**Защита Сетей и Коммуникаций**

Защищённые прокси — практичная альтернатива VPN



В интернете есть достаточное количество информации по теме шифрования и защиты трафика от вмешательств, однако сложился некоторый перекос в сторону различных VPN-технологий. Возможно, отчасти он вызван статьями VPN-сервисов, которые так или иначе утверждают о строгом превосходстве VPN-решений перед прокси. При этом многие решения тех же VPN-провайдеров, не смотря на маркетинговое позиционирование в качестве VPN, технически являются прокси.

На практике прокси больше подходят для повседневной защиты веб-трафика, не создавая при этом неудобств в виде заметной потери скорости и неизбирательности туннелирования. То есть при использовании хорошего прокси не стоит необходимость его отключать для комфортного пользования интернетом.

В этой статье расказано о преимуществах защищённого прокси перед VPN и предложены различные реализации, готовые к использованию.

В чём различие между VPN и прокси?

VPN — это общее название технологий для объединения внутренних сетей на уровне сетевых пакетов или кадров через соединение, установленное поверх другой сети (чаще всего публичной).

Прокси — это серверное приложение, осуществляющее соединения или запросы от своего имени и сетевого адреса в пользу подключившегося к нему клиента, пересылая в результате ему все полученные данные.

VPN осуществляет пересылку полезной нагрузки, находящейся на третьем или втором уровне сетевой модели OSI. Прокси осуществляют пересылку полезной нагрузки между четвёртым и седьмым уровнями сетевой модели OSI включительно.

И VPN, и прокси могут иметь или не иметь шифрования между клиентом и сервером. Обе технологии пригодны для того, чтобы направить трафик пользователя через доверенный сервер, применяя шифрование по пути до него. Однако, подключение через прокси делает это более прямолинейным способом, не привнося дополнительную инкапсуляцию сетевых пакетов, серые адреса самой VPN сети и изменения таблицы маршрутизации, которые привносит VPN просто лишь для того, чтобы сетевой стек системы пользователя направил трафик через нужный сервер.

Преимущества прокси

1. В условиях реальной сети с потерями практическая скорость прокси зачастую выше, чем у VPN-решений. Это вызвано тем, что при проксировании TCP-соединений ретрансмиты на участках клиент-прокси и прокси-целевой узел происходят независимо. Прокси имеет свои TCP-буферы и кратковременные задержки ввода-вывода в обе стороны не сказываются на передаче с противоположной стороны. VPN же работает только на сетевом уровне (IP) и потерянные сегменты TCP будут пересылаться по всей длине пути от VPN-клиента до целевого сервера.
2. Гибкость. Проще настроить избирательное проксирование. Использование прокси можно ограничить конкретными приложениями, в браузере — конкретными доменами. Можно использовать несколько разных прокси для разных адресов назначения одновременно.
3. Трудно обнаружить с помощью DPI, в том числе DPI осуществляющими активные пробы. Однако, для этого необходима некоторая донастройка. В случае с прокси через TLS, такое соединение можно выдать за обычное HTTPS-соединение. В случае с VPN факт его использования виден даже пассивному DPI. Даже если это Wireguard.
4. Нет целого класса проблем с внезапно прервавшимся VPN-соединением. В худшем сценарии VPN-соединение может прерваться и пользователь не заметит, что его трафик уже не защищён и/или он уже работает со своего «домашнего» IP-адреса. В случае с прокси такие проблемы исключены.
5. Нет принципиальной возможности проводить атаки, пускающие трафик мимо VPN-туннеля. [Пример такой проблемы](https://www.opennet.ru/opennews/art.shtml?num=51986).
6. Не нужны высокие привилегии в системе ни для клиента прокси, ни для сервера. Это может быть весьма полезно в случаях, когда у Вас нет высоких прав в системе.

