# Легковесные алгоритмы: понятие, требования

## Понятие и область применения легковесной криптографии

Легковесная криптография (LW-криптография) изучает криптографические алгоритмы, ориентированные на использование в небольших вычислительных устройствах, заведомо не обладающих значительными ресурсами.

LW-криптография используется для обеспечения безопасности встраиваемых систем (англ. embedded systems). Встраиваемая система представляет собой микроконтроллер или небольшой компьютер, встроенный в управляемое устройство. Таким устройством может быть станок с ЧПУ или платежный терминал. Данные, которыми оперирует такое устройство, как платежный терминал, представляют собой существенную ценность, поэтому нуждаются в защите. Однако возможна разработка вирусов и «широкого действия», не нацеленных на определенное устройство, но могущих нанести большой вред из-за широкой распространенности встраиваемых систем. Примером вируса такого типа является RFID-вирус[[1]](#footnote-1).

Другим важным применением легковесной криптографии являются задачи обеспечения безопасности устройств интернета вещей. Этому варианту использования LW-криптография и посвящена данная работа.

В интернете вещей LW-криптография применяется главным образом для обеспечения безопасности устройств сенсорного уровня. В то время как безопасность узлов сетевого/транспортного уровня можно организовать на основе модели криптомаршрутизатора [Жданов, стр 217], а остальные компоненты сети IoT (клиентские приложения прикладного уровня, сервера уровней обработки и бизнес-логики) обладают достаточными ресурсами для применения методов общей криптографии.

В контексте использования LW-криптографии в устройствах интернета вещей, эти устройства считаются «атомарными». Они только собирают данные, шифруют их и отправляют на узлы транспортного уровня. И защите подвергается главным образом отправляемый устройствами трафик. Среди основных направлений защиты пакетов данных можно выделить следующие.

* Подтверждение достоверности и целостности пакета данных. Требуется уверенность в том, что пакет отправил законный отправитель, и по пути данные не подверглись модификации злоумышленником или помехам канала передачи.
* Подтверждение подлинности пакета данных. Требуется уверенность в том, что пакет, который мы получаем, это именно тот пакет, который сейчас отправил законный отправитель. А не пакет, который он отправил ранее, а злоумышленник сейчас ретранслировал.
* Обеспечение конфиденциальности пакета данных. Требуется уверенность в том, что злоумышленник не смог расшифровать данные, даже если перехватит пакет.

Существенным фактором является *неинтерактивность* трафика. Это означает, что пакет передается устройством лишь однократно, без долгих процедур аутентификации. Неинтерактивность обуславливается, во-первых, весьма вероятным отсутствием постоянного канала беспроводной связи, а во-вторых, необходимостью быстро передать пакет данных при наличии такого подключения. Неинтерактивность сильно ограничивает количество доступных для использования криптографических методов и протоколов.

Среди других требований к криптографическим алгоритмам и их реализации можно выделить следующие.

* Бизнес-требования:
  + Требуемый уровень безопасности устройства.
  + Стоимость устройства.
  + Итоговая производительность устройства.
* Ограничения аппаратной части:
  + Ограничения энергетических ресурсов.
  + Ограничения по объему ОЗУ.
  + Ограничения размера микросхемы (GE-мера).
* Ограничения программной реализации:
  + Объем программного кода.
  + Количество потребляемой оперативной памяти.
  + Время работы.
* Возможные дополнительные ограничения:
  + Ширина полосы рабочих частот и пропускная способность канала связи.

## Бизнес-требования

Бизнес-требования определяют необходимость поиска компромисса между стоимостью, безопасностью и производительностью устройства. Достаточно легко оптимизировать два из трех свойств, а вот оптимизировать все три – обычно весьма трудная задача для разработчиков.

Процедура определения требуемого уровня безопасности определяется в соответствии с конкретной областью применения конкретной системы. В коммерческих структурах может использоваться, например, модель системы безопасности *с полным перекрытием*. В рамках данной модели перечисляются все защищаемые объекты системы и возможные атаки на них, а также вероятность появления и наносимый ущерб для каждой из атак. После чего определяются приоритетные направления защиты. В контексте систем интернета вещей представляется разумным ввести атаки, общие для всех уровней системы (например, DoS-атаки) и атаки, специфичные для конкретного слоя (например, кодовые инъекции нацелены главным образом на прикладной уровень). Атаки на каждый уровень и меры защиты от них кратко приведены в первой главе данной работы.

Стоимость и итоговая производительность устройства определяется, главным образом, используется ли в этом качестве устройство «общего назначения» (например, смартфон) или же некоторое «кастомное», уникальное аппаратное решение.

