Республика Казахстан

Министерство образования и науки

Академия Кайнар

М.М. ИЛИПОВ, Г.Б. ИСАЕВА

РАЗРАБОТКА ЭФФЕКТИВНЫХ АЛГОРИТМОВ ЗАЩИТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ МИКРОПРОЦЕССОРНЫХ КАРТ

монография

Алматы 2020

УДК 004 (075.8)
ББК 32. 973. 202 я73
И 43

Рекомендовано к печати Ученым советом Академии Кайнар
(протокол №1/65 от 28.08.2020. г)

Рецензенты:
д. и. н., профессор Е. С. Андасбаев, Жетысуский университет им. И. Жансугурова.
к. п. н., профессор, Г. Б. Алимбекова, Казахский национальный педагогический университет имени Абая

Илипов М.М.
И43 «Разработка эффективных алгоритмов защиты программного обеспечения микропроцессорных карт»: *Авторы: , Академия Кайнар, магистр..ф.к.т., М.М.Илипов., к.п.н Г.Б.Исаева: монография.*/ Алматы: Академия Кайнар, 2020.-184 бет.

ISBN 978-601-7984-81-6

Монография посвящена предусмотрены виды педагогических технологий и средств обучения. Широко раскрыты методы и приемы использования новых технологий, продемонстрированы дидактические возможности технических средств обучения. Особое внимание уделено методике использования электронных учебников.

Монография предназначено для преподавателей и мастеров обучения, работающих в высших учебных заведениях и профессиональных колледжах.

ISBN 978-601-7984-81-6

УДК 004 (075.8)
ББК 32. 973. 202 я73

Введение

В современных условиях развития мирохозяйственных связей происходит процесс интеграции экономик отдельных государств и изменение условий функционирования платежных систем, что находит свое проявление, прежде всего, в развитии безналичных форм расчетов. Одним из инструментов безналичных расчетов, бурное развитие которого наблюдается в течение последних лет, является банковская карта, его широкое применение одновременно характеризует степень интегрированности банковской системы и общества, развития банковских операций и платежного оборота.

Очевидно, что уровень развития платежных систем позволяет решать весьма актуальную для банковского сектора проблему — поддержание ее ликвидности. В то же время в условиях глобализации мирового финансового пространства происходит втягивание национальных экономик различных стран в мировой финансовый кризис. В этой связи, на современном этапе следует применять экстренные меры, направленные на укрепление устойчивости казахстанской банковской и платежной систем. Нехватка ликвидности, ощущаемая в последнее время, снижает доверие к банковскому сектору, приводит к оттоку депозитов и сворачиванию деловой активности. Все это обуславливает потребность в перестройке денежно-кредитных отношений, в том числе и платежного оборота.

Платежные карты, являющиеся по своей сути инструментом расчетов высокой ликвидности, в полной мере подвержены воздействию кризисных явлений. В связи с этим вопрос о перспективах развития платежного оборота, в том числе на базе широкого применения банковских карт, приобретает особую значимость в современных условиях.

В научном мире платежную карту называют одним из ключевых элементов технологической революции в банковском деле. В этой связи изучение проблем функционирования и расширения границ использования пластиковых карт в Казахстане представляется особенно актуальным и приобретает в настоящее время практическую значимость.

Экономисты называют платежную карту одним из ключевых элементов технологической революции в банковском деле. Поэтому изучение проблем использования и более обширного внедрения пластиковых карт в Казахстане представляется особенно актуальным и приобретает в настоящее время конкретную значимость.

Платежная карта - один из самых динамично развивающихся инновационных банковских продуктов во всем мире, с каждым годом обнаруживаются новые возможности использования данного инструмента расчетов. Стоит отметить, что для Казахстана важны не только изменения самого продукта и новые возможности его использования, но и макроэкономический аспект внедрения платежных карт в оборот. Ведь платежная карта - это не только средство для формирования устойчивых пассивов, она также является одним из самых мощных инструментов, оказывающих влияние на структуру платежного оборота. Поэтому изучение карточных платежных систем на макроуровне и развитие соответствующих теоретических исследований приобрело важное значение для развития экономической науки. Кроме того, стоить отметить высокий уровень интеграции платежных карт в международное экономическое сообщество и непосредственную связь с многочисленными банковскими продуктами: от текущих счетов до паевых инвестиционных фондов. Постоянное расширение использования платежных карт на практике обуславливает необходимость дополнительного теоретического.

В Казахстане, где остро стоит проблема зависимости от международных карточных платежных систем и недостаточной разработанности нормативной базы, регулирующей использование платежных карт, особую актуальность приобретают исследования, направленные на изучение перспектив развития карточного платежного оборота и направлений его совершенствования.

Глава 1. Анализ методов и технологий современных микропроцессорных карт.

1.1. Микропроцессорные карты классификация их по функциональным возможностям.

Начало возникновения микропроцессорных карт берет с 50-х гг. нашего века в США. Это были первые кредитные карты для оплаты счетов в ресторанах и гостиницах с рельефным рисунком, который копировался на бумажный бланк счета. Держатель карты ставил в ручную свою подпись, а чтобы узнать платежеспособность при снятии с карты связывались с организацией-эмитентом по телефону. Это были ранние версии смарт-карт.

В 60-х гг. второй половины использовались магнитные карты с полосой. В начале они применялись для выдачи денег в банковских терминалах, позже в качестве пропусков, транспорте, в удостоверении личности.

В 1974 г. французский инженер Ролан Морено предложил вмонтировать микросхему в микропроцессорные карты.

Самым ранним были полупроводниковые (микроэлектронные) карты со встроенными микросхемами памяти и логическими схемами с допуском к памяти и выполняли простые функции. Чуть позже появились гибридные карты имевшие в себе микросхемы и магнитную память. Смарт-карты начали использовать в 1982-84 гг. во Франции. Необходимо учитывать уровень надежности, безопасности информации, ущерб от потери или несанкционированного доступа к информации в смарт-карте. Например: С 1997 г. автоматизированная система оплаты проезда в зарубежном и стран СНГ, а также в Алматинском метрополитене с недавних пор.

Также используется одноразовых и месячных проездных билетов используется магнитный билет в соответствии со стандартами ISO, а льготные варианты билетов выпускаются для долговременного применения как бесконтактные микропроцессорные карты. Также вышли в свет и используются и другие типы карт, между ними и есть более известные это оптические его особенности:

- Хранение большие объемы информации.
- Стойкость к внешней среде.
- Считывание информации.
- Сложна в изготовлении и дорогостоящее сравнительно с картами с памятью.

Стандарты оптической карты и ее физические свойства и техника линейной записи данных ISO/IEC 11693, 11694.

Существуют такие виды карт: штрих-кодовые, перфорационные, голографические с устройством ввода/вывода(микроклавиатура, жидкокристаллический дисплей, сенсоры и др.), но такие не были эффективные, т.к легко подделать, неудобные и дорогие.

Рис. 1. Классификация пластиковых карт

Карты с магнитной полосой (magnetic stripe cards)

Такие карты начали применятся в конце 60-х гг. в США для выдачи наличных денег (Automatic Teller Machines-ATM).

В них имеется функции хранения данных, запись и чтения карты. В стандарте ISO 50,51 должны иметь магнитные дорожки, в одном из низ записаны данные, а другом чтение. Недостатки таких карт:

- Легкость подделки;
- Малая надежность;
- Ограниченная память;
- Пассивны, легко прочитать и перезаписать;
- Дорогостоящие устройства и ненадежные.
- В них используются простые средства защиты от подделок:
- Магнитные водяные знаки (magnetic watermarks);
- Метод "Сэндвича" т.е одна полоса с участками разных уровней намагниченностей.

