libksba в НАЙС.ОС

libksba — это ключевая библиотека в экосистеме GnuPG, отвечающая за парсинг и обработку сертификатов X.509, CMS-сообщений, OCSP и CRL. Мы добавили в неё полноценную поддержку российских криптографических алгоритмов ГОСТ Р 34.10-2012 и ГОСТ Р 34.11-2012, включая корректный разбор и проверку подписей, генерацию запросов на сертификат (CSR), валидацию цепочек и расширений, поддержку OID семейства ТК26, а также десятки тестов с реальными сертификатами и CRL. Для конечного пользователя это означает, что теперь можно открывать, подписывать и проверять документы, подписанные российской электронной подписью (ЭП), «из коробки» — без сторонних патчей и плясок с OpenSSL. А разработчики получают стабильный API для интеграции ГОСТ в свои C/C++-приложения, совместимый с существующим стеком GnuPG и libgcrypt.

ГОСТ-криптография в 2025: поддержка ГОСТ в libksba и практическое применение

Документ устанавливает структуру и технические пояснения по поддержке ГОСТ-алгоритмов в экосистеме GnuPG через библиотеку libksba. Термины «должен», «следует», «допускается», «не допускается» применяются в нормативном смысле.
Область применения. Материал предназначен для инженеров и администраторов, которые обеспечивают работу электронной подписи (ЭП), проверку сертификатов X.509, CMS/PKCS#7, CRL и OCSP в средах, где ГОСТ-алгоритмы являются обязательными. Примеры команд ориентированы на НАЙС.ОС (RPM/dnf). В иных дистрибутивах имена пакетов и пути могут отличаться.
Содержание

---

1. Зачем вообще ГОСТ в 2025-м

В 2025 году ГОСТ-алгоритмы остаются ключевым стандартом для российской криптографии, в том числе в контексте электронной подписи (ЭП), применяемой в юридически значимом документообороте. Поддержка ГОСТ-алгоритмов в программном обеспечении требуется, когда организация должна обеспечивать выпуск/использование сертификатов и проверку подписей, совместимых с российскими профилями и удостоверяющими центрами.

Как правило, необходимость ГОСТ-совместимости привязана к нормативным и профильным требованиям. В практическом описании (в терминах этого документа) учитываются следующие источники и профили:

- **Федеральный закон № 63-ФЗ** «Об электронной подписи» — правовая основа применения ЭП и общие требования к её использованию.
- **Приказ ФСБ № 795** — перечень/регламентация допустимых криптографических алгоритмов, включая ГОСТ Р 34.10-2012 (подпись) и ГОСТ Р 34.11-2012 (хеширование).
- **Профиль ТК 26** — ограничения и правила для сертификатов и подписей: допустимые OID, расширения `KeyUsage`, `ExtendedKeyUsage` (EKU), политики сертификатов и связанные проверки совместимости.

Поддержка ГОСТ-алгоритмов в ПО необходима, когда выполняется хотя бы один из следующих сценариев:

- Подписание и проверка документов (PDF, XML, ZIP) в электронном документообороте (ЭДО).
- Интеграция с государственными системами (ЕПГУ, СМЭВ, ФНС, Росреестр и др.), где ГОСТ-подписи обязательны.

- Проверка подписей/сертификатов от российских УЦ (ФНС, Минцифры, корпоративные УЦ) без «переупаковки» подписи в альтернативные алгоритмы.
- Сопряжение с сертифицированными СКЗИ и аппаратными носителями (например, КриптоПро CSP, Рутокен, JaCarta) на уровне инфраструктуры.