Требования к прокси

Выбирая для себя реализацию защищённого прокси-сервера, я отметил несколько критериев, которым она должна удовлетворять:

1. Обязательное шифрование и защита целостности данных внутри соединения.
2. Надёжная и проверенная криптография. Доморощенные криптопротоколы крайне нежелательны.
3. Устойчивость к DPI, в том числе к [активным пробам](https://en.wikipedia.org/wiki/Great_Firewall#Active_probing). В идеале протокол должен внешне выглядеть неотличимо от протоколов, к которым обычно не возникает претензий. Например, как HTTPS.
4. Отсутствие мультиплексирования нескольких TCP-соединений внутри одного. Причина этого требования такова: нежелательно, чтобы скорость нескольких соединений была ограничена скоростью одного TCP-соединения. Кроме того, при мультиплексировании нескольких соединений внутри одного приостановка получения данных из одного внутреннего (подвергнутого мультиплексированию) соединения может застопорить все остальные на неопределённое время. Для этого достаточно, чтобы со стороны прокси-сервера ожидало отправки больше данных, чем суммарный размер буфер отправки клиента-демультиплексора и буфер приёма сокета у застопорившего приём приложения. В частности такого эффекта можно иногда добиться, начав качать большой файл через SSH SOCKS5 прокси (ssh -ND 1080), и поставив скачивание на паузу. При неудачном стечении обстоятельств никакой трафик через туннель больше не будет принят. [Более подробно о проблеме head-of-line blocking](https://softwareengineering.stackexchange.com/questions/325881/is-it-a-good-idea-to-multiplex-blocking-streams-into-a-tcp-connection).
5. Отсутствие привязки к поставщику или сервису.

Я рассмотрел несколько известных протоколов для шифрования соединения с прокси:

|  |  |
| --- | --- |
| **Наименование** | **Резюме** |
| OpenSSH dynamic port forwarding (ssh -ND 1080) | Использует мультиплексирование: низкая скорость, задержки |
| shadowsocks | [Провал DPI](https://github.com/net4people/bbs/issues/22) |
| obfs4 | По внешнему виду он удовлетворяет всем требованиям, но есть моменты, вызывающие сомнения |

Особенности obfs4

В [спецификации протокола obfs4](https://github.com/Yawning/obfs4/blob/master/doc/obfs4-spec.txt) есть места, которые вызывают вопросы. В рукопожатии со стороны клиента используется номер часа от начала эпохи UNIX, который потом участвует в HMAC-подписи. Сервер, принимая такой пакет от клиента проверяет его, подставляя номер часа по своему времени. Если всё верно, то отвечает своей частью рукопожатия. Для борьбы с разбросом часов сервер должен ещё проверять предыдущий и следующий час.

Зная такое характерное поведение, можно проверить сервер на границе следующего и послеследующего часа, воспроизведя одно из прошлых записанных рукопожатий со стороны клиента. Если сервер перестанет отвечать своей частью рукопожатия в это самое время, то это достаточное основание, чтобы судить, что сервер обслуживает протокол obfs4.

Судя по всему, автор со временем осознал эту проблему и в коде obfs4 [реализована защита](https://github.com/Yawning/obfs4/blob/a8288437e30d50a3fcbf409103f2f72dcf4d03bf/common/replayfilter/replay_filter.go) от обнаружения через воспроизведение. В спецификации она нигде не описана.

Однако, такая защита наоборот упрощает работу по блокировке протокола: сетевому фильтру достаточно в случае сомнений задержать отправку рукопожатия от клиента, перехватив её, а затем отправить сообщение с рукопожатием первым. Таким образом он спровоцирует защиту от воспроизведения уже против клиента, вынуждая сам сервер блокировать клиента.

Следующий момент, вызывающий сомнения это формат «кадра» с данными. Выглядит он следующий образом:

 +------------+----------+--------+--------------+------------+------------+

 | 2 bytes | 16 bytes | 1 byte | 2 bytes | (optional) | (optional) |

 | Frame len. | Tag | Type | Payload len. | Payload | Padding |

 +------------+----------+--------+--------------+------------+------------+

 \\_ Obfs. \_/ \\_\_\_\_\_\_\_\_\_\_\_ NaCl secretbox (Poly1305/XSalsa20) \_\_\_\_\_\_\_\_\_\_\_/

Первые два байта каждого кадра это длина пакета, [гаммированная](https://ru.wikipedia.org/wiki/%D0%93%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) с ключом, который вычисляется от предыдущих ключей. Как он вычисляется ключ не столь важно, главное, что настоящая длина пакета подвергается операции побитового исключающего ИЛИ каким-то ключом. Это значит, что можно инвертировать бит в этой части данных и подмена не будет сразу замечена. Если инвертировать младший значащий бит этого поля, то длина кадра станет либо на единицу меньше истинной, либо на единицу больше. В первом случае это приведёт к сбросу соединения через небольшое случайное время из-за ошибки распаковки NaCl secretbox.

