iv>, например aes-xts-plain64. Опция --key-size задает размер ключа в битах.

Практический порядок подбора строки cipher для ГОСТ

1. Определить имя базового алгоритма по /proc/crypto (например, kuznyechik или

magma).
2. Определить доступный режим (для дисков — xts при наличии реализации).
3. Выбрать IV-генератор (для LUKS/dm-crypt типовой вариант — plain64).
4. Сформировать строку: kuznyechik-xts-plain64 (пример) и проверить пробным созданием LUKS.

Примечание
Реальные имена алгоритмов и поддерживаемые режимы зависят от версии ядра и конфигурации Crypto API. Перед эксплуатацией требуется подтверждение на целевой системе по п.6.

8 LUKS2 + Кузнечик (рекомендуемый профиль)

8.1 Предварительные условия

- Определено блочное устройство: /dev/sdXn или /dev/nvme0n1pX.
- Выполнено резервное копирование данных (операция luksFormat уничтожает данные на устройстве).
- Алгоритм Кузнечик доступен ядру (см. раздел 6).

8.2 Инициализация LUKS2

Ниже приведен базовый пример создания LUKS2-тома с режимом XTS. Для XTS требуется удвоенный ключ (две независимые половины). Соответственно, если базовый ключ алгоритма 256 бит, для XTS задается 512 бит (пример). Параметры задаются через --cipher, --key-size, --type и PBKDF-опции.

```
# Пример: Кузнечик-XTS (cipher-строку подтвердить по /proc/crypto)
DEV="/dev/nvme0n1p3"
NAME="luks_gost"

# Рекомендуемый формат: LUKS2 + Argon2id (по умолчанию для LUKS2 часто используется Argon2)
# Параметр --iter-time задает целевое время вычисления PBKDF (мс).
cryptsetup luksFormat \
 --type luks2 \
 --cipher kuznyechik-xts-plain64 \
 --key-size 512 \
 --pbkdf argon2id \
 --iter-time 5000 \
 "$DEV"
```

Контроль правильности cipher
Если ядро не поддерживает указанный cipher, luksFormat завершится ошибкой. В этом случае требуется уточнить реальное имя алгоритма/режима по /proc/crypto и повторить операцию.

8.3 Открытие тома и создание файловой системы

```
DEV="/dev/nvme0n1p3"
NAME="luks_gost"

cryptsetup open "$DEV" "$NAME"
# После открытия устройство доступно как /dev/mapper/luks_gost
mkfs.ext4 -L DATA_GOST "/dev/mapper/$NAME"

mkdir -p /mnt/data_gost
mount "/dev/mapper/$NAME" /mnt/data_gost

# Проверка статуса
cryptsetup status "$NAME"
```

8.4 Завершение работы

```
NAME="luks_gost"
umount /mnt/data_gost
cryptsetup close "$NAME"
```

9 Магма/ГОСТ 28147 (наследуемые профили)

Применение Магмы и ГОСТ 28147 для дискового шифрования общего назначения не рекомендуется из-за 64-битного блока и рисков при больших объемах данных. Допускается применение для ограниченных сценариев (малые контейнеры, лабораторные стенды, совместимость с наследуемыми требованиями) при наличии формализованной оценки рисков.

9.1 Пример (лабораторный): LUKS2 + Магма в CBC

CBC для дисков является компромиссным вариантом и требует внимательной настройки IV (например, ESSIV). Пример приведен исключительно как иллюстрация механизма выбора cipher-строки.

```
DEV="/dev/loop10"
NAME="luks_magma_lab"

# Создание test-файла и loop-устройства (пример лабораторной проверки)
dd if=/dev/zero of=/var/tmp/magma.img bs=1M count=256
losetup -fP /var/tmp/magma.img
DEV="$(losetup -j /var/tmp/magma.img | awk -F: '{print $1}')"

# Вариант cipher-строки зависит от реализации ядра.
cryptsetup luksFormat \
 --type luks2 \
 --cipher magma-cbc-essiv:sha256 \
 --key-size 256 \
 --pbkdf argon2id \
 --iter-time 3000 \
 "$DEV"

cryptsetup open "$DEV" "$NAME"
mkfs.ext4 "/dev/mapper/$NAME"
mount "/dev/mapper/$NAME" /mnt
```

Контроль допустимости
Для производственных систем следует использовать Кузнечик (128-битный блок) и режим XTS при наличии поддержки в ядре, как базовый профиль для dm-crypt.

10 Эксплуатационные операции LUKS

Команды управления LUKS (добавление ключей, удаление ключей, резервное копирование заголовка) выполняются средствами cryptsetup. Полный перечень команд и назначение опций определяется документацией cryptsetup.