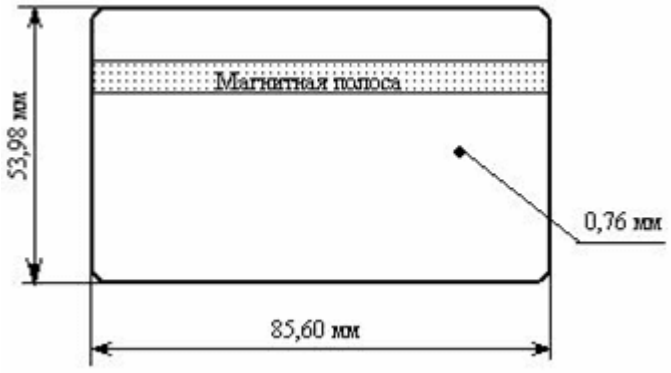


Рис.2. Геометрия магнитной карты.

На сегодняшний день такие карты применяются с малоценной информацией.
Чтобы не нанести большого ущерба: проездные документы со сроком действия на транспорте. Однако по всему миру еще существует проблема перевода платежной системы, так то в банковском секторе еще используются магнитные карты. А банковские платежные системы в России недавно, проблема перешла к нам с платежными системами мира.

Преимущество таких карт в их низкой стоимости и простоте технологии производства.
Самым новым технологическим достижением в этой области стала разработка удаленного (Бесконтактного) чтения данных с карты (The remote read magnetic tag system), может совместить преимущества радиоинтерфейса, низкая стоимость и простое применение данных. Поэтому магнитные карты будут все еще применятся в автоматизированных системах, где это нужно.

Полупроводниковые карты
После магнитных карт появились полупроводниковые карты. В такой карте имеется интегральная схема, в ней объедены два типа памяти:

- постоянная (ROM);
- Перепрограммируемая (имеет качество перезаписи) постоянная память (EEPROM) логическая схема для управления ячейки памяти;

На внешней части карты есть металлические контакты, чтобы взаимодействовать с устройством. Взаимодействия с аппаратурой происходит через физический контакт электрических цепей карты и устройства доступа. А доступ к ячейкам памяти карты через управляющую логическую схему.
Виды полупроводниковых карт:
1. Карты с памятью (memory cards) - постоянная память (ROM), перепрограммируемая постоянная память (EEPROM) с логической схемой для управления памятью и объемом до сотни байт.
Для более продвинутых моделях применяются микроконтроллеры набором команд для управления памятью, предназначенные, чтобы формировать управляющие сигналы для доступа памяти. К чему привело к созданию карты с защищенной памятью, функции чтения, запись, стирание информации с помощью секретного ключа, при идентификации с информации владельца. Чаще всего ключ вычисляется динамически как одна из параметров карты.
Карты-счетчики -это карты с незащищенной памятью выполняют дискретное снижение записанное в память величины. Такие карты применяются в таких местах: телефонных картах, приводящим чаще всего к недостачам, выпуск таких карт обходится дешевле чем микропроцессорных карт, а снабженные с рельефными обозначениями защищали от подделок.

2. Карты с логической схемой (hardwired logic cards) - имеют Rom, Prom (программируемую постоянную память), EEPROM хранение данных, управление памяти с логической схемой выполнено в одной интегральной микросхеме. Такие карты с аппаратной логикой выполняют алгоритмы обработки данных с разрешение обращения к памяти. Это следующий уровень от карт с памятью к смарт-картам. Чтобы не было конфликтной ситуации в системе появились гибридные с магнитной полосой с интегральной микросхемой и металлическими контактами, а передача данных только через внешние устройства.

Недостатки всех перечисленных типов карт:

- неавтоматизированная обработка информации;
- невозможно перепрограммировать определенные функции, такие производители полупроводниковых карт как-Gem plus, Schlumberger (Франция), Giesecke & Devrient, Orda Kartensysteme (ФРГ). Все карты в соответствии со стандартами ISO 7816-1,2,3. Стоимость с аппаратной логикой от15 центов до 40 центов. Использования таких карт в таксофонах, в медицине, в удостоверении личности, торговых автоматах, в системах парковки автомобилей, в предоплаченных схемах выполняющие функции наполняющая платежная карта с функцией аутентификация, счетчик транзакций, или проверкой идентификации информации, защита от чтения, записи, со счетчиком ошибок.

Смарт-карты (Микропроцессорные карты)

От английского "смарт-карты"-пластиковые карты со встроенным микропроцессором от русскоязычного "интеллектуальная карта" (ИК). Различие ИК от пассивных устройств способность к обработке хранимой и поступающей информации с помощью встроенного микропроцессора. с использованием микропроцессора неузнаваемо изменило роль пластиковой карты в автоматизированной системе: микропроцессор ведет контроль доступа к памяти смарт-карты во время взаимодействия с автоматизированной системой, выполняя специфические функции криптографического характера, а выполнение алгоритмов связанные с финансовыми транзакциями, выполняя криптографические преобразования информации (схема шифрования, электронной цифровой подписи, аутентификации). Также смарт-карты снабжают средствами аутентификации (биометрического и логического характера).

Характеристики современных смарт-карт такие: объем ROM от 2 до 20 Кбайт, RAM от 128 байт до 1 Кб, EEPROM от 1 до 8 Кбайт, 8-разрядные микропроцессоры (бывают проекты ИК с 16 и 32 разрядными RISC-процессорами). Например европейский проект разработки 32-разрядного RISC-процессора Cascade-Chip Architecture for Smart Cards and Portable Intelligent Devices), тактовая частота ИК от 5 до 15 МГц. Также 8-разрядный МП 68HC05 фирмы Motorola. Стоимость смарт-карты от 60 центов до 5 $.

Показатели ИК приближаются по своим функциональным возможностям к первым моделям персональных компьютеров, а отличие в том ИК не имеет собственного источника питания, питается за счет энергии передаваемой от устройства считывания в момент взаимодействия с картой. Чаще всего не имеют встроенных средств ввода/вывода и отображения информации. А последние выпущенные модели карт уже с радиоинтерфейсом и с собственным источником питания.
Бывают два типа смарт-карт, различают их принципом взаимодействия с устройством считывания: Контакные и бесконтактные.

1. Контактные карты (полупроводниковые карты с памятью), с металлическими контактами взаимодействуя с устройством считывания посредством передачи электрических сигналов (рис. 3). Стандарт ISO 7816 предусматривает наличие на карте с микросхемой восемь контактных площадок, из которых реально используются только шесть:

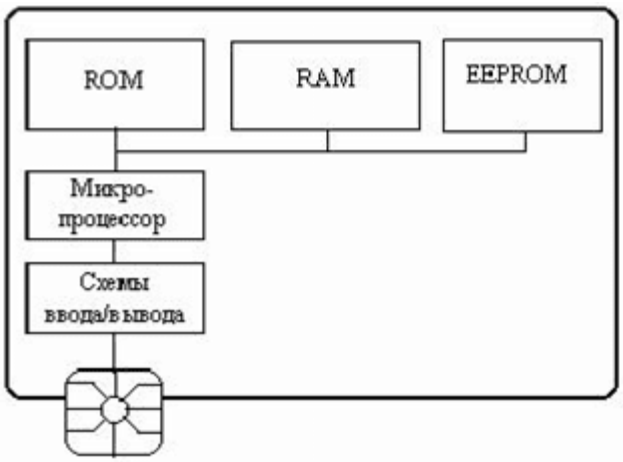
- Vcc - контакт для подачи рабочего напряжения питания (5 В);
- GND - “земля”;
- RST - контакт инициализации карточки;
- CLK - контакт тактового генератора;
- I/O - контакт ввода/вывода;
- Vpp - контакт подачи напряжения записи данных в память (обычно от 12,5 до 15 В). Популярные фирмы, выпускающие контактные ИК: Bull PTS, Gem Plus Card International, Schlumberger Technologies, Motorola, Siemens Nixdorf, Philips, SGS Thompson, Solaic, Orga Kartensysteme GmbH.

2. Бесконтактные карты работая с терминалом (ридером) с помощью радиоканала обмена данными (используя фазовую модуляцию сигнала). Карта и ридер в комплекте с антеннами (контурами индуктивности) (рис. 4).
Основными характеристиками таких карт- это предельное расстояние взаимодействия. Приблизительно от 2 мм до 1 м. В настоящее время появились карты с дальностью связи до нескольких метров. Также важнейшей характеристикой является частота и устройство доступа электромагнитного сигнала. Частоты сигнала соответствуют стандартам ISO/IEC 14443:

- в низкочастотной области - до 135 кГц;
- в диапазоне средних волн - 6.78 и 13.56 МГц;
- в высокочастотном диапазоне - 915 МГц, 2.45 и 5.8 ГГц.