Второй случай более интересный: сервер будет ждать ещё один байт для того, чтобы начать распаковку криптобокса. Получив ещё ровно один байт он также сбросит соединение из-за ошибки распаковки криптобокса. Это поведение можно считать специфичным для obfs4 и можно судить, что мы с высокой вероятностью имеем дело с ним. Таким образом, удачно разрушив одно из соединений клиента, можно с примерно 50%-ным шансом обнаружить obfs4.

Конечно, определение границ одного кадра в потоке тоже может представлять непростую задачу, но нет чётких гарантий, что нерешаемую. В случае обмена короткими сообщениями границы одного кадра могут совпасть с границами TCP-сегмента.

Ну и последняя особенность заключается в том, что сам по себе протокол внешне не похож ни на один из общепринятых. Он спроектирован с предположением, что DPI играет по правилам и незнакомые протоколы просто не трогает. Суровая реальность может показать, что это не так.

По всем этим соображениям я воздержался от использования obfs4.

TLS и SSH в качестве криптографического транспорта

Разумно было бы воспользоваться стандартными защищёнными сетевыми протоколами вроде TLS или SSH для обёртывания соединений с прокси. Действительно, к таким протоколам обычно не возникает претензий со стороны DPI, потому что ими может быть зашифрован легетимный трафик. Что же касается активных проб со стороны DPI, этот вопрос можно решить в частном порядке, в зависимости от конкретного протокола прокси.

Ниже будут представлены несколько готовых решений на базе этих протоколов, пригодных для повседневной постоянной защиты трафика.

SOCKS5 внутри SSH

Вариант с использованием функции dynamic port forwarding у OpenSSH рассматривался выше, но он имеет большие проблемы со скоростью. Единственный способ избавиться от мультиплексирования соединений — это использовать альтернативную реализацию клиента SSH, который обеспечивал бы каждое проксируемое соединение отдельной SSH-сессией.

Я проделал такую работу и реализовал его: [Rapid SSH Proxy](https://github.com/Snawoot/rsp). Эта реализация обеспечивает каждое проксируемое соединение отдельной SSH-сессией, поддерживая пул подготовленных SSH-сессий для удовлетворения поступающих запросов подключения с минимальной задержкой.

Порядок работы с ним такой же, как у ssh -ND 1080: на стороне клиента запускается локальный SOCKS5-прокси, принимающий соединения и напрявляющий их в туннель через SSH-сервер.

Следует особо отметить ключевую особенность: никакого стороннего ПО не нужно устанавливать на сервер — rsp работает как ssh-клиент с обычным сервером OpenSSH. Сервером может быть любая unix-подобная операционная система, а так же Windows и Windows Server (в новых версиях OpenSSH теперь доступен в компонентах системы).

Приведу сравнение скорости через сервер в США:

|  |  |
| --- | --- |
| **OpenSSH** | **rsp** |
| Speedtest - OpenSSH | Speedtest - rsp |

SOCKS5 внутри TLS

В случае с TLS очевидным решением было бы использовать [stunnel](https://www.stunnel.org/) или аналогичную TLS-обёртку для TCP-соединений с SOCKS5-сервером. Это действительно вполне хорошо работает, но возникает следующая проблема: рукопожатие TLS для каждого нового соединения занимает дополнительное время и появляется заметная на глаз задержка при установлении новых соединений из браузера. Это несколько ухудшает комфорт при веб-серфинге.