Определение (кратко). ГОСТ-подпись — электронная подпись, сформированная с использованием российских стандартов: **ГОСТ Р 34.10-2012** (эллиптические кривые, формирование подписи) и **ГОСТ Р 34.11-2012** (Streebog, вычисление хеш-дайджеста). Такие подписи применяются в юридически значимых документах и являются обязательными в ряде регламентированных процессов.
Для специалистов. Поддержка ГОСТ в **libksba** увязана с международными описаниями форматов и применений ГОСТ в контейнерах и транспорте ключей, включая RFC 7836 и RFC 9215. В терминах реализации это означает корректную обработку ASN.1/DER, идентификаторов алгоритмов (OID), особенностей представления подписи и применение профильных ограничений (в т.ч. ТК-26), где это требуется политикой валидации. Ссылки: RFC 7836, RFC 9215.
Предупреждение. Нормативная и профильная база может уточняться и дополняться. Перед внедрением в продуктивной среде следует сверить требования с актуальными регламентами организации и применимыми отраслевыми документами.

---

2. Что такое libksba и где он живёт

libksba — библиотека с открытым исходным кодом для работы с криптографическими структурами в формате ASN.1/DER. Название связано с задачей обработки синтаксиса X.509/CMS. В контексте инфраструктуры ЭП это означает разбор и генерацию объектов, с которыми работают PKI-системы.

libksba должна обеспечивать обработку следующих типов объектов:

- Сертификаты X.509.
- Запросы на сертификат (CSR, PKCS#10).
- Сообщения CMS (PKCS#7), включая подписи и зашифрованные данные.
- Списки отзыва сертификатов (CRL).
- Ответы OCSP (проверка статуса сертификата).

Библиотека входит в экосистему **GnuPG** и используется в компонентах и приложениях, включая:

- **Kleopatra** — графический интерфейс управления ключами и подписями.
- **GpgOL** — интеграция для S/MIME и OpenPGP.
- **GpgSM** — утилита для X.509 и CMS-подписей.
- PKI-решения и системы ЭДО, которые используют libksba как слой разбора/проверки X.509/CMS.

До внедрения поддержки ГОСТ (описание в рамках этой статьи) библиотека соответствовала международным стандартам (RFC 5280 для X.509, RFC 5652 для CMS, RFC 6960 для OCSP), но не могла обрабатывать российские ГОСТ-алгоритмы. В результате попытка разобрать ГОСТ-объект приводила к ошибке, например:

```
unsupported algorithm 1.2.643.7.1.1.3.2
```

Примечание. Добавление ГОСТ-поддержки в libksba расширяет совместимость экосистемы GnuPG с российской криптографией и снижает зависимость от проприетарных компонентов в сценариях, где достаточно open-source стека и допускается выбранная модель доверия.

При этом следует учитывать, что криптографическая «математика» (подпись/проверка, хеширование) выполняется не libksba напрямую, а через криптобэкенд (например, `libgcrypt`). Следовательно, для полноценной работы ГОСТ-цепочки должны быть поддержаны как минимум: OID/ASN.1 на стороне libksba и алгоритмы на стороне криптобиблиотеки.

---

3. Что именно добавлено в поддержку ГОСТ

Поддержка ГОСТ в **libksba** представляет собой набор изменений, охватывающих распознавание алгоритмов, проверку подписей в различных контейнерах, обработку PKCS#10 и валидацию профилей (включая ТК-26). Ниже приводится структурированное описание основных подсистем и изменений.