10.1 Просмотр параметров LUKS

```
DEV="/dev/nvme0n1p3"
cryptsetup luksDump "$DEV"
```

10.2 Резервное копирование заголовка LUKS

```
DEV="/dev/nvme0n1p3"
cryptsetup luksHeaderBackup "$DEV" --header-backup-file /root/luks-header.backup

# Восстановление (операция высокого риска; применяется по регламенту восстановления)
```

```
# cryptsetup luksHeaderRestore "$DEV" --header-backup-file /root/luks-header.backup
```

10.3 Управление ключевыми слотами

```
DEV="/dev/nvme0n1p3"

# Добавление дополнительного ключа (новый пароль/ключевой материал)
cryptsetup luksAddKey "$DEV"

# Смена ключа (замена пароля для существующего слота)
cryptsetup luksChangeKey "$DEV"

# Удаление ключа по вводу удаляемого пароля (аккуратно: не потерять последний ключ)
cryptsetup luksRemoveKey "$DEV"
```

Требование непрерывности доступа
Любые операции со слотами выполнять только при наличии минимум одного альтернативного работающего ключа и актуального резервного заголовка LUKS. Потеря всех ключей приводит к необратимой потере данных.

11 Авторазблокировка при загрузке

Для автоматического открытия томов на этапе загрузки используется файл /etc/crypttab (в системах с systemd) и соответствующие записи в /etc/fstab. Вариант подключения зависит от политики безопасности: пароль вручную, ключевой файл, аппаратный токен, сетевой агент и др.

11.1 Пример crypttab (ввод пароля вручную)

```
# /etc/crypttab
# <name>    <source device>     <keyfile>   <options>
luks_gost   /dev/nvme0n1p3      none        luks
```

11.2 Пример fstab

```
# /etc/fstab
/dev/mapper/luks_gost  /data  ext4  defaults  0  2
```

Примечание

В случае применения ключевых файлов требуется регламент хранения (права доступа, изоляция, резервирование), а также модель угроз (риски компрометации root/backup).

12 Контроль и аудит параметров шифрования

12.1 Статус открытого тома

```
NAME="luks_gost"
cryptsetup status "$NAME"
```

12.2 Проверка таблицы device-mapper

```
NAME="luks_gost"
dmsetup table "/dev/mapper/$NAME"
dmsetup info "/dev/mapper/$NAME"
```

12.3 Контроль соответствия политике

Для контроля соответствия утвержденной криптополитике рекомендуется фиксировать: (1) тип LUKS (LUKS2); (2) PBKDF (Argon2id/PBKDF2 и параметры); (3) cipher-строку dm-crypt; (4) размер ключа; (5) режим работы (XTS/другой). Параметры извлекаются через luksDump, status, dmsetup.

13 Производительность и тестирование

13.1 Сравнительное тестирование cipher

```
# Общий бенчмарк (выводит набор алгоритмов и производительность)
cryptsetup benchmark
```

13.2 Прикладной тест чтения/записи

```
# Пример: измерение скорости записи на смонтированный том
dd if=/dev/zero of=/mnt/data_gost/test.bin bs=64M count=16 oflag=direct status=progress
```

```
sync
dd if=/mnt/data_gost/test.bin of=/dev/null bs=64M iflag=direct status=progress
```

Требование корректности тестов
Для сравнения алгоритмов тестировать одинаковые условия: одинаковый носитель, одинаковый размер блока, одинаковый режим монтирования, отключение кэшей (direct), повторяемость прогонов.

14 Рекомендации по усилению

- Формат: применять LUKS2 для новых томов.
- Режим: для дисков применять XTS при поддержке ядра; избегать 64-битных блочных шифров для больших томов.
- PBKDF: увеличить --iter-time до значения, соответствующего эксплуатационной политике (например, 3000–10000 мс) с учетом класса оборудования.
- Пароли: использовать уникальные длинные пароли; запретить повторное использование; хранение — по регламенту.
- Резервирование: выполнять регулярный backup заголовков LUKS и проверку восстановления на стенде.
- Аудит: фиксировать параметры cipher/PBKDF в конфигурационной базе и в журнале изменений.

Приложение А (справочное) — шаблоны команд

А.1 Быстрая диагностика наличия ГОСТ-алгоритмов

```
echo "[1] cryptsetup version"
cryptsetup --version

echo "[2] kernel crypto algorithms (filtered)"
grep -Ei 'kuznyechik|grasshopper|magma|gost28147|gost' /proc/crypto | sed -n '1,120p'

echo "[3] cryptsetup benchmark (top)"
cryptsetup benchmark | sed -n '1,120p'
```

А.2 Шаблон создания LUKS2 (подстановочный)

```
DEV="/dev/XXX"
NAME="luks_YYY"
```

```
CIPHER="kuznyechik-xts-plain64"   # уточнить по /proc/crypto
KEYBITS="512"                     # для XTS обычно 2×ключ

cryptsetup luksFormat --type luks2 --cipher "$CIPHER" --key-size "$KEYBITS" --pbkdf argon2id --iter-time 5000 "$DEV"
cryptsetup open "$DEV" "$NAME"
cryptsetup status "$NAME"
```