В первом случае и процессор, и системы сетевого взаимодействия, и иногда даже системы безопасности уже реализованы, и требуется реализовать только модули для сбора данных и для их шифрования. Обратной стороной являются возможные скрытые уязвимости или сознательно добавленные «бэкдоры» устройства, которые крайне трудно обнаружить. Кроме этого, использование функционала «из коробки» означает необходимость положиться на разработчиков этого функционала и потерю полного контроля над аппаратным обеспечением.

Во втором случае разработка начинается «с нуля» (или почти с нуля). С одной стороны, это требует гораздо больших навыков от команды разработчиков и больших затрат времени и денежных средств на разработку. Однако, с другой стороны, это может позволить создать в итоге более дешевое и надежное устройство – опять же в зависимости от уровня компетенций команды.

## Требования к аппаратной части

В случае использования «кастомной» аппаратной реализации, аппаратная часть должна удовлетворять требованиям по количеству энергетических ресурсов, объему ОЗУ и площади микросхемы. При использовании аппаратного решения «общего назначения» количество имеющихся ресурсов обычно значительно больше, чем требуется для работы криптографических алгоритмов (но, естественно, тоже ограниченно, что необходимо учитывать).

Ограничения энергетических ресурсов могут явно задаваться конкретным технологическим решением (например, RFID-метки рассчитаны не более чем на 15 микроватт[[2]](#footnote-2)). Кроме того, для предупреждения *атак по энергетическим ресурсам* требуется по возможности уменьшить скачки в энергопотреблении устройства. Это отдельная тема, выходящая за рамки данной работы.

Ограничения размера (площади) микросхемы связаны с тем, что зачастую в слабых устройствах (в том числе в устройствах интернета вещей) используются «кастомные» аппаратные решения, направленные на удешевление и повышение надежности устройств. Это означает, что требуется, насколько возможно, уменьшить количество используемых функциональных элементов. Оно определяется, во-первых, используемым алгоритмом, а во-вторых, его программной и аппаратной реализацией. В качестве единицы измерения данного ресурса используется количество элементов NAND (Not And), необходимое для реализации. Обозначается эта величина GE (logic Gate Elements number, количество логических вентилей). Существуют следующие категории реализаций алгоритмов[[3]](#footnote-3):

* Легковесные реализации (lightweight) – не более 3000 GE.
* Низкостоимостные реализации (low-cost) –не более 2000 GE.
* Ультралегкие реализации (ultra-lightweight) – не более 1000 GE.

При создании конкретной аппаратной реализации необходимо искать баланс между скоростью работы и размером микросхемы. Оптимизация размера микросхемы достигается за счет использования последовательной архитектуры (обрабатывающей информацию побайтно), оптимизация скорости – за счет распараллеливания и конвейеризации, что влечет увеличение размера.

Основной стандартом, определяющим количество доступных ресурсов для легковесных криптографических алгоритмов, является ISO 29192[[4]](#footnote-4).

## Требования к программной части

При разработке программной реализации алгоритма необходимо оптимизировать те же направления, что при разработке любого ПО: оптимизация потребления памяти и времени, а также специфичный для данной области параметр – объем кода.

Количество потребляемой памяти особенно критично для «кастомных» реализаций, при использовании более мощной аппаратной части уже не столь критично, хотя все равно подлежит оптимизации.

Время работы состоит из двух величин. Задержка (англ. latency) – время инициализации алгоритма, и пропускная способность (англ. throughput) – количество информации, обрабатываемой в единицу времени. Обычно требования по каждому из этих параметров формулируются отдельно. Например, для систем автоматического осуществления дорожных сборов время реакции устройства должно быть менее 10 миллисекунд [Жуков, стр 29], это ограничение на задержку. С другой стороны, если система должна еще и зафиксировать изображение машины, нарушившей ПДД, то требуется передать (а значит, зашифровать) достаточно большое количество информации, а это требует относительно большой пропускной способности.

Объем программного кода тесно связан с количеством используемого кода. Так, S-блоки, определяющие порядок перестановки блочного шифра, можно хранить в оперативной памяти, а можно напрямую «зашить» в код. Второй способ быстрее, однако исполнение кода тоже требует определенных накладных расходов (при исполнении код также хранится в ОЗУ). Поэтому конкретное соотношение хранимых в ОЗУ и заданных в коде данных определяется под конкретную программу.

# Использование криптографических методов в легковесной криптографии

Как указывалось выше, для защиты трафика устройств IoT необходимо решить следующие задачи:

* Подтверждение достоверности и целостности пакета данных.
* Подтверждение подлинности пакета данных.
* Обеспечение конфиденциальности пакета данных.