Бесконтактные карты делятся на «близкодействующие» (proximity, close coupling cards) и «дальнодействующие» (remote couple cards), или удалённый доступ.
«Близкодействующие» нужно физический (но не электрический) контакта с устройством доступа: Держателю карты нужно прикоснуться картой к поверхности устройства доступа. Максимальное расстояние взаимодействия этих карт - 2 мм. Преимущества этих карт перед контактной в том, что:
1) Без быстро изнашивающихся и загрязняющихся металлических контактов;

2) Взаимодействия бесконтактной карты происходит быстрее, чем контактной.



Удаленный доступ карт осуществляют радиочастотную передачу данных до устройства на расстоянии от 1 до 1м. Бывают два вида таких карт: пассивные и активные.

Рис. 3. Схема устройства контактной ИК.

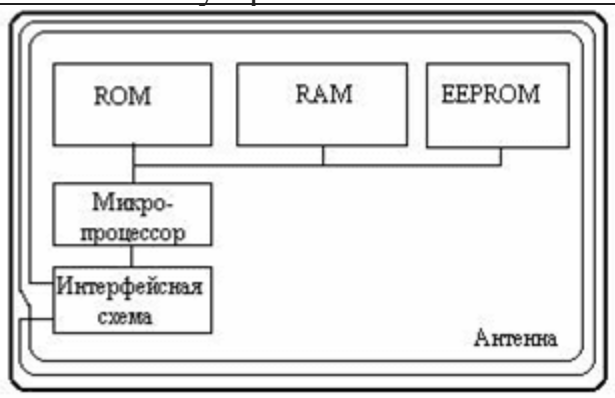


Рис. 4. Схема устройства бесконтактной ИК

Пассивные карты бывают без внутреннего источника питания. Энергия проходит через электромагнитное поле от устройства доступа. Важнейшим элементом пассивной бесконтактной карты -антенна, защищённый модуль микросхема и пластиковая основа.

В активных картах бывают встроенные батареи, обеспечивая энергией для радиосвязи. Если частота обмена >300 МГц карта требует обязательного наличия встроенного источника питания, а при частотах <13,56 МГц и ограничении расстояния между ридером и картой, получать энергию для питания смарт-карты от радиосигнала. С удалённым доступом карты, делят на карты фиксированного взаимодействия (remote fixed couple) - максимальное расстоянием связи <10 см - и карты свободного взаимодействия (remote loose coupling) - максимальное расстоянием связи >10 см. Карты фиксированного взаимодействия обеспечивают устойчивую связь с ридером и ограниченная область вблизи от приёмной антенны. Карты со свободным взаимодействием (их еще называют «free-hand cards») могут

взаимодействовать с устройством с большого расстояния, что удобно держателю карты, даже не показывая носить в карманах.

3. Гибридные микропроцессорные карты (hybrid cards) - Гибрид обычных микропроцессорных карт с контактными и бесконтактными картами. Антенна и микросхема внутри одной пластиковой карты, но внутри они не соединены между микросхемами, имеющий контактный и бесконтактный интерфейс. Данными можно обмениваться только через внешнее устройство, также как и с гибридным картами с полупроводниковой памятью и магнитной полосой. Например комбинация платёжной карты и бесконтактной карты-пропуска в закрытое помещение.

4. Микропроцессорные карты с двойным интерфейсом (dual interface, combi-cards) соединяют преимущества микропроцессорные карты обоих типов. Два интерфейса и одна интегральная микросхема в ней: контактный и бесконтактный (радиоинтерфейс), имеет доступом к общей памяти и одному микропроцессору на (рис. 5).

С доступом к одним и тем же ресурсам и двойным интерфейс позволяет, делать авторизацию карты и закинуть на неё определенную сумму на стационарном контактном устройстве, при необходимости списывать с неё сумму оплаты либо проверять карту можно в городском транспорте, при посадке в поезд при помощи бесконтактного ридера.

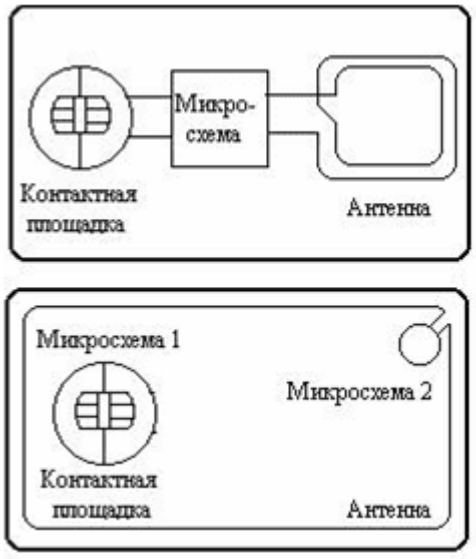
На рис. 6 показаны различия в устройстве гибридной карты и с двойным интерфейсом:

- С двумя интерфейсами карта имеет одну микросхему МЭС, к которой через интерфейсные схемы соединены контактные площадки и антенна в виде контура индуктивности;
- В гибридной карте есть два электрически не соединённых между собой микросхемы МЭС1 и МЭС2, к одному из них соединены металлические контакты, а к следующей - антенна.

С двумя интерфейсами карты, отличаются внутренней организации и доступа к памяти (рис. 7): область памяти может быть общей для обоих типов или раздельной как совместного и общего пользования.

Бесконтактные карты с двойным интерфейсом, бывают со встроенным криптографическим сопроцессор, что к примеру, осуществляют с помощью карт удалённого доступа к вычислительной сети с процедурой аутентификации держателя, используя криптографические методы и протоколы безопасной передачи данных между картой и узлами сети.

Такой вид карт является более совершенным с технологической точки зрения и удобным с точки зрения держателя устройством. Большое значение микропроцессорной карты двумя интерфейсами используют с разработкой многофункциональных карт, которые способны работать в различных системах, и бывают разного типа интерфейса. Одними из наиболее известных идеологов и разработчиков смарт-карт с двумя интерфейсами такие фирмы как IBM, AT&T, Siemens Nixdorf.



Бывает рассказывают о суперинтеллектуальных картах, имея в виду с собственным источником питания со средством ввода и отображения информации (микроклавиатуры, ЖК-дисплея, сенсорных контактов и т.п.). Вот такие модели карт похожи к электронным записным книжкам и ПК-о, но еще пока дорогие для широкого использования.

Рис. 5. Гибридные микропроцессорные карты и с двойным интерфейсом карты.

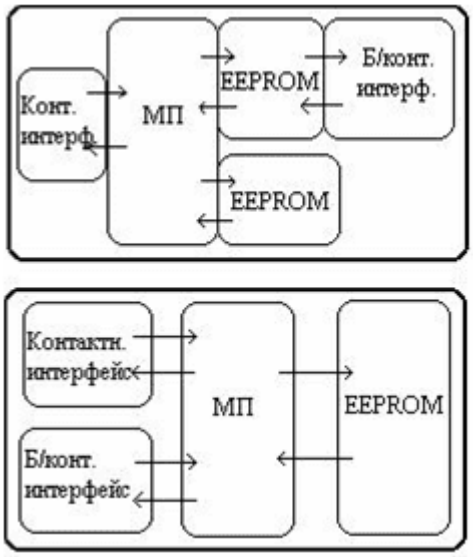


Рис. 6. Способы внутренней организации смарт-карт с двойным интерфейсом.

Методы защиты от подделки микропроцессорных карт. Выпуск микропроцессорных карт возможно лишь в промышленных условиях. Тонкости их технологии производства такие, что достигнуть низкой себестоимости карт возможно только при определенных объёмах их выпуска. Не все крупные и средние фирмы выпускавшие микропроцессорные карты продолжили их выпуск. По экономическим соображениям стала нецелесообразной подделка разовой карты или выпуск небольшими сериями.