Подсистема	Ключевые изменения	Файлы / функции
Алгоритмический слой	Распознавание и маппинг OID для ГОСТ Р 34.10-2012 (256/512) и ГОСТ Р 34.11-2012 (Streebog). Обработка алгоритмов в рамках форматов и описаний, применяемых в практике (в т.ч. RFC-профили).	`is_gost_oid`, `is_gost_algo`, `oid.c`
Подпись CMS / OCSP / CRL	Проверка подписей для CMS/OCSP/CRL с учётом особенностей представления ГОСТ-подписей. Включая корректировки подписи (например, инверсия R/S) для совместимости с реализацией в криптобэкенде.	`cms.c`, `ocsp.c`, `crl.c`, `gost_adjust_signature`
PKCS#10 (CSR)	Поддержка генерации, парсинга и валидации CSR с ГОСТ-алгоритмами.	`pkcs10-gost.c`, `_ksba_pkcs10_check_gost`
Валидация политик	Проверки соответствия профилю ТК-26: допустимые OID/политики, правила `KeyUsage` и `ExtendedKeyUsage`.	`check_policy_tk26_only`, `parse_eku_for_gost`, `parse_key_usage_for_gost`
API / ABI	Добавлены функции для ГОСТ-проверок и расширены структуры для отражения ГОСТ-вариантов использования. Изменения выполняются так, чтобы минимизировать риск нарушения совместимости.	`ksba.h`, `_ksba_check_cert_sig`, `_ksba_crl_check_signature_gost`
Тесты	Набор тестовых данных (сертификаты/ключи/CRL/OCSP), генерация и интеграция в тестовый контур.	`tests/samples/gost_certs2/*`, `generate_gost_certs3.sh`
Утилиты	Утилита проверки CMS-подписей, демонстрирующая практическое использование API для ГОСТ.	`utils/check_cms_signed.c`

3.1. Пример: проверка сертификата с ГОСТ-подписью

Ниже приведён упрощённый фрагмент кода, иллюстрирующий типовую структуру проверки подписи сертификата, включая: определение ГОСТ по OID, профильные проверки (KeyUsage/ТК-26), вычисление хеша и проверку подписи, а также запасной вариант с корректировкой подписи для совместимости.

Примечание. Фрагмент приведён как иллюстрация архитектуры и порядка действий. Конкретные детали зависят от версий libksba/libgcrypt, формата входных объектов и активированной политики валидации.

```
/* Проверка подписи сертификата с использованием issuer_cert */
gpg_error_t
_ksba_check_cert_sig (ksba_cert_t issuer_cert, ksba_cert_t cert)
{
 gpg_error_t err;
 const char *algoid;
 gcry_md_hd_t md;
 int algo, gost_key;

 /* Получение алгоритма подписи */
 algoid = ksba_cert_get_digest_algo (cert);
 algo = gcry_md_map_name (algoid);
 if (!algo)
   return gpg_error (GPG_ERR_DIGEST_ALGO);

 /* Определение, является ли алгоритм ГОСТ */
 gost_key = algoid && !memcmp (algoid, "1.2.643", 7);

 if (gost_key) {
   /* Проверка KeyUsage для ГОСТ-сертификата */
   err = check_key_usage_for_gost (cert, KSBA_KEYUSAGE_DIGITAL_SIGNATURE);
   if (err)
     return err;

   /* Проверка политики ТК-26 для issuer_cert */
   err = check_policy_tk26 (issuer_cert);
   if (err)
     return err;
 }

 /* Инициализация хеша */
 err = gcry_md_open (&md, algo, 0);
 if (err)
   return err;

 /* Вычисление хеша сертификата */
 err = ksba_cert_hash (cert, 1, HASH_FNC, md);
 if (err) {
```

```
    gcry_md_close (md);
    return err;
  }
  gcry_md_final (md);

  /* Проверка подписи с помощью libgcrypt */
  gcry_sexp_t s_sig, s_hash, s_pkey;
  /* ... (получение sig_val, public_key и построение S-выражений) ... */
  err = gcry_pk_verify (s_sig, s_hash, s_pkey);

  if (err && gost_key) {
    /* Попытка инверсии R/S для ГОСТ */
    gcry_sexp_t tmp = s_sig;
    gpg_error_t e2 = gost_adjust_signature (&tmp);
    if (!e2) {
      s_sig = tmp;
      err = gcry_pk_verify (s_sig, s_hash, s_pkey);
    }
  }

  /* Очистка ресурсов */
  gcry_md_close (md);
  gcry_sexp_release (s_sig);
  gcry_sexp_release (s_hash);
  gcry_sexp_release (s_pkey);
  return err;
}
```

Функциональный смысл фрагмента:

- Определение ГОСТ-алгоритма по префиксу OID `1.2.643`.
- Проверка профильных ограничений (`KeyUsage`, политика ТК-26), когда это требуется.
- Вычисление дайджеста и проверка подписи через криптобэкенд.
- Дополнительная ветка совместимости: корректировка подписи (R/S) при ошибке проверки.