Для того, чтобы скрыть эту задержку, я подготовил специализированную замену stunnel на клиенте, которая поддерживает пул уже установленных, готовых к запросу TLS-соединений. Даже целых два — первый из них послужил прототипом:

1. [Pooling TLS Wrapper](https://github.com/Snawoot/ptw)
2. [steady-tun](https://github.com/Snawoot/steady-tun)

В качестве сервера я предлагаю два варианта:

* Связка из реверс-прокси haproxy и SOCKS5-прокси dante, настроенная на защиту от активных проб со стороны клиентов, не прошедших аутентификацию по сертификату: [github.com/Snawoot/ptw/tree/master/docker\_deploy](https://github.com/Snawoot/ptw/tree/master/docker_deploy)
* Мой форк go-socks5-proxy со встроенной поддержкой TLS: [github.com/Snawoot/socks5-server](https://github.com/Snawoot/socks5-server)

HTTP-прокси внутри TLS aka HTTPS-прокси

Есть небольшая путаница в отношении сочетания слов «HTTPS» и «прокси». Есть два понимания такого словосочетания:

1. Обычный HTTP-прокси без шифрования, который поддерживает метод HTTP CONNECT и через который может успешно работать HTTPS.
2. HTTP-прокси, принимающий TLS-соединения.

Речь пойдёт о втором. Все платные и бесплатные браузерные расширения, предоставляющие «VPN», по сути настраивают браузер на использование такого прокси.

Примечательно, что ни в одном браузере нет простой возможности задать в настройках HTTPS-прокси через пользовательский интерфейс (то поле HTTPS-прокси, которое там есть, как раз относится к первому случаю). Но это не представляет собой большой трудности.

Поперебирав различные готовые варианты, я решил написать свой HTTP(S) прокси-сервер: [dumbproxy](https://github.com/Snawoot/dumbproxy).

Ключевой особенностью получившегося решения является то, что современные браузеры (Firefox и семейство Chrome, включая новый MS Edge) могут работать с ним без какого-либо дополнительного ПО на клиенте (см. [руководство по настройке клиентов](https://github.com/Snawoot/dumbproxy#using-http-over-tls-proxy)).

Следует отметить особенности реализации в отношении противодействия активным пробам со стороны DPI. HTTP-прокси легко распознать, подключившись к нему и осуществив попытку какого-либо запроса стороннего ресурса. Если прокси имеет авторизацию, то по стандарту он должен отвергнуть запрос с кодом 407, специфичным именно для HTTP-прокси, и предложить возможную схему для авторизации. Если прокси работает без авторизации, то он выполнит запрос, чем так же себя выдаст. Есть как минимум два способа решения этой проблемы (и оба они реализованы в dumbproxy).

Первый способ заключается в том, чтобы использовать аутентификацию клиентов по сертификатам ещё на этапе TLS-рукопожатия. Это самый стойкий метод, и это действительно корректная причина, по которой любой обычный веб-сервер мог бы отвергнуть клиента.

Второй способ заключается в том, чтобы скрыть от неавторизованных клиентов код ответа 407, возвращая вместо него любой другой ответ с ошибкой. Это вызывает другую проблему: браузеры не смогут понять, что для прокси требуется авторизация. Даже если браузер имеет сохранённый логин и пароль для этого прокси, ответ 407 важен для определения схемы авторизации, по которой эти учётные данные должны быть отправлены (Basic, Digest и т. д.). Для этого нужно позволить прокси генерировать ответ 407 на секретный запрос, чтобы браузер мог запомнить схему авторизации. В качестве такого секретного запроса используется настраиваемый секретный домен (не обязательно реально существующий). По умолчанию этот режим выключен. Подробности можно посмотреть в [разделе об аутентификации](https://github.com/Snawoot/dumbproxy#authentication).

Заключение

пользуюсь этими решениями уже один год и в итоге они мне полностью заменили VPN, вместе с этим сняв все проблемы, с которыми сопряжено его использование.

P.S. Если вам важна анонимность вашего адреса в интернете, не забудьте [выключить WebRTC в своём браузере](https://www.vpnmentor.com/blog/disable-webrtc-in-seconds/). Провериться на утечки IP-адреса можно [здесь](https://browserleaks.com/ip).