Производители и разработчики микропроцессорных карт предусматривают еще и другие методы защиты микропроцессорных карт от возможного «обмана» злоумышленниками устройств (терминалов), работающих со смарт-картами.
Ниже некоторые из них:

- Эмбоссирование - рельефные надписи, нанесение термическим способом на поверхности пластиковой карты ф.и.о держателя, названия банка и т.п.
- Голографические изображения;
- Нанесение дифракционной решётки;
- Нанесение надписей и рисунков, которые видны в ультрафиолетовом свете;
- Использование специальных красителей;
- Нанесение микрорисунка и микротекста;
- Фотография и полноцветная графика.

1.2. Архитектура и аппаратная организация микропроцессорных карт.

Архитектура бесконтактных и контактных микропроцессорных карт схожие. Различие их в физических принципах реализации интерфейса между терминалом (считыватель) и картой.

Контактные и бесконтактные интерфейсы микропроцессорных карт.

Контактный интерфейс микропроцессорных карт с терминалом представляет электрическое соединение между встроенным в карту терминалом и микросхемой, которое происходит через контактную площадку на лицевой поверхности карты.

По международному стандарту ISO 7816-1 определено, что микросхема может подключатся к восьми электрическим контактам с обозначениями от C1 до C8, расположенные на контактной площадке с лицевой поверхности карты, а некоторые контакты электрически соединяются с чипом, встроенная в карте, а некоторые не соединяются для расширения возможностей в будущем.

На рис. показана назначение контактов и контактная площадка обеспечивающая восемь отдельных соединений со встроенной микросхемой. Для работы полудуплексного канала ввода/вывода в карту из нее используется два контакта контактной площадки. В контактном интерфейсе микропроцессорных карт два резервных контакта (RFU) на контактной площадке используются для обеспечения полудуплексного канала через USB (контакты D+ и D-). Упрощено соединение с хост-компьютером, уже не нужен специальный считыватель для микропроцессорных карт: соединение осуществляется через USB-порт хост-компьютера.

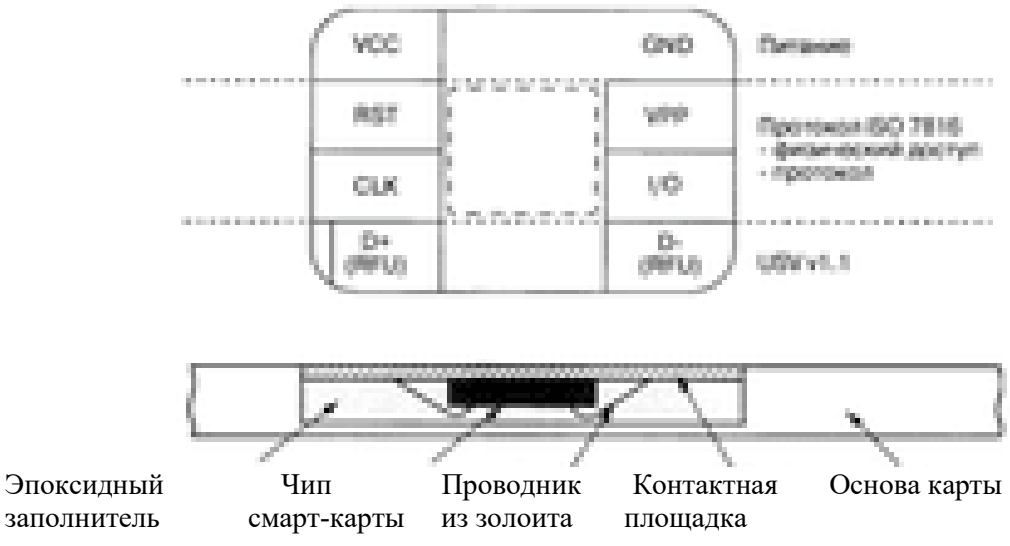


Рис. 6.2. Модуль микросхемы и контактная площадка смарт-карты

Таблица 6.2. Описание контактов смарт-карты

C1	VCC (supply voltage)	Подключение питания для подачи рабочего напряжения на микропроцессорный чип карты
C2	RST (reset input)	Линия сброса, через которую IFD может передать сигнал микропроцессорному чипу смарт-карты для инициирования последовательности команд начальной установки
C3	CLK (clock input)	Линия тактового сигнала, по которой тактовый сигнал может передаваться на микропроцессорный чип смарт-карты для управления скоростью его работы и для синхронизации при передаче данных между считывателем и смарт-картой
C4	RFU (reserved for future)	Зарезервировано для применения в дальнейшем
C5	GND (ground)	Линия заземления, обеспечивающая общую электрическую «землю» между считывалетем и смарт-картой
C6	VPP (programming voltage)	Подключение программирующего питания для обеспечения отдельного источника электрического питания (от рабочего напряжения), который можно использовать для программирования ЭСППЗУ микропроцессорного чипа
C7	I/O (input/output)	Линия ввода/вывода, обеспечивающая полудуплексный канал связи между считывателем и смарт-картой
C8	RFU	Зарезервировано для дальнейшего использования

Также на рисунке показана компоновка интегральной схемы и ее соединение в модуле, заключенном в эпоксидную смолу. Такая компоновка

микросхемы микропроцессорных карт обеспечивает как защищенность микросхемы от внешнего вмешательства, так как есть способность засвидетельствовать вмешательство. Через эпоксидальный состав нельзя проникнуть не разрушив его, а для того чтобы сделать нужно завладеть картой. При проникновении в модуль остаются следы о факте вмешательства. Сказанные свойства конструкции микропроцессорной карты факторы ее физической безопасности.

Для контактов микропроцессорной карты установлены строгие механические допуски обеспечивают сочетание и надежное прикосновение контактов карты и терминала. Контакты не должны отклонятся от поверхности карты больше чем на 0,1 мм., также карта должна быть прочной и выдерживать деформацию при сгибание, если надавить на ее поверхность карта должна возвращаться в плоское состояние. Электрическое сопротивление контактов должна попадать в допустимые пределы, установленные стандартом ISO 7816-1. В последнее время надежность таких карт повышается благодаря опыту при производстве таких карт. Теперь отказы телефонных карт в течении ожидаемого срока службы меньше одного на тысячу, но контакты также входят в число наиболее частых источников отказов в электромеханических системах, а нарушения бывают при загрязнении и ли износа контактов. Когда оборудование подвижна вибрирует и возникает кратковременные размыкания контактов, так контакты расположены на поверхности карты подсоединены к входам интегральной схемы, вмонтированная в карту, поэтому существует риск повреждения интегральной схемы электростатическими разрядами - заряды в несколько тысяч вольт не редкие.

Бесконтактный интерфейс микропроцессорной карты

Такие карты позволяют преодолевать отмеченные технические трудности., а также не нужно электрического соединения между картой и терминалом, для передачи энергии и данные на короткие расстояния. Здесь тоже два компонента, а точнее карту и терминал функционирующий как считыватель или устройство считывания/записи зависит от используемой технологии. Терминал имеет дополнительный интерфейс, осуществляющий связь с базовой системой.

- На рис. Чтобы бесконтактная микропроцессорная карта может осуществлять связь с терминалом, должна выполнять следующие четыре процесса: передача энергии для снабжения электропитанием интегральной схемы микропроцессорной карты;
- передача тактового сигнала;
- передача данных на микропроцессорную карту;
- передача данных от микропроцессорной карты на терминал;

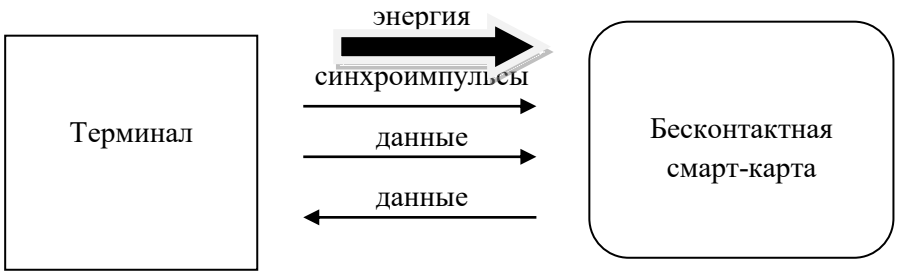


Рис. 6.3. Направления передачи энергии и данных между терминалом и бесконтактной смарт-карты

Методы используемые в таких картах не новые, они используются в системах бесконтактной радиочастотной идентификации RFID, а для идентификации системами RFID объектов на коротких или длинных расстояниях разработаны много методов и они основаны на радиотехнических и радиолокационных принципах, но для использования в микропроцессорных картах подходят лишь некоторые, так как функциональные элементы бесконтактного интерфейса должен разместится в гибкой тонкой карте, толщиной 0,76 мм. При такой толщине карты невозможно применение батарей, для подачи питания на электронные схемы карты.