Итог: после внесённых изменений libksba становится пригодной для полного цикла обработки ГОСТ-объектов (CSR сертификаты/цепочки CMS CRL/OCSP) при условии наличия ГОСТ-поддержки в криптобиблиотеке.

---

4. Влияние на конечного пользователя

Поддержка ГОСТ в **libksba** влияет на пользователей разных ролей по-разному. Ниже

описаны эффекты, которые должны наблюдаться при корректной интеграции libksba и криптобэкенда.

4.1. Обычный пользователь

- Прозрачная работа с ГОСТ-подписями: документы (PDF/XML/ZIP), подписанные ГОСТ-алгоритмами, должны открываться и проверяться в приложениях, использующих libksba (например, Kleopatra), без ошибок разбора алгоритмов.
- Типовая ошибка до внедрения ГОСТ-поддержки выглядела так:

```
cannot parse digest algorithm 1.2.643.7.1.1.2.2
```

- Упрощение взаимодействия с гос-системами и контрагентами, где ГОСТ обязателен.
- Возможность проверки подписей российских УЦ без дополнительных пользовательских «костылей», при условии корректной настройки доверия и цепочек.

4.2. Системный администратор

- Упрощение развертывания: ГОСТ-поддержка включена в библиотеку и не требует ручного сопровождения «локальных патч-сборок» в каждом контуре.
- Снижение рисков при обновлениях: при корректной упаковке под RPM (НАЙС.ОС) обновления становятся воспроизводимыми и контролируемыми.
- Повышение совместимости с инфраструктурой российских УЦ при условии, что криптобэкенд (например, libgcrypt) также поддерживает ГОСТ.
- Упрощение интеграций с аппаратными носителями и внешними криптопровайдерами на уровне инфраструктуры, когда это предусмотрено политикой.

4.3. Разработчик / интегратор

- Доступ к единым точкам интеграции: функции для ГОСТ-проверок и утилиты-примеры позволяют стандартизировать обработку ГОСТ-контейнеров.
- Пример (иллюстративно): проверка CMS-подписи в утилите `check_cms_signed` демонстрирует разбор CMS, выбор алгоритма и особенности ГОСТ-обработки:

```
/* Часть утилиты check_cms_signed.c (иллюстративный фрагмент) */
```

```
static gcry_error_t process_file(const char *sig_file, const char *content_file, const char *cert_file) {
  gcry_error_t err;
  ksba_cms_t cms;
  gcry_md_hd_t data_md;
  gcry_sexp_t s_sig, s_hash, s_pkey;

  /* Инициализация CMS и хеша */
  err = ksba_cms_new(&cms);
  err = gcry_md_open(&data_md, 0, 0);  /* Парсинг CMS */
  ksba_cms_set_reader_writer(cms, reader, writer);
  while (stopreason != KSBA_SR_READY) {
    err = ksba_cms_parse(cms, &stopreason);
  }

  /* Проверка алгоритма */
  const char *algoid = ksba_cms_get_digest_algo(cms, 0);
  int is_gost = !strncmp(algoid, "1.2.643", 7);

  gcry_md_enable(data_md, gcry_md_map_name(algoid));

  /* Проверка подписи */
  if (is_gost) {
    /* Здесь могут выполняться специфичные преобразования для ГОСТ (например, порядок
байтов) */
    err = gcry_sexp_build(&s_hash, NULL, "(data(flags gost)(value %b))", digest_len, digest);
  }

  err = gcry_pk_verify(s_sig, s_hash, s_pkey);
  return err;
}
```

Примечание. Реальная интеграция должна включать: корректную загрузку доверенных цепочек, политику проверки (ТК-26 при необходимости), работу с хранилищем сертификатов и обработку ошибок валидации как части бизнес-логики (ЭДО/шлюзы/PKI).

