nettle в НАЙС.ОС - полная поддержка ГОСТ

ГОСТ Р 34.12-2015: симметричные шифры «Кузнечик» и «Магма» в НАЙС.ОС (Nettle/GnuTLS)

Техническое описание реализации и применения. Объект: системные криптографические библиотеки `libnettle` и `libgnutls`. Назначение: обеспечение доступности современных ГОСТ-блочных шифров и режимов применения в прикладных TLS/криптосценариях.

1. Введение

Настоящий документ устанавливает сведения о симметричных блочных шифрах «Кузнечик» и «Магма», определённых в **ГОСТ Р 34.12-2015**, а также описывает их интеграцию и доступность в составе **НАЙС.ОС** на уровне библиотек `libnettle` и `libgnutls`.

«Кузнечик» является 128-битным блочным шифром. «Магма» является 64-битным блочным шифром, совместимым по архитектурной идеологии с ГОСТ 28147-89, при этом параметры подстановки и описание алгоритма стандартизованы в ГОСТ Р 34.12-2015. ГОСТ 28147-89 продолжает встречаться в наследуемых системах; «Кузнечик» и «Магма» применяются как современные стандартизованные примитивы для новых разработок и протокольных профилей.

Примечание (о режимах работы и MAC): ГОСТ Р 34.12-2015 определяет блочные шифры. Режимы применения (например, CTR-ACPKM) и имитовставка на базе блочного шифра (например, OMAC/CMAC-подобные конструкции) задаются отдельными документами/профилями. В TLS-контексте применяются стандартизованные наборы шифров и режимы, реализованные на стороне `GnuTLS`.

В составе **НАЙС.ОС** реализована штатная поддержка «Кузнечика» и «Магмы» в libnettle с последующей доступностью в потребителях libgnutls. Это обеспечивает возможность применения ГОСТ Р 34.12-2015 в прикладных сценариях, использующих стек GnuTLS/Nettle, без установки внешних криптомодулей и без пересборки сторонних пакетов.

Инженерная цель: обеспечить воспроизводимую криптографическую функциональность «из коробки» для стендов разработки и эксплуатации (включая TLS-тестирование, проверку совместимости, регрессию и диагностику), снижая стоимость интеграции и ускоряя ввод сервисов в тестовые контуры.

1.1. Область применения

- использование «Кузнечика» и «Магмы» в приложениях, использующих libgnutls (TLS/криптопротоколы);
- использование «Кузнечика» и «Магмы» в приложениях, вызывающих libnettle напрямую (встроенная криптография, утилиты, агенты);
- контроль доступности и корректности реализаций по тестовым векторам и через TLS-рукопожатие в ГОСТ-профиле;
- оценка производительности (benchmark) и анализ корректности режимов применения.

1.2. Контроль готовности среды (оперативная проверка)

```
# Версии библиотек (контроль факта применения Nettle/GnuTLS)
gnutls-cli --version
certtool --version

# Перечень алгоритмов/возможностей в GnuTLS (фактический список поддерживаемого на хосте)
gnutls-cli --list | sed -n '1,260p'
```

2. Сведения об апстриме: Nettle и GnuTLS

Nettle — криптографическая библиотека общего назначения, ориентированная на предоставление примитивов (шифры, хеш-функции, подписи) и низкоуровневых интерфейсов, удобных для интеграции в прикладные и протокольные реализации. **GnuTLS** — библиотека реализации TLS/DTLS, использующая Nettle как один из

криптографических бэкендов.

Архитектурно, доступность конкретного блочного шифра в TLS-реализации GnuTLS зависит от наличия:

- реализации шифра на уровне криптобэкенда (в данном случае — Nettle);
- включения поддержки на уровне GnuTLS (регистрация алгоритмов, приоритеты, ограничения версий протокола);
- наличия совместимых наборов шифров (cipher suites) и параметров профиля.

2.1. Наличие ГОСТ-примитивов в апстриме (контекст)

В типовых апстримных поставках криптобиблиотек встречается поддержка части ГОСТ-примитивов (например, для хеширования и подписи). При этом отсутствие реализации конкретного блочного шифра в Nettle приводит к невозможности его использования в GnuTLS даже при наличии идентификаторов алгоритмов на уровне заголовков/внутренних таблиц.

Техническая причина блокировки: при отсутствии реализаций «Кузнечика»/«Магмы» в Nettle GnuTLS не может предоставить рабочие алгоритмы шифрования/имитовставки в составе ГОСТ-профилей TLS, поскольку не имеет примитивов для выполнения криптоопераций.