**Теги:**

* [vpn](https://habr.com/ru/search/?target_type=posts&order=relevance&q=%5Bvpn%5D)
* [vpn-сервер](https://habr.com/ru/search/?target_type=posts&order=relevance&q=%5Bvpn-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%5D)
* [proxy](https://habr.com/ru/search/?target_type=posts&order=relevance&q=%5Bproxy%5D)
* [https proxy](https://habr.com/ru/search/?target_type=posts&order=relevance&q=%5Bhttps%20proxy%5D)
* [tls proxy](https://habr.com/ru/search/?target_type=posts&order=relevance&q=%5Btls%20proxy%5D)
* [tls](https://habr.com/ru/search/?target_type=posts&order=relevance&q=%5Btls%5D)
* [tls tunnel](https://habr.com/ru/search/?target_type=posts&order=relevance&q=%5Btls%20tunnel%5D)
* [туннелирование трафика](https://habr.com/ru/search/?target_type=posts&order=relevance&q=%5B%D1%82%D1%83%D0%BD%D0%BD%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20%D1%82%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0%5D)
* [туннелирование ssh](https://habr.com/ru/search/?target_type=posts&order=relevance&q=%5B%D1%82%D1%83%D0%BD%D0%BD%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20ssh%5D)
* [https](https://habr.com/ru/search/?target_type=posts&order=relevance&q=%5Bhttps%5D)
* [socks-прокси](https://habr.com/ru/search/?target_type=posts&order=relevance&q=%5Bsocks-%D0%BF%D1%80%D0%BE%D0%BA%D1%81%D0%B8%5D)
* [socks прокси](https://habr.com/ru/search/?target_type=posts&order=relevance&q=%5Bsocks%20%D0%BF%D1%80%D0%BE%D0%BA%D1%81%D0%B8%5D)
* [socks5](https://habr.com/ru/search/?target_type=posts&order=relevance&q=%5Bsocks5%5D)

**Хабы:**

* [Firefox](https://habr.com/ru/hubs/firefox/)
* [Информационная безопасность](https://habr.com/ru/hubs/infosecurity/)
* [Google Chrome](https://habr.com/ru/hubs/google_chrome/)
* [Сетевые технологии](https://habr.com/ru/hubs/network_technologies/)
* [Софт](https://habr.com/ru/hubs/soft/)
* Защита Wi-Fi сетей и средств связи.

Практика защиты информации в Wi-Fi сетях на основе современных программно-аппаратных средств

Пожалуйста, не забудьте правильно оформить цитату:
Визавитин, О. И. Практика защиты информации в Wi-Fi сетях на основе современных программно-аппаратных средств / О. И. Визавитин. — Текст : непосредственный // Молодой ученый. — 2016. — № 5 (109). — С. 182-184. — URL: https://moluch.ru/archive/109/26458/ (дата обращения: 24.10.2023).