Итог: ГОСТ-поддержка в libksba делает библиотеку универсальным «синтаксическим» слоем для российской криптографии, что упрощает эксплуатацию и разработку решений, совместимых с инфраструктурой ЭП в России.

---

5. Под капотом: детали реализации

Раздел предназначен для специалистов. Здесь фиксируются технические особенности, которые следует учитывать при анализе, отладке и сопровождении.

5.1. Инверсия координат R/S в подписи

ГОСТ-подписи (ГОСТ Р 34.10-2012) в X.509/CMS часто представляются как пара `R||S`. В зависимости от формата и реализации в криптобэкенде может возникать необходимость преобразования порядка байтов/компонент подписи для корректной проверки. Для этого применяется функция `gost_adjust_signature`.

```
/* Инверсия байтов для R и S в ГОСТ-подписи */
static gpg_error_t
gost_adjust_signature (gcry_sexp_t *sig)
{
  gcry_sexp_t r = NULL, s = NULL;
  const unsigned char *rbuf, *sbuf;
  size_t rlen, slen;
  unsigned char *rrev = NULL, *srev = NULL;
  gpg_error_t err = 0;

  r = gcry_sexp_find_token (*sig, "r", 0);
  s = gcry_sexp_find_token (*sig, "s", 0);
  if (!r || !s)
    return gpg_error (GPG_ERR_INV_SEXP);

  rbuf = gcry_sexp_nth_buffer (r, 1, &rlen);
  sbuf = gcry_sexp_nth_buffer (s, 1, &slen);
  if (!rbuf || !sbuf || !rlen || rlen != slen)
    return gpg_error (GPG_ERR_INV_SEXP);

  rrev = gcry_xmalloc (rlen);
  srev = gcry_xmalloc (slen);
  invert_bytes (rrev, rbuf, rlen);
  invert_bytes (srev, sbuf, slen);

  gcry_sexp_release (*sig);
  err = gcry_sexp_build (sig, NULL,
                         "(sig-val (gost (r %b)(s %b)))",
                         (int)rlen, rrev, (int)slen, srev);

  gcry_sexp_release (r);
  gcry_sexp_release (s);
  gcry_free (rrev);
  gcry_free (srev);
  return err;
}
```

Назначение функции:

- извлечь компоненты `r` и `s` из S-выражения;
- преобразовать представление подписи (при необходимости);

- собрать новое S-выражение, совместимое с проверкой подписи в криптобэкенде.

5.2. Проверка требований профиля ТК-26

Профиль ТК-26 накладывает строгие ограничения на сертификаты и подписи. В рамках описанной логики проверки учитываются, в частности:

- `KeyUsage` для конечных сертификатов должен включать `digitalSignature`.
- Для CA-сертификатов и подписывающих сущностей должны присутствовать соответствующие назначения (`keyCertSign`, `cRLSign`) по роли.
- `CertificatePolicies` должны включать допустимые OID из семейства `1.2.643.*` (с учётом исключений вроде anyPolicy по правилам проверки).
- `ExtendedKeyUsage` (EKU) должен соответствовать разрешённым вариантам применения.

```
/* Проверка политик ТК-26 */
static gpg_error_t
check_policy_tk26_only (ksba_cert_t cert)
{
 gpg_error_t err;
 char *pols = NULL;
 int any = 0;

 err = ksba_cert_get_cert_policies (cert, &pols);
 if (gpg_err_code (err) == GPG_ERR_NO_DATA)
  return gpg_error (GPG_ERR_NO_POLICY_MATCH);
 if (err)
  return err;

 char *line = pols;
 while (line && *line) {
  char *end = strchr (line, '\n');
  if (!end)
   end = line + strlen (line);

  if (end - line >= 7 && !memcmp (line, "1.2.643", 7))
   any = 1;
  else if (end - line == 13 && !memcmp (line, "2.5.29.32.0", 11))
   { /* Ignore anyPolicy */ }
  else {
   xfree (pols);
   return gpg_error (GPG_ERR_NO_POLICY_MATCH);
  }

  if (*end)
   line = end + 1;
```

```
    else
      break;
  }

  xfree (pols);
  return any ? 0 : gpg_error (GPG_ERR_NO_POLICY_MATCH);
}
```

Примечание. Проверки профиля должны соответствовать политике доверия организации: какие УЦ считаются доверенными, какие цепочки допустимы, какие исключения разрешены, и как трактуются неоднозначные расширения (например, anyPolicy).