На сегодняшний день индуктивная связь широко используется методом создания бесконтактного интерфейса между микропроцессорной картой и терминалом. Индуктивная связь используется для передачи энергии и данных. На рис. основные конструктивные элементы бесконтакной микропроцессорной карты (без передней защитной пластины).

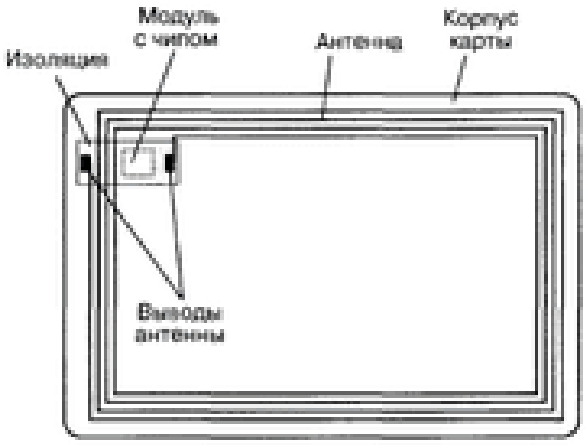


Рис. 6.4. Основные конструктивные элементы бесконтактной смарт-карты

В составе микропроцессорной карты индуктивная антенна и чип с интегральной схемой. Для надежной защиты интегральной схемы чип помещается в миниатюрный модуль, который подключается к концам антенны. Благодаря индуктивной связи с антенной терминал передает по

воздуху к бесконтактной карте как энергию, так и данные. Энергия магнитного поля, принимаемая вмонтированная в карту антенной, преобразуется в напряжение постоянного тока для подачи питания на внутренние схемы карты. Различие требований и внешних ограничений, как правила лицензирования радиочастот, приводить к многообразию конкретных реализаций, а для некоторых приложении, к примеру контроль доступа - достаточно только считывать данные с карты. Низкая потребляемая мощность (несколько десятков микроватт) используемая дальность считывания карт ограничивается примерно 1 м, а емкость памяти несколько сотен битов, при записи данных, потребляемая картой мощность превышает 100мкВт. Дальность в режиме записи ограничиваются примерно 10см, так как лицензионная ограниченная мощность излучаемая терминалом, не может быть увеличена произвольно. Бесконтактные карты с микропроцессором характеризуются с более высокой потребляемой мощностью - около 100 мВт, а расстояние от терминала ограничивается в нескольких миллиметров. Независим от расстояния и потребляемой мощностью все карты применяющие индуктивную связь работают на одном и том же принципе. К одному и более чипам в тело карты встроены одна или более обмоток. На сегодняшний день нет такой батарей, которые были такими тонкими и гибкими для встраивания в микропроцессорные карты, а по соображениям экологической безопасности целесообразно использовать в микропроцессорных картах батарей, так как многие из них содержат ядовитые вещества. Вот почему вся энергия, необходимая для работы чипа в микропроцессорной карте, должна передаваться к микропроцессорной карте от считывателя.

Эта передача энергии основана на принципе слабосвязанного трансформатора, чтобы передать энергию, в обмотке терминала генерируется сильное высокочастотное магнитное поле на рис. Используются частоты 125 кГц и 13,56 МГц. Если карту внести в окрестность терминала, через обмотку этой карты проходит часть магнитного поля терминала, которое наводит напряжение U_i. Это напряжение выпрямляется, чтобы служить в качестве источника энергии для чипа. Связь между обмотками терминала и карты слабая, эффективность этой схемы передачи энергии низкая. Поэтому в обмотке терминала необходимо высокие значения тока, чтобы добиться необходимых уровней напряженностей магнитного поля. это достигается параллельным соединением конденсатора C_T с обмоткой L_T и выбором такого значения конденсатора C_T чтобы резонансная частота параллельного резонансного контура совпадала с частотой предаваемого сигнала.

В микропроцессорной карте обмотка L_C и конденсатор C_1образуют резонансную цепь с той же самой резонансной частотой на рис. Напряжение индуцируемое в карте, пропорционально частоте сигнала, числу витков обмотки L_C и площади охватываемая обмоткой. Число витков обмотки

снижается с увеличением частоты сигнала. На частоте 125 кГц число витков обмотки должно быть от 100 до 1000, а то время как на частоте 13,56 МГц нужно только от 3 до 10 витков. Для передачи от терминала к карте используются известные методы цифровой модуляции.

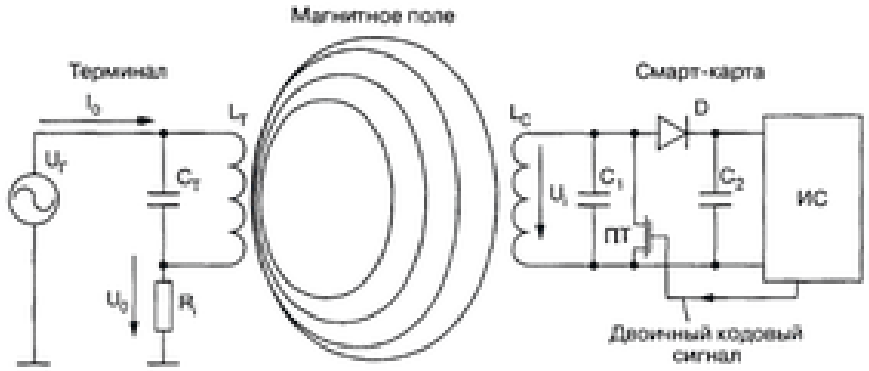


Рис. 6.5. Использование индуктивной связи для передачи энергии от терминала на смарт-карту и данных со смарт-карты на терминал

Наиболее часто употребляют метод амплитудной модуляции ASK, частотная модуляция FSK, фазовая модуляция PSK. Обычно используются ASK и PSK, так как они удобны для выполнения демодуляции. В передаче данных в обратном направлении - от микропроцессорной карты к терминалу -используют разновидность амплитной модуляции, называемую нагрузочной модуляцией. Она создается путем дискретного изменения нагрузки в карте с помощью сигнала данных, подлежащих передаче на терминал. Если микропроцессорная карта, настроенная на резонансную частоту терминала, вносится в ближнее поле терминала, она извлекает из этого поля часть энергии, как было описано ранее. Это вызывает увеличение тока I_0 в обмотке связи терминала, что может быть детектировано как возросшее падение напряжения на внутреннем сопротивлении R_i терминала. Микропроцессорная карта может модулировать амплитуду напряжения U_0 в терминале путем изменения нагрузки в своей обмотке, к примеру переключая в состоянии "включено " или "выключено" полевой транзистор (ПТ)-рис. Если переключение ПТ управляется сигналом передаваемых данных, тогда эти данные могут быть детектированы и восстановлены в терминале.

Из-за слабой индуктивной связи между обмоткой в терминале и обмоткой в карте изменения напряжения, индуцируемого в терминале нагрузочной моделяцией, очень малы. На практике амплитуда используемого сигнала составляет только несколько милливольт. Такие сигналы могут быть выделены только сложной схемой, поскольку на них накладывается значительно большой сигнал(около 80 дБ), передаваемый терминалом.

Однако, если применить дополнительный поднесущий сигнал с частотой f_A , тогда сигнал принимаемых терминалом данных появляется в виде двух боковых полос на частотах ($f_T \pm f_A$), где f_T – частота несущего сигнала терминала. При использовании полосового фильтра сигнал одной из боковых полос может быть отфильтрован от значительно более сильного сигнала терминала и затем усилен. После этого его можно легко демодулировать. Недостатком модуляции с использованием поднесущей является необходимость в более широкой полосе пропускания, чем при прямой модуляции. По этой причине данный метод может быть использован не на всех диапазонах частот.