2.2. Признаки «частично объявлено, но недоступно»

Типовой признак несоответствия между объявлением и фактической реализацией — наличие упоминаний алгоритмов/режимов в исходных текстах или заголовках при отсутствии их в фактическом выводе перечисления поддерживаемых алгоритмов и невозможности выбора приоритета, включающего эти алгоритмы.

```
# Фактический контроль осуществляется по выводу gnutls-cli --list и результатам рукопожатия.
gnutls-cli --list | grep -Ei 'kuznyechik|magma|gost' || true
```

3. Реализация в НАЙС.ОС: интеграция «Кузнечика» и «Магмы» в libnettle

В **НАЙС.ОС** выполнена интеграция блочных шифров «Кузнечик» и «Магма» на уровне `libnettle`. Реализация включает криптографическое ядро шифров, интерфейсы

контекста, регистрацию в мета-описателях (meta-ciphers) и тестовое покрытие.

3.1. Параметры алгоритмов (нормативные характеристики)

Алгоритм	Размер блока	Размер ключа	Назначение	Примечание
Кузнечик	128 бит	256 бит	симметричное шифрование данных	современный блочный шифр ГОСТ Р 34.12-2015
Магма	64 бит	256 бит	симметричное шифрование данных	блочный шифр ГОСТ Р 34.12-2015 (наследуемый класс 64-битных блоков)

3.2. Состав изменений на уровне Nettle (логическая структура)

- реализация шифра «Кузнечик» (контекст ключа, процедуры шифрования/расшифрования блока);
- реализация шифра «Магма» (контекст ключа, процедуры шифрования/расшифрования блока);
- регистрация алгоритмов в мета-таблицах Nettle (для потребителей и для единообразного доступа);
- добавление тестов с контрольными векторами (шифрование/расшифрование, ключевые расписания, регрессия);
- добавление в утилиты производительности (benchmark) для сравнимой оценки на целевых платформах.

Результат: после интеграции «Кузнечик» и «Магма» доступны в системе как штатные криптопримитивы Nettle, что позволяет потребителям (в т.ч. GnuTLS) использовать их без внешних модулей.

3.3. Требования к тестированию (минимальный набор)

Минимально необходимая проверка корректности реализации включает:

- шифрование/расшифрование блока по контрольным векторам ГОСТ Р 34.12-2015;
- проверка устойчивости к типовым ошибкам интерфейса (неинициализированный контекст, некорректные длины буфера);
- регрессионные тесты на неизменность результатов между обновлениями;
- интеграционная проверка через потребителя (GnuTLS), подтверждающая фактическую доступность алгоритмов в TLS-профиле.

4. Состав ГОСТ-стека в НАЙС.ОС на уровне Nettle/GnuTLS

После интеграции «Кузнечика» и «Магмы» криптографический стек ГОСТ на базе Nettle/GnuTLS включает симметричные шифры, функции хеширования, механизмы имитовставки и асимметрические примитивы (в объёме, предусмотренном сборкой и профилем применения). Фактический перечень поддерживаемых алгоритмов определяется выводом `gnutls-cli --list` и профилем приоритетов.

4.1. Перечень категорий

Категория	Компоненты	Назначение
Блочные шифры	`kuznyechik`, `magma` (а также наследуемые при наличии)	конфиденциальность данных и транспортных сеансов
Режимы применения	CTR-ACPKM (в TLS-профиле при наличии), CTR/CBC/иные режимы в потребителях	режимы шифрования на базе блочного шифра
Имитозащита	OMAC/CMAC-подобные конструкции (в TLS-профиле при наличии)	контроль целостности и аутентичности
TLS	`libgnutls` с ГОСТ-профилями и наборами шифров	защищённые каналы (TLS 1.2 в ГОСТ-профиле)

4.2. Контроль доступности по месту (обязательная процедура)

```
# Поиск упоминаний алгоритмов в фактическом перечне возможностей
gnutls-cli --list | grep -Ei 'kuznyechik|magma' || true

# Поиск упоминаний ГОСТ-профилей/наборов шифров
gnutls-cli --list | grep -Ei 'gost|acpkm|omac' || true
```

Примечание: наименование алгоритмов в выводе зависит от версии GnuTLS и внутренней схемы именования. Контроль выполняется по факту: наличие алгоритма в списках и успешное TLS-рукопожатие при включении соответствующего приоритета.