5.3. Безопасность и совместимость

При расширении библиотеки криптографическими профилями должны соблюдаться следующие принципы:

- ГОСТ-путь активируется только при обнаружении соответствующих OID и наличии поддержки алгоритмов в криптобэкенде.
- Изменения API/ABI должны выполняться так, чтобы не ломать существующие потребители (добавление полей — в конец структур, новые функции — без удаления старых).
- Ошибки проверки должны быть различимы: «неизвестный алгоритм», «несовпадение подписи», «не проходит политика ТК-26», «нет данных» и т.п.

5.4. Производительность (оценка)

Примечание. Значения ниже следует рассматривать как иллюстративные. Фактическая производительность зависит от CPU, реализации алгоритмов, параметров ключей и профиля нагрузки (дисковый ввод/вывод, парсинг цепочек, кэширование и т.п.).

Алгоритм	Среднее время проверки (ms)	Примечание
RSA-2048 / SHA-256	~0.42	Типовой профиль для S/MIME и ряда PDF-сценариев
GOST-512 / GOST12-512	~0.51	Профиль ТК-26, в среднем умеренно тяжелее

5.5. Тестирование

Полноценная реализация должна сопровождаться тестовым контуром, который покрывает:

- проверку цепочек сертификатов (включая профильные ограничения);
- валидацию CRL и OCSP-ответов;
- разные комбинации политик и EKU;
- негативные сценарии (несовпадение подписи, неподходящая политика, некорректные расширения).

```
/* Тест цепочки сертификатов ТК-26 (иллюстративный фрагмент) */
fname = prepend_srcdir ("samples/gost_certs2/root_gost_tk26.crt");
chain3[0] = read_cert (fname); xfree (fname);
fname = prepend_srcdir ("samples/gost_certs2/test_gost_policy.crt");
chain3[1] = read_cert (fname); xfree (fname);
fname = prepend_srcdir ("samples/gost_certs2/leaf_gost_tk26.crt");
chain3[2] = read_cert (fname); xfree (fname);
err = _ksba_check_cert_chain_tk26 (chain3, 3, 0);
if (err) {
  fprintf (stderr, "test11: expected %d got %s (%d)\n", 0,
           gpg_strerror (err), gpg_err_code (err));
  for (int i=0; i < 3; i++)
    ksba_cert_release (chain3[i]);
  return 1;
}
```

---

6. Дорожная карта

Поддержка ГОСТ в libksba может расширяться в сторону унификации обработки атрибутов, аппаратных токенов и дополнительных режимов шифрования. Ниже приведены направления, которые часто встречаются в практических планах развития подобных интеграций.

6.1. Унификация CMS-атрибутов

Следует стремиться к унификации обработки общих CMS-атрибутов (`messageDigest`, `signingTime`, `ESSCertID`), чтобы упрощать расширение на новые алгоритмы и снижать сложность отладки.

6.2. Интеграция с PKCS#11

Поддержка аппаратных токенов (Рутокен, JaCarta и др.) через PKCS#11 позволяет выполнять операции подписи и генерации ключей на токене. Для этого требуется корректная связка: приложение libksba криптобэкенд PKCS#11-модуль.

6.3. Поддержка ГОСТ Р 34.12-2015 (Кузнечик, CTR-ACPKM) в CMS

Добавление шифрования CMS-объектов (`EncryptedData`) с ГОСТ Р 34.12-2015 может потребовать доработки криптобэкенда или подключения альтернативного механизма шифрования, в зависимости от выбранной архитектуры. Ссылка на документ: ГОСТ Р 34.12-2015.