Кроме отмеченных выше технических преимуществ технология бесконтактных смарт-карт предлагает ряд новых возможностей для пользователей. В частности, не требуется обязательно вставлять бесконтактную карту в считыватель карт, поскольку существуют системы считывания, которые действуют на расстоянии до 1 метра. Это свойство очень полезно в системах контроля доступа, поскольку для авторизации доступа человеку не нужно вынимать карту из кармана или кошелька и вставлять ее в считыватель. Одной из областей применения такой технологии является общественный транспорт, в котором за короткое время требуется идентифицировать большое количество пассажиров. Другая интересная возможность в использовании бесконтактных карт связана с применением терминала с рабочей плоскостью считывания. В этом случае карта не вставляется в щель, а просто прикладывается к маркированной площадке на поверхности считывателя. Кроме простоты использования это решение привлекательно благодаря значительному снижению риска вандализма (из-за перегиба или застревания карты в щели считывателя).

Технология массового производства бесконтактных карт достигла уровня зрелости, когда высококачественные продукты для бесконтактной электронной идентификации доступны по ценам, которые незначительно отличаются от цен сопоставимых смарт-карт контактного типа.

6.2.2. Смарт- карта с микросхемой памяти

Самые первые смарт-карты были картами с микросхемой памяти, содержащей только энергонезависимую память и схемы, необходимые для чтения из памяти. Карты с микросхемой памяти используются для хранения информации в постоянном режиме или в режиме с перезаписью. Сегодня эти карты составляют наибольшую долю используемых смарт-карт. Такие карты относительно недороги и обеспечивают невысокую безопасность для различных применений: от карточек по оплате проезда до телефонных карточек. [85,140]

На рисунке 6.6 показана типичная архитектура смарт-карты с микросхемой памяти, схемой безопасности и контактным интерфейсом. На

этой схеме показаны не только основные потоки энергии и данных и не приводятся подробные схемы блоков.

Доступ к этой памяти контролируется блоком безопасности, который в простейшем случае включает защиту от записи или стирания всей памяти либо только некоторых ее областей. Данные передаются к карте и с карты через порт ввода/вывода (I/O). Чаще всего в картах памяти применяется синхронный механизм обмена между считывателем и картой. Существенно, что схемы смарт-карты выполняют в прямом синхронном режиме только команды низкого уровня, выдаваемые считывателем для указания адресов памяти с чтением по этим адресам или с записью в них. В стандарте ISO 7816 (Часть 3) определен специальный протокол синхронной передачи, который позволяет реализовать простую и экономичную ИС чипа.

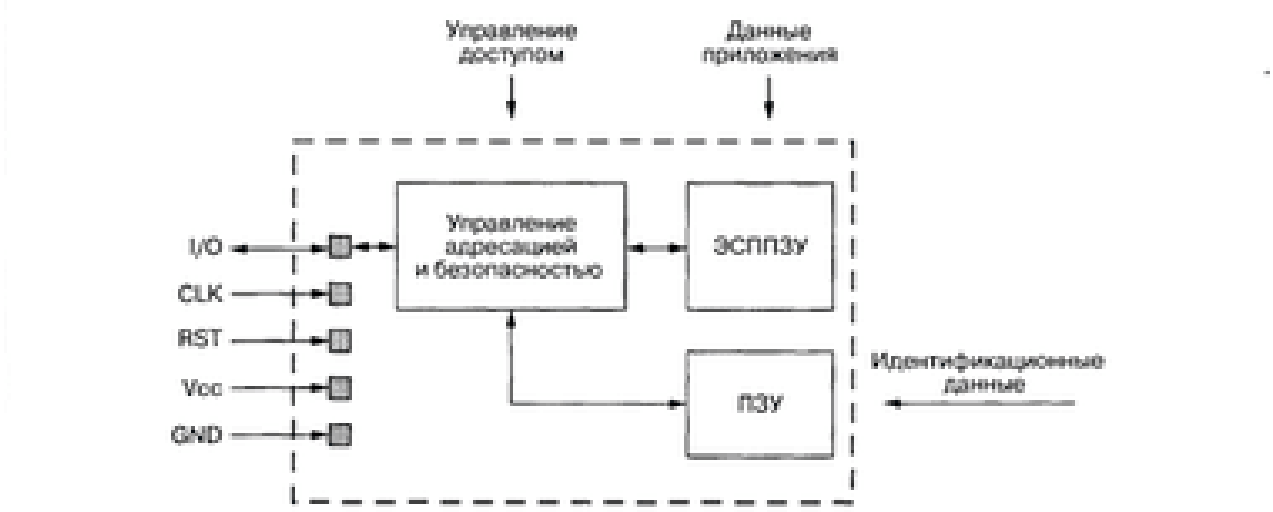


Рис. 6.6. Типичная архитектура смарт-карты с микросхемой памяти, схемой безопасности и контактным интерфейсом

Канал обмена находится всегда над прямым управлением со стороны считывателя и схемы карты выполняют в прямом синхронном режиме только команды низкого уровня, выдаваемые считывателем для указания адресов памяти с чтением по этим адресам или с записью в них. В стандарте ISO 7816 (часть 3) определен специальный протокол синхронной передачи, который позволяет реализовать простую и экономическую ИС чипа.

В некоторых новых картах с микросхемой памяти улучшена безопасность: используютя более сложные схемы безопасности, которые могут выполнять простое шифрование; введена схема адресации памяти, требующая наличия общего секрета между терминалом, производящим запись в чип карты, и самой картой.

На рис. 6.7 показана типичная архитектура карты с памятью со схемой безопасности и бесконтактным интерфейсом.

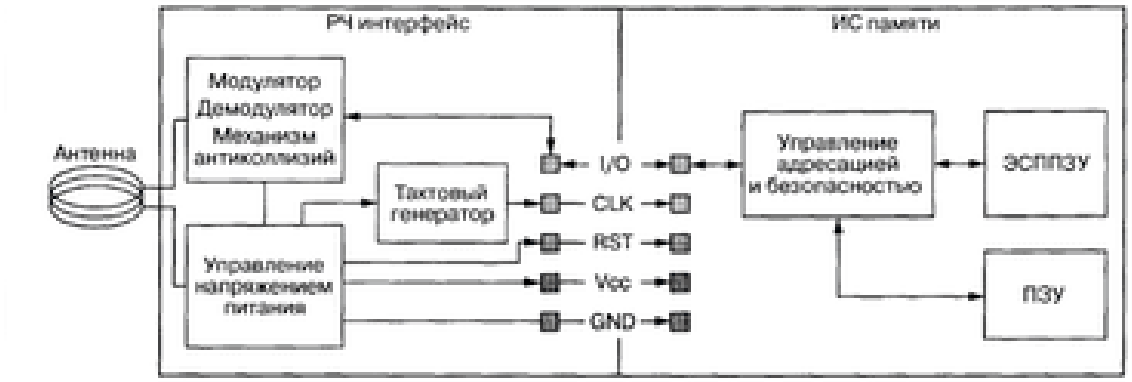


Рис. 6.7. Типичная архитектура смарт-карты с памятью со схемой безопасности и бесконтактным интерфейсом

Рис. 6.7. Типичная архитектура смарт-карты с памятью со схемой безопасности и бесконтактным интерфейсом

Смарт-карта с бесконтактным интерфейсом имеет встроенную микросхему, которая использует электромагнитный сигнал для обеспечения между картой и считывателем. В этих картах энергия, необходимая для работы чипа в карте, принимается на рабочей частоте считывателя. В бесконтактный радиочастотный интерфейс входят тактовый генератор, модулятор, демодулятор, механизм предотвращения коллизий и блок управления напряжением, состоящий из регулятора напряжения и генератора сброса в начальное положение. Применение бесконтактного РЧ интерфейса значительно облегчает использование карт по сравнению с контактным вариантом интерфейса, требующим ввода карт в щель считывателя.