Проанализируем основные типы криптографических алгоритмов на пригодность для решения этих задач.

## Подтверждение достоверности и целостности пакета

Фактически эта задача является задачей аутентификации. Она распадается на две части: аутентификация отправителя и аутентификация пакета.

В системах обмена пакетами общего назначения (не «легковесных») обычно используются протоколы аутентификации отправителя. В начале общения с помощью одного из протоколов (CHAP, Kerberos и другие) подтверждается подлинность отправителя или обеих сторон. После чего начинается доверенный обмен сообщениями. Спустя некоторое время требуется повторная аутентификация. Такая схема используется на следующих уровнях модели OSI: канальный (например, протоколы EAP, PPP), сеансовый (PAP) и прикладной (SSH).

Для использования в системах IoT такой метод не слишком подходит, так как пакеты могут отправляться через краткие, длительные или неравные промежутки времени. При этом каждый пакет может быть отправлен или модифицирован злоумышленником. Это обстоятельство диктует необходимость подтверждения достоверности и целостности каждого передаваемого пакета данных. Это позволяет распознать факт введения фальшивого узла.

Аутентификация также может производиться с использованием таких инструментов, как хешированный или нехешированный пароль, а также электронная цифровая подпись (ЭЦП). Эти методы подходят для легковесной криптографии.

ЭЦП применяется следующим образом. Имеется закрытый ключ, он хранится на отправляющем устройстве, и открытый ключ, он хранится на принимающем устройстве и сопоставлен отправляющему устройству. Открытый ключ однозначно вычисляется по закрытому.

* Вычислить контрольную сумму (хэш) пакета данных.
* Подписать хэш ЭЦП (т. е. зашифровать хэш закрытым ключом, получив в результате подпись).
* Отправить пакет данных, снабдив ЭЦП. Принимающая сторона расшифровывает ЭЦП с помощью открытого ключа. Если получился верный хэш документа, значит хэш был зашифрован с помощью верного закрытого ключа, что подтверждает законность отправителя.

Закрытый ключ известен только отправляющему устройству. Злоумышленник его не знает и, следовательно, правильно зашифровать хэш не сможет.

Цифровая подпись, в отличие от пароля, еще и гарантирует целостность доставленного сообщения (защита от случайных или намеренных искажений по пути). Поэтому она является более предпочтительной, хотя и требует бОльших расходов времени, т. к. требуется предварительно вычислить хэш сообщения (зависит от длины сообщения, т. е. влияет на пропускную способность). Шифрование хэша требует уже фиксированного времени (т. е. влияет только на задержку).

## Подтверждение подлинности пакета

Такое подтверждение необходимо для защиты от атаки повторного воспроизведения. В существующих системах представлены различные способы обеспечения такой защиты.

Один из способов реализован в протоколе Kerberos[[5]](#footnote-5). Сообщение снабжается сроком действия и временной меткой. Если срок действия истек, можно соответствующим образом отреагировать: отбросить пакет, зафиксировать подозрительную активность или даже заблокировать трафик от скомпрометированного отправителя.

Схожий подход под названием Hop-by-hop transport[[6]](#footnote-6) используется в IPv4/IPv6 маршрутизации. Заголовок пакета (IP-заголовок) содержит специальное восьмиразрядное поле, в IPv4 оно называется TTL (Time to live), в IPv6 – Hop Limit. Это поле содержит максимальное количество прыжков (передач пакета между маршрутизаторами), и после каждого прохождения через маршрутизатор оно уменьшается на единицу. Если оно достигает нуля, пакет считается устаревшим и отбрасывается.

Другим возможным подходом является добавление так называемой «nonce-вставки». Nonce-вставка (от англ. number that can be used only once) – случайно сгенерированное число, известное обеим сторонам и хранимое ими в тайне. Генерируется сервером, отправляется клиенту, который затем добавляет его к паролю при шифровании. Используется однократно, после чего меняется. Повторение или использование неверной nonce-вставки говорит о том, что пакет отправлен злоумышленником. Не подходит по причине неинтерактивности трафика, передаваемого устройствами IoT (устройство не ждет сигнала сервера, вся коммуникация состоит в одном пакете).

Таким образом, оптимальным вариантом является добавление к зашифрованному сообщению метки времени. Эта операция не является требовательной ни по времени, ни по памяти. Но необходимо наличие системного таймера. При этом исходим из того, что злоумышленник не сможет расшифровать сообщение, а значит и изменить метку не может. Кроме того, метку можно подписать цифровой подписью. Тогда злоумышленник не сможет ее изменить даже в случае, если сможет прочитать данные.