5. Проверка работоспособности: процедуры

5.1. Проверка TLS-рукопожатия в ГОСТ-профиле

Контроль осуществляется путём запуска тестового TLS-сервера на базе `gnutls-serv` и подключения к нему клиентом `gnutls-cli` с фиксированным приоритетом протокола TLS 1.2 и ГОСТ-наборов шифров.

Ограничение протокола: ГОСТ-наборы шифров, как правило, применяются в TLS 1.2-профилях. Для теста рекомендуется явно фиксировать TLS 1.2. Фактическая поддержка определяется сборкой и возможностями используемой версии GnuTLS.

Шаг 1. Генерация тестового ключа/сертификата (для стенда)

Для стендовых испытаний допускается применение самоподписанного сертификата. Для продуктивных контуров используется сертификат от УЦ согласно политике организации.

```
# Пример генерации ключа и самоподписанного сертификата средствами certtool (для тестового сервера)
# Примечание: конкретные параметры ключа/алгоритмов зависят от доступности в текущей сборке GnuTLS.
certtool --generate-privkey --outfile tls-test.key

certtool --generate-self-signed \
  --load-privkey tls-test.key \
  --template <<EOF \
```

```
 --outfile tls-test.crt
cn = "niceos-tls-test"
expiration_days = 365
tls_www_server
dns_name = "localhost"
EOF
```

Шаг 2. Запуск тестового сервера

```
# Запуск сервера. Приоритет задаёт жёсткое ограничение профиля.
# В реальном применении приоритеты должны соответствовать криптополитике организации.
gnutls-serv \
 --port 5555 \
 --x509keyfile tls-test.key \
 --x509certfile tls-test.crt \
 --priority "NONE:+VERS-TLS1.2:+GOST"
```

Шаг 3. Подключение клиентом и фиксация параметров

```
# Подключение с выводом параметров рукопожатия.
gnutls-cli \
 --priority "NONE:+VERS-TLS1.2:+GOST" \
 -p 5555 localhost
```

Критерий успешности: установка соединения и отображение выбранного набора шифров, содержащего ГОСТ-компоненты. В отчёте фиксируются: версия GnuTLS, приоритетная строка, выбранный ciphersuite, а также дамп параметров сеанса.

5.2. Проверка производительности (оценочная)

Для оценочной проверки производительности используются штатные бенчмарки библиотеки (если установлены) либо целевые микротесты в прикладном контуре. Результаты рассматриваются как инженерные и не заменяют профильных испытаний.

```
# Пример: поиск упоминаний в бенчмарке (при наличии инструмента в поставке)
nettle-benchmark 2>/dev/null | grep -Ei 'kuznyechik|magma' || true
```

6. Прямое использование libnettle: пример

прикладного кода

Настоящий раздел предназначен для разработчиков, использующих libnettle напрямую (встроенные агенты, сервисы, утилиты). Пример носит демонстрационный характер и показывает структуру вызовов: инициализация контекста, установка ключа, шифрование блока. Режимы шифрования (CTR/CBC/иные) реализуются поверх блочного шифра согласно выбранному профилю.

Примечание: точные имена структур и функций зависят от версии Nettle и способа интеграции. При разработке следует использовать заголовки, установленные в системе, и собирать код в целевом окружении НАЙС.ОС.

6.1. Демонстрация: шифрование одного блока «Кузнечиком»

```
#include <stdint.h>
#include <string.h>

/*
 * ВНИМАНИЕ:
 * Это демонстрационный каркас. Используйте фактические заголовки Nettle и реальные API-символы,
 * доступные в вашей версии libnettle в НАЙС.ОС.
 *
 * Цель: показать требуемые этапы: ключ -> контекст -> encrypt_block().
 */

int main(void) {
  /* Примерные размеры: Кузнечик: ключ 32 байта, блок 16 байт */
  uint8_t key[32] = {0};
  uint8_t in[16]  = {0};
  uint8_t out[16] = {0};

  /* Контекст шифра (имя зависит от версии Nettle) */
  /* struct kuznyechik_ctx ctx; */

  /* 1) Заполнение ключа и входного блока данными теста */
  memset(key, 0x01, sizeof(key));
  memset(in,  0x02, sizeof(in));

  /* 2) Инициализация контекста и установка ключа */
  /* kuznyechik_set_key(&ctx, key); */

  /* 3) Шифрование одного блока */
  /* kuznyechik_encrypt(&ctx, 16, out, in); */
```

```
 /* 4) out содержит шифртекст блока */
 (void)out;
 return 0;
}
```

Для промышленного применения следует реализовать режим (например, CTR) с корректной обработкой синхропосылки/счётчика, длины сообщений, а также обеспечить требования к генерации случайных значений и управлению ключами.