При современном состоянии технологии скорость передачи данных между считывателем и бесконтактной картой ограничивается энергитическими возможностями карты. Для таких карт энергия для работы встроенной в карту микросхемы должна извлекаться из сигнала, передаваемого карте от считывателя. Механизм обратной связи от считывателя к карте, посредством которого производится верификация владельца карты, менее удобен в случае бесконтактной карты. В силу этого такие карты более популярны для применений, в которых обладание картой считается равнозначным разрешению на ее использование.

Функции карты с памятью обычно оптимизируются под конкретное приложение. Правда, при этом значительно ограничивается их гибкость, но цена остается умеренной. Карты с памятью обычно используются в качестве карт медицинского страхования, телефонных карт с предоплатой, карт для оплаты проезда в общественном транспорте и других областях.

6.2.3. Микропроцессорные смарт-схемы

Микропроцессорные смарт-карты все шире применяются в разнообразных приложениях. На рис. 6.8 показана типичная архитектура микропроцессорной карты контактного типа с сопроцессором [85,140]

Ядром чипа в микропроцессорной карте является центральный процессор (ЦП), который, как правило, окружен четырьмя дополнительными функциональными

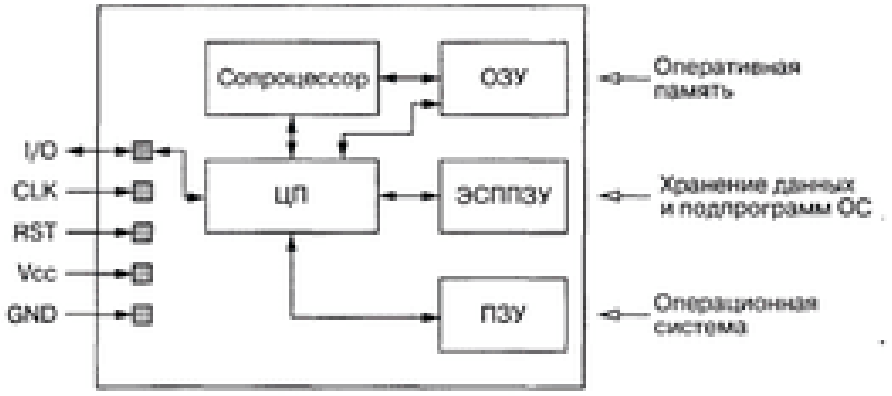


Рис. 6.8. типичная архитекутра микропроцессорной смарт-карты контактного типа с сопроцессором

Блоками: сопроцессор, ПЗУ (КОМ - read only memory), ЭСППЗУ (EEPROM - electrically erasable programmable read-only memory), ОЗУ (RAM - random access memory) и порт ввода/вывода. Сопроцессор предназначен для разгрузки ЦП от трудоемких операций криптозащиты смарт-карты. ПЗУ содержит операционную систему чипа, которая записывается в ПЗУ при изготовлении чипа. Таким образом, содержимое ПЗУ одинаково для всех чипов произведенной партии и оно не может быть изменено в течение всего срока службы чипа. ЭСППЗУ - это не разрушаемая память чипа. Данные и программный код могут быть записаны и считаны из ЭСППЗУ под управлением операционной системы. ОЗУ является оперативной памятью процессора. Эта память является энергозависимой, и все данные, хранимые в ней, теряются, когда выключается питание чипа. Последовательный интерфейс ввода/вывода обычно состоит только из одного регистра, через который передаются данные бит за битом.

Традиционными областями применения микропроцессорных смарт-карт с контактным интерфейсом являются платежные приложения (денежная карта, электронный кошелек) и мобильные телефоны (SIM-карта для мобильного телефона GSM), то есть приложения, которым требуется высокая степень безопасности при обработке и передаче данных. Связанная с этим необходимость иметь возможность быстро и просто вычислять сложные

криптографические алгоритмы привела к использованию в карточных чипах мощных криптографических сопроцессоров.

На рис. 6.9 показана типичная архитектура бесконтактной микропроцессорной карты с сопроцессором [85,140]. Часть интегральной микросхемы, содержащая блоки микроконтроллера, аналогична микросхеме микропроцессорной смарт-карты контактного типа. В качестве практического примера использования микропроцессорных смарт-карт можем привести введенный в эксплуатацию проект СКУД для одного из наших Заказчиков, который выполняет работы по ремонту бензогенераторов Honda и других мировых производителей, причем общая емкость внедренной системы составила более 600 пользователей БД. В бесконтактный радиочастотный интерфейс входят следующие блоки: тактовый генератор, модулятор, демодулятор, механизм предотвращения коллизий и блок управления напряжением, состоящий из регулятора напряжения и генератора сброса в начальное положение.

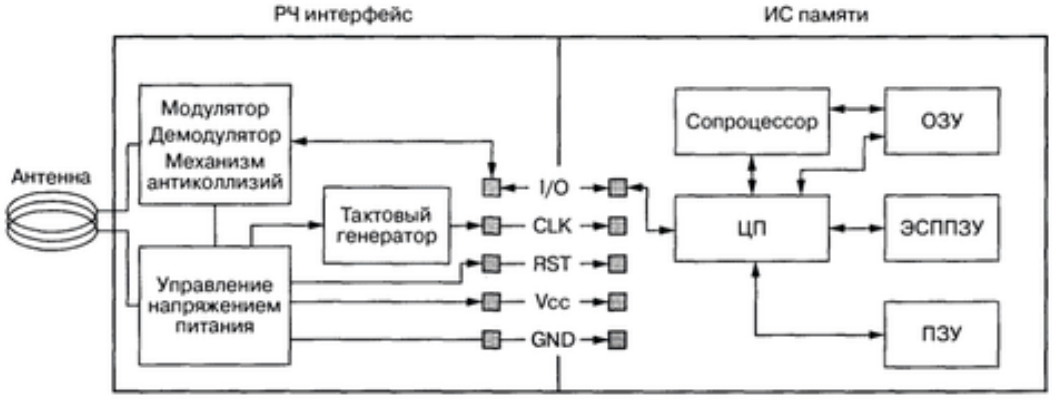


Рис. 6.9. Типичная архитектура микропроцессорной карты с сопроцессором и бесконтактным интерфейсом

Микропроцессорные карты отличаются высокой гибкостью в использовании. В простейшем случае они содержат программу, оптимизированную для единственного приложения, и соответственно могут быть использованы только для этого приложения. Однако современные операционные системы смарт-карты допускают размещение на одной карте нескольких различных приложений. В этом случае ПЗУ содержит только базовые команды операционной системы, в то время как связанная с: конкретным приложением часть программы загружается в ЭСП ПЗУ уже после изготовления карты.

Недавние разработки допускают загрузку прикладных программ в карту даже после того, как карта уже была персонализирована и передана держателю карты. Специальные исследования аппаратного и программного обеспечения подтверждают, что эта новая возможность не нарушает требования безопасности отдельных приложений. В настоящее время

разрабатываются специальные оптимизированные микропроцессорные чипы, обладающие высокой производительностью и большой емкостью памяти, - такие карты будут доступны в недалеком будущем.

6.2.4. Микропроцессорные смарт-карты с двойным интерфейсом

Как отмечалось выше, традиционными сферами применения микропроцессорных контактных смарт-карт являются платежные приложения (денежная карта, электронный кошелек). Бесконтактные микропроцессорные смарт-карты традиционно используются в приложениях, которые требуют дружественности по отношению к пользователю, например в системах контроля доступа, и короткого времени транзакций, например в электронных билетных системах оплаты проезда в общественном транспорте. Тенденция, направленная на комбинирование платежных приложений с типичными бесконтактными приложениями (смарт-карта с билетной функцией), в конечном счете привела к разработке смарт-карты с двойным интерфейсом, в которой на одном чипе доступны как контактный, так и бесконтактный интерфейсы. Соответственно, к карте с двойным интерфейсом можно обращаться как через контактный, так и через бесконтактный интерфейс.