## Обеспечение конфиденциальности пакета

Эта задача является одной из важнейших и наиболее сложных задач, которые необходимо решить при организации защиты данных, передаваемых IoT-устройствами. Данная задача является основным объектом интереса легковесной криптографии.

Итак, задача состоит в шифровании пакета данных. Ясно, что шифрование должно быть обратимым преобразованием. Принимающая сторона должна иметь возможность эффективно (достаточно быстро) расшифровать пришедший пакет данных. Как указывалось выше, требования к скорости работы определяются задержкой (определяется минимальным интервалом между сообщениями) и пропускной способностью (объем данных в пакете, который должен быть зашифрован до начала шифрования следующего пакета). Шифратор не должен потреблять больше доступного количества оперативной памяти, должен тратить как можно меньшее количество энергии (либо энергопотребление должно быть в пределах возможностей устройства, но близким к постоянному, чтобы затруднить криптоанализ по колебаниям энергопотребления). При этом следует помнить о необходимости минимизации требуемого размера микросхемы (хотя бы не более 3000 GE, в идеале – не более 1000 GE) и обеспечении максимального уровня стойкости шифрования.

Таким образом, задача формулируется следующим образом. Какие алгоритмы позволяют достичь максимального уровня производительности и стойкости, требуя при этом а) не более 3000 GE и б) не более 1000 GE?

При выборе способа шифрования в первую очередь следует проверить, не могут ли подойти известные криптоалгоритмы общего назначения. Если удастся, не изменяя сами алгоритмы, выполнить их программную и аппаратную реализацию, работающую в заданных условиях, это будет наилучшим вариантом. Потому что они уже хорошо исследованы, что позволяет рассчитывать на их стойкость.

Следующим шагом должно быть рассмотрение известных (в том числе закрепленных в стандартах, о них ниже) легковесных криптоалгоритмов. И только если и они не подходят, можно попробовать модифицировать их в сторону ослабления и дальнейшего облегчения. Впрочем, это весьма рискованный путь, так как требует от создателей значительных навыков в криптоанализе, чтобы гарантировать стойкость шифра.

Итак, ниже будут рассмотрены на пригодность для легковесного шифрования различные типы шифров общего назначения и легковесных шифров.

### Симметричное блочное шифрование

В дело исследования блочных легковесных шифров внесли значительный вклад работы [[7]](#footnote-7) [[8]](#footnote-8). Легковесные блочные шифры являются одним из наиболее динамично развивающихся разделов низкоресурсной криптографии. Исследования идут по двум направлениям: создание оптимальных реализаций алгоритмов общего назначения и создание новых алгоритмов, нацеленных именно на использование в низкоресурсных устройствах.

#### Криптоалгоритмы общего назначения

Одним из лучших на данный момент блочных шифров является алгоритм **AES** (Rijndael). Он оперирует 128-битными блоками данных и 128-, 192- и 256-битными ключами, производя 10, 12 и 14 раундов шифрования соответственно.

Помимо высокой криптостойкости, он является весьма производительным. Существует его программная реализация с производительностью порядка 7 процессорных тактов на байт[[9]](#footnote-9) на стандартных ЦП. Добавление специальной процессорной инструкции для этого алгоритма позволило достичь производительности примерно 0.7 тактов на байт[[10]](#footnote-10). Помимо этого, стоит отметить низкое потребление памяти.

Он, однако, требует большого размера микросхемы, порядка 250 000 GE[[11]](#footnote-11) для достижения максимальной скорости (до 70 Гб/сек[[12]](#footnote-12)). Наиболее компактная последовательная реализация требует 2400 GE[[13]](#footnote-13) и имеет производительность 226 циклов на блок. Это значительно лучше, и может подойти для многих устройств, даже очень простых. Тем не менее, «ультралегким» алгоритмом, т. е. оптимальная реализация которого требует менее 1000 GE, его все-таки назвать нельзя.

**DES**. Данный алгоритм также является весьма известным. Длина его ключа значительно меньше, чем у AES (56 бит), что означает меньшую стойкость. Также он был взломан методом линейного криптоанализа[[14]](#footnote-14). В системах общего назначения (не легковесных) он применяется, в основном, в виде Triple DES, то есть троекратное шифрование с тремя ключами. Также существует вариация DESX, работающая с 184-битным ключом при таком же размере блока и числе раундов, как у DES. Наиболее компактные реализации DES и DESX требуют 2309 GE и 1848 GE соответственно[[15]](#footnote-15).