6.2. Демонстрация: шифрование одного блока «Магмой»

```
#include <stdint.h>
#include <string.h>

int main(void) {
 /* Магма: ключ 32 байта, блок 8 байт */
 uint8_t key[32] = {0};
 uint8_t in[8]   = {0};
 uint8_t out[8]  = {0};

 /* struct magma_ctx ctx; */

 memset(key, 0x11, sizeof(key));
 memset(in,  0x22, sizeof(in));

 /* magma_set_key(&ctx, key); */
 /* magma_encrypt(&ctx, 8, out, in); */

 (void)out;
 return 0;
}
```

Требование безопасности: блочный шифр не должен применяться в режиме «ECB» для произвольных данных. Для прикладного шифрования применяются режимы (CTR/CBC/иные) согласно криптополитике. Для протокольных применений используется реализация на стороне TLS-библиотеки (GnuTLS), обеспечивающая согласование параметров.

7. Практический эффект для пользователей и

эксплуатации

7.1. Для системных администраторов

- возможность развёртывания сервисов с GnuTLS в ГОСТ-профиле без внешних криптомодулей;
- воспроизводимая проверка параметров TLS-соединений через `gnutls-cli`/`gnutls-serv`;
- снижение числа нестандартных сборок и локальных патчей в инфраструктуре;
- упрощение регрессионных проверок при обновлениях библиотек.

7.2. Для разработчиков

- доступность «Кузнечика»/«Магмы» как примитивов на уровне `libnettle` для встроенной криптографии;
- ускорение прототипирования и тестирования интеграций, связанных с ГОСТ-профилями TLS;
- снижение затрат на ранней стадии: тестовые контуры не требуют приобретения криптопровайдера для базовой совместимости;
- возможность формировать автоматические тесты (CI) на предмет корректности рукопожатия и выбора ciphersuite.

Граница применения: наличие криптопримитивов «из коробки» предназначено для инженерной эксплуатации и тестовых/пилотных контуров. В контуре, где требуется применение сертифицированного СКЗИ (в соответствии с требованиями регулятора и локальных регламентов), используется сертифицированный криптопровайдер и утверждённая схема управления ключами.

8. Эксплуатационные сведения: «из коробки» в НАЙС.ОС

Поддержка «Кузнечика» и «Магмы» предоставляется на уровне системных библиотек. Для эксплуатации достаточно наличия пакетов, поставляемых в базовом репозитории НАЙС.ОС, и стандартного механизма обновления.

8.1. Контроль наличия библиотек и версий

```
# Проверка версий инструментов GnuTLS
```

```
gnutls-cli --version
certtool --version

# Перечень поддерживаемых возможностей (фактическое состояние)
gnutls-cli --list | sed -n '1,260p'
```

8.2. Контроль доступности алгоритмов

```
# Поиск "kuznyechik/magma" в списках (именование зависит от версии)
gnutls-cli --list | grep -Ei 'kuznyechik|magma' || true

# Валидация через рукопожатие (результат должен показывать ГОСТ-набор шифров)
gnutls-cli --priority "NONE:+VERS-TLS1.2:+GOST" example.invalid 2>&1 | sed -n '1,120p'
```

Примечание: команда к удалённому узлу приведена как пример. Для воспроизводимого контроля требуется локальный тестовый сервер (см. раздел 5.1), поскольку доступность ГОСТ-наборов шифров на стороне внешнего сервера не гарантируется.

9. Заключение

В **НАЙС.ОС** обеспечена штатная доступность симметричных блочных шифров «Кузнечик» и «Магма» (ГОСТ Р 34.12-2015) на уровне криптобиблиотеки `libnettle` с последующей применимостью в `libgnutls`. Интеграция предоставляет возможность использования современных ГОСТ-примитивов в TLS- и прикладных сценариях без внешних модулей и без дополнительных действий после установки системы.

Практический эффект: ускорение развёртывания тестовых контуров, снижение затрат на интеграцию и повышение воспроизводимости диагностики. Для контуров, требующих применения сертифицированных СКЗИ, предусмотрено использование сертифицированных криптопровайдеров согласно нормативным и организационным требованиям.

Формальный критерий готовности: успешное TLS-рукопожатие в заданном ГОСТ-профиле и подтверждение доступности «Кузнечика»/«Магмы» по фактическому перечню возможностей GnuTLS в целевом окружении НАЙС.ОС.