6.4. Планы по выпуску

- **libksba-gost** как отдельная ветка/релиз с консолидированным набором ГОСТ-функций и тестов.
- CLI-утилиты: проверка подписей, генерация CMS-контейнеров, инструменты работы с OCSP/CRL.
- CI/CD-профиль тестирования на нескольких платформах с воспроизводимыми наборами тест-данных.

Предупреждение. «Дорожная карта» фиксирует намерения и направления и не является гарантией сроков. Планирование внедрения в продуктив следует выполнять на основании тестируемых релизов и внутреннего календаря изменений.

---

7. Как начать использовать

7.1. Базовая установка и сборка (НАЙС.ОС)

Для сборки libksba из исходников следует установить инструменты сборки и зависимости. На НАЙС.ОС это выполняется через `dnf`. Пакеты и их точные имена зависят от репозиторного состава, но общий класс зависимостей стабилен.

```
# Инструменты сборки и типовые зависимости (пример для RPM/НАЙС.ОС)
```

```
sudo dnf makecache

# Если доступна группа инструментов разработки:
sudo dnf groupinstall -y "Development Tools" || true

# Базовые зависимости для сборки autotools-проекта и связок GnuPG
sudo dnf install -y \
  autoconf automake libtool gettext bison texinfo \
  gcc gcc-c++ make pkgconf-pkg-config \
  libgpg-error-devel libgcrypt-devel
```

Предупреждение. Для реальной ГОСТ-функциональности `libgcrypt` должна быть собрана/установлена с поддержкой ГОСТ-алгоритмов. Если криптобэкенд не поддерживает ГОСТ, libksba сможет распознать OID и структуры, но проверка подписи может завершаться ошибкой алгоритма/проверки.

7.2. Сборка libksba (пример)

```
# Пример последовательности для autotools
./autogen.sh --force
./configure --enable-maintainer-mode
make -j"$(nproc)"
```

7.3. Проверка CMS-подписи через утилиту (пример)

```
# Пример использования утилиты (имена файлов условные)
./check_cms_signed sig.p7s document.pdf cert.der
```

7.4. Интеграция в проект (минимальный пример)

```
#include <ksba.h>
#include <gpg-error.h>
#include <stdio.h>

int main() {
  ksba_cert_t issuer_cert, cert;
  gpg_error_t err;

  ksba_cert_new(&issuer_cert);
  ksba_cert_new(&cert);

  /* ... загрузка issuer_cert и cert из DER ... */
```

```
  err = _ksba_check_cert_sig(issuer_cert, cert);
  if (err) {
    fprintf(stderr, "Ошибка проверки: %s\n", gpg_strerror(err));
    return 1;
  }

  printf("Подпись сертификата верна!\n");

  ksba_cert_release(issuer_cert);
  ksba_cert_release(cert);
  return 0;
}
```

Примечание. Для реального применения требуется дополнить пример: загрузкой доверенных цепочек, политиками проверки (в т.ч. ТК-26 при необходимости), обработкой CRL/OCSP, а также корректным управлением источниками доверия (store/anchors).

---

8. Заключение

Поддержка ГОСТ в **libksba** расширяет open-source стек для работы с российской криптографией в сценариях ЭП и PKI. При корректной интеграции с криптобэкендом библиотека должна обеспечивать:

- обработку сертификатов X.509, CMS, CRL и OCSP с ГОСТ-алгоритмами;
- проверки профильных требований (включая ТК-26) там, где это требуется политикой;
- возможность интеграции в приложения и инфраструктурные сервисы без ручных «обходных» механизмов.

Для пользователей это снижает количество ошибок совместимости при проверке подписей. Для администраторов — упрощает сопровождение и обновления (особенно при корректной RPM-упаковке в НАЙС.ОС). Для разработчиков — даёт единый программный слой работы с ASN.1/DER и контейнерами, совместимый с российскими OID и профилями.