#### Легковесные криптоалгоритмы

Наиболее строгим условием по размеру микросхемы для легковесных алгоритмов является максимальная граница в 1000 GE. Ее удалось преодолеть двум алгоритмам, включенным в стандарт ISO/IEC 29192-2 (Block ciphers). Это PRESENT и CLEFIA.

**PRESENT**. Данный шифр использует 80- и 128-битные ключи для шифрования данных 64-битными блоками в 31 раунд. Он, вообще говоря, не совсем укладывается в границу, т. к. требует 1030 GE[[16]](#footnote-16). Но даже такой результат явился почти революционным. Его удалось достичь за счет последовательной архитектуры – это первый шифр с такой архитектурой. Еще одним улучшением является уменьшение числа S-боксов с 8 (обычное количество для блочных шифров) на 1.

**CLEFIA**. Данный алгоритм является очень популярным благодаря существующим очень производительным реализациям. Он оперирует 128-битными блоками с длиной ключа 128, 192 и 256 бит с 18, 22 и 26 раундами, соответственно.

### Симметричное поточное шифрование

### Асимметричное шифрование

### Хэширование и ЭЦП

Криптографическое хэширование однозначно не подходит для целей шифрования, так как является односторонним преобразованием: расшифровка хэша априори невозможна, возможно только сравнение хэшей. Впрочем, блочная сжимающая функция, на которой основана хэширующая функция, может быть интересна с криптографической точки зрения, и часто может быть использована как примитив при создании нового шифра, в том числе легковесного.

Электронная цифровая подпись также не подходит для хэширования, так как использует в своем составе хэш. ЭЦП предназначены для гарантирования достоверности и целостности данных. В этом качестве она вполне может быть использована, о чем написано выше.

1. <http://www.rfidvirus.org/papers/press_release.pdf> [↑](#footnote-ref-1)
2. ISO/IEC 18000-3:2004 Information technology – Radio frequency identification for item management – Part 3: Parameters for air interface communications at 13.56 MHz [↑](#footnote-ref-2)
3. Manifavas C., Hatzivasilis G., Fysarakis K., Rantos K. Lightweight Cryptography for Embedded Systems - A Comparative Analysis, SETOP’2013 [↑](#footnote-ref-3)
4. ISO/ IEC FDIS 29192 – Information technology – Security techniques – Lightweight cryptography. [↑](#footnote-ref-4)
5. <https://www.osp.ru/winitpro/2014/03/13039728/> [↑](#footnote-ref-5)
6. <https://www.researchgate.net/publication/220653823_End-to-end_vs_hop-by-hop_transport> [↑](#footnote-ref-6)
7. <https://www.researchgate.net/publication/316078586_A_review_of_lightweight_block_ciphers> [↑](#footnote-ref-7)
8. <https://cyberleninka.ru/article/n/legkovesnaya-kriptografiya-chast-1> [↑](#footnote-ref-8)
9. Kasper E., Schwabe P. Faster and Timing-Attack Resistant AES-GCM. CHES 2009, LNCS 5747, pp. 1–17, 2009 [↑](#footnote-ref-9)
10. Preneel B. Perspectives on Lightweight Cryptography,” Inscript 2010, Shanghai, China, 20-24 October 2010. [↑](#footnote-ref-10)
11. Hodjat A., Verbauwhede I. Minimum Area Cost for a 30 to 70 Gbits/s AES Processor. In IEEE Computer Society Annual Symposium on VLSI (ISVLSI 2004), pp. 498–502, 2004. [↑](#footnote-ref-11)
12. Dinur I., Dunkelman O., Shamir A. Improved Attacks on Full GOST. Report 2011/558, available via http://eprint.iacr.org/, 2011. [↑](#footnote-ref-12)
13. Moradi A., Poschmann A., Ling S., Paar C., Wang H. Pushing the Limits: A Very Compact and a Threshold Implementation of AES”, EUROCRYPT 2011, LNCS 6632, pp. 69-88, Springer-Verlag, 2011. [↑](#footnote-ref-13)
14. P. Junod, On the Complexity of Matsui’s Attack, Selected Areas in Cryptography (SAC’01), Springer, LNCS, 2259, 2001, pp.199211. [↑](#footnote-ref-14)
15. G. Leander, C. Paar, A. Poschmann, and K. Schramm, New Lightweight DES Variants, Fast Software Encryption, FSE 2007, Springer, LNCS, 4593, 2007, pp.196210. [↑](#footnote-ref-15)
16. 117. C. Rolfes, A. Poschmann, G. Leander, and C. Paar, Ultra-lightweight implementations for smart devicessecurity for 1000 gate equivalents, Smart Card Research and Advanced Applications, Springer, LNCS, 5189, 2008, pp.89103. [↑](#footnote-ref-16)