Целью работы является изучение современного состояния средств защиты информации в беспроводных сетях на основе групп протоколов IEEE 802.11, а также разработка комплекса мер для усиления безопасности на основе практических пошаговых рекомендаций. Данное исследование может представлять интерес как для специалистов в области разработки средств защиты беспроводных сетей, так и для технических сотрудников, выполняющих непосредственное администрирование беспроводных сетей. Ключевые слова: IEEE 802.11, беспроводные сети, защита информации, шифрование, безопасность.   Защита беспроводных сетей по технологии групп протоколов IEEE 802.11 (известных под торговой маркой Wi-Fi) является одной из важных задач, стоящих перед разработчиками и администраторами коммуникаций. В общем случае, защита должна обеспечивать невозможность доступа в сеть без разрешения администратора сети, выражаемого в выдаче кодов доступа или специальных устройств доступа. Особенность беспроводных сетей на базе протоколов IEEE 802.11 приводит к следующим сложностям защиты, по сравнению с проводными компьютерными сетями [1]: 1)                 Для подключения к беспроводной сети, не требуется физический доступ к кабелю витой пары или оптоволокну — достаточно находиться в зоне приёма сигнала маршрутизатора; 2)                 Сама передача данных по беспроводному каналу может быть перехвачена и обработана даже без устройства доступа, специальными аппаратными или программными средствами. Основные стандартные меры по защите информации в Wi-Fi сетях. К стандартным мерам защиты относятся программные и аппаратные средства, предназначенные для решения следующих задач: 1)                 Предотвращение несанкционированного подключения к беспроводной сети пользователей; 2)                 Предотвращение доступа к запрещенным ресурсам уже подключившихся пользователей; 3)                 В случае уже произошедшего доступа, выполнить меры по сбору информации для предотвращения следующего инцидента доступа. Как правило, в большинстве случаев выполняются следующие стандартные меры по повышению уровня защиты беспроводной сети [2]: 1)                 Замена ключей доступа на более комплексные; 2)                 Смена протоколов шифрования на более современные и устойчивые к взлому методом перебора; 3)                 Установка программного обеспечения для протоколирования доступа пользователей к ресурсам внутри сети. Отдельными средствами являются меры, направленные на противодействие социальным методам взлома, таким, как доступ легальными техническими мерами с нелегальными целями, или подменой лица доступа из-за удаленности терминала. В общем случае, противодействие таким методам не является задачей технических мероприятий, однако, предлагаемая система аппаратно-программной защиты несколько снижает вероятность взлома за счет «обезличенности» мер защиты и независимости от линейного персонала, обеспечивающего безопасность сети. Реализация аппаратно-программной защиты. Для администраторов беспроводной сети, предлагается расширенный комплекс мер на основе автоматизированного контроля за доступом к сети, программируемой смены ключа доступа и перехода на последние стандарты шифрования. Комплекс предназначен для повышения всех уровней защиты беспроводной сети. Перечислим каждый шаг по усилению защиты. Шаг 1: Контроль доступа за ресурсами. На компьютеры, доступ к которым осуществляется через сеть, устанавливается дополнительное «проксирующее» программное обеспечение, которое записывает в базу данных сведения о случаях доступа к ресурсам, как одобренные, так и отклоненные системой. Схематично, организация такой контролирующей прослойки показана на рисунке 1. Рис. 1. Установка контролирующего программного обеспечения для отслеживания доступа к ресурсам   Как видно, запись инцидентов доступа ведется за пределы защищаемой сети, что даже в худшем случае совершенного несанкционированного доступа, позволит сохранить и расследовать историю инцидента. Шаг 2: Замена протоколов шифрования и доступа Традиционным алгоритмом шифрования данных в сети Wi-Fi является WEP (Wired Equivalent Privacy) [3]. Он повсеместно распространен и легко конфигурируется, однако существенно уязвим, особенно к методам силового перебора. Обязательной мерой для повышения безопасности беспроводной сети является перевод всех маршрутизаторов и клиентских терминалов на протоколы шифрования данных WPA и WPA2, которые представляют собой следующее поколение алгоритмов шифрования [3]. Помимо установки новых алгоритмов для оборудования, необходимо также усиление собственной сети за счет введения виртуальной внутренней сети, известной как технология VPN (Virtual Private Network). Создание VPN вводит дополнительное шифрование поверх уже используемых уровней [4], что на порядок повышает сложность взлома и делает практически невозможным силовой подбор ключей и паролей. Шаг 3: Автоматическая регенерация ключей доступа внутри беспроводной сети. Наконец, исключительной по своей эффективности мерой является автоматическая регенерация ключей доступа, производимая по расписанию и заданному алгоритму на всех устройствах доступа и клиентских терминалах. Такая мера требует разработки и установки специального программного обеспечения, которое выполняет следующие действия: 1)                 Создает новый ключ доступа в соответствии с правилами, заданными администратором сети; 2)                 Устанавливает этот ключ на все устройства, используя для подключения еще действующий предыдущий ключ; 3)                 Повторяет действия не реже периода, заданного администратором сети. Ведение собственной базы ключей позволит избежать повторного использования ранее примененной последовательности, а использование аппаратно-программного генератора случайных чисел сделает создаваемый ключ статистически непредсказуемым [3]. При правильной настройке такого комплекса, силовой подбор ключа доступа становится практически невозможен, даже при полном доступе злоумышленника к каналу связи. На рисунке 2 показана архитектура такого решения. Рис. 2. Архитектура система автоматизированной смены ключей доступа в сети Wi-Fi   Рассмотренные меры позволяют сделать невозможным чисто силовые методы взлома беспроводной сети Wi-Fi и существенно затрудняют прочие способы, такие, как социальные и логические. Для обеспечения максимальной степени защиты, рекомендуется комбинировать предложенные меры с другими, например, контролем доступа персонала и расширенные методы идентификации пользователей с использованием электромагнитных карт или датчиков отпечатков пальцев.