Принцип действия, лежащий в основе смарт-карты с двойным интерфейсом, состоит в том, что интерфейс смарт-карты является полностью независимым от логики и программного обеспечения смарт-карты. Интерфейс, контактный или бесконтактный, полностью прозрачен для передаваемых прикладных данных, поэтому с точки зрения прикладного программного обеспечения тип используемого интерфейса является несущественным. Соответственно, интерфейс является заменяемым по желанию, и интерфейс, и логические компоненты могут комбинироваться как угодно.[98,140]

Типичная архитектура микропроцессорной карты с двойным интерфейсом показана на рис. 6.10. К каждому интерфейсу смарт-карты можно обращаться независимо от другого.

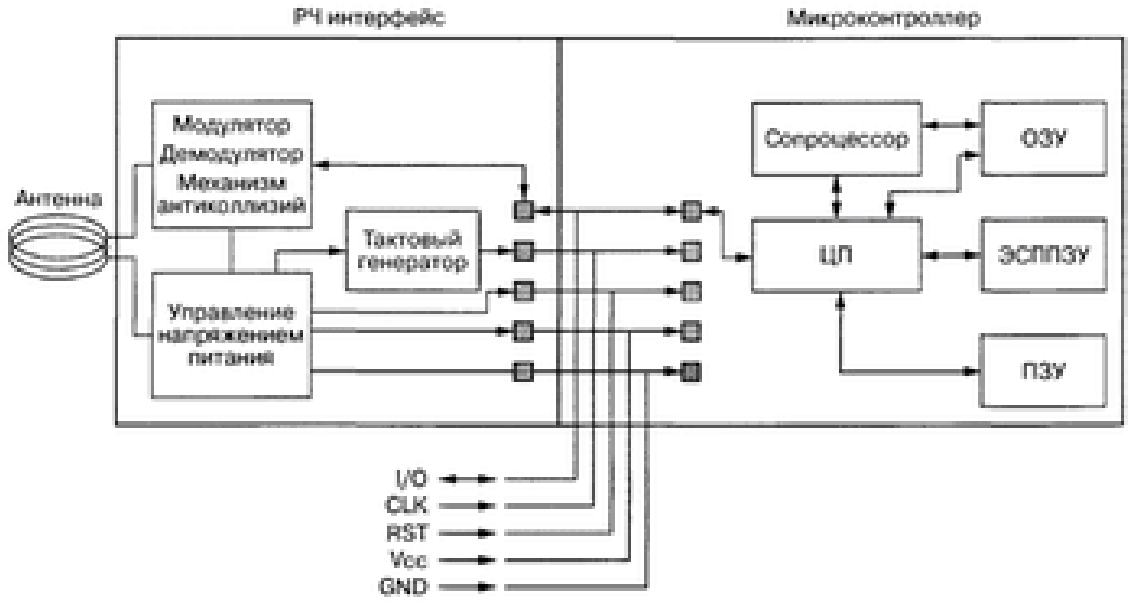


Рис. 6.10. типичная архитектура микропроцессорной карты с сопроцессором и двойным интерфейсом

Благодаря прозрачности интерфейса методы, определенные в стандарте ISO/IEC "7816. Безопасная передача сообщений", эффективно препятствуют повтору или подмене нуждающихся в защите передаваемых данных независимо от используемого интерфейса. Некоторые различия в подходах к обеспечению безопасности в контактных и бесконтактных смарт-картах анализируются в 13 главе.

Наибольшее различие между бесконтактными и контактными смарт-картами состоит в доступной мощности. В соответствии со стандартом ISO 14443 бесконтактной смарт-карте доступна мощность примерно 5 мВт для работы на максимальном расстоянии от считывателя (при H_{min}=1,5 А/м). Для контактной смарт-карты, в зависимости от ее спецификации доступна мощность до 300 мВт (ISO 7816-3 класс А - 5 В, 60 mA).

Это различие должно учитываться при разработке бесконтактных микропроцессорных чипов. Кроме того, во всех микросхемах с двойным интерфейсом используется низковольтная технология сверхнизкой мощности, чтобы доступная мощность могла расходоваться оптимальным образом. Нет необходимости в явном переключении бесконтактного и контактного режимов работы микросхемы. В простейшем случае достаточно использовать данные, принятые через один из интерфейсов, в качестве оценочного критерия для дальнейших действий. Некоторые микросхемы предоставляют программисту флажки состояния, которые позволяют запрашивать текущий активный режим работы.

Важным достоинством карты с двойным интерфейсом для пользователя и системного оператора является возможность использовать

существующую инфраструктуру (как правило, контактные считыватели) при вводе новых приложений.

6.3. Аппаратные компоненты смарт-карт

Как омечалось в разделах 6.1 и 6.2 в зависимости от типа встроенной интегральной микросхемы различают две группы смарт-карт:карты с интегральной схемой памяти и карты с интегральной схемой микроконтроллера. Карты с интегральной схемой памяти (карты с памятью) используются для хранения информации различных приложений. Память на таких типах карт может быть свободной для доступа или содержать схемы контроля доступа к памяти карты для ограничения и защиты операций чтения и записи данных. В качестве памяти обычно используется электрически стираемое программируемое постоянное запоминающее устройство ЭСППЗУ. В некоторых новых разработках карт с микросхемой памяти улучшена безопасность: используются более сложные схемы безопасности, которые могут выполнять простое шифрование; введена схема адресации памяти, требующая наличия общего секрета между терминалом, производящим запись в чип карты, и самой картой.

Карты с интегральной схемой микроконтроллера (микропроцессорные карты), в отличие от карт с памятью, содержат в себе микроконтроллер со специальной программой или небольшой операционной системой, которая позволяет преобразовывать данные по определенному алгоритму, осуществлять защиту хранящейся на карте информации при передаче, чтении и записи. Ядром чипа в микропроцессорной карте является центральный процессор, который, как правило, окружен дополнительными функциональными блоками: сопроцессор, ПЗУ, ЭСППЗУ, ОЗУ и порт ввода/вывода.

Память ЭСППЗУ входит в состав как ИС карты с памятью, так и ИС микроконтроллера микропроцессорной смарт-карты. Поэтому ЭСППЗУ целесообразно более подробно рассмотреть в составе функциональных блоков ИС микроконтроллера.

6.3.1. Микроконтроллеры смарт-карт

Центральным элементом микропроцессорной смарт-карты является микроконтроллер, встроенный в карту. В контактных смарт-картах чип с микроконтроллером располагается под контактной площадкой. Микроконтроллер инициирует, управляет и отслеживает все операции смарт-карты.

Различают два основных варианта архитектуры микроконтроллеров: классическую архитектуру фон-Неймана и гарвардскую архитектуру (рис. 6.11)

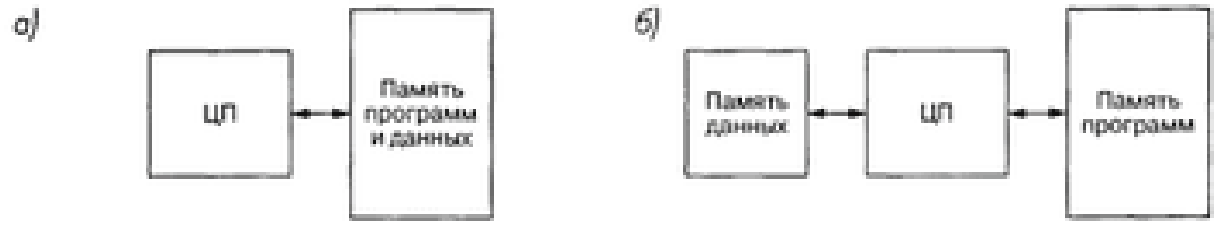


Рис. 6.11. Основные варианты архитектуры микроконтроллеров: а) архитектура фон-Неймана; б) гарвардская архитектура

Для архитектуры фон-Неймана характерен общий доступ к памяти программ и данных. При гарвардской архитектуре доступ к памяти данных и памяти про- граммы осуществляется раздельно. Большинство микроконтроллеров для смарт-карт строятся на основе гарвардской архитектуры.

Микроконтроллеры, разработанные для применения в смарт-картах, являются функционально полными компьютерами, то есть они содержат процессоры, несколько типов памяти и интерфейсы с внешней средой. (рис. 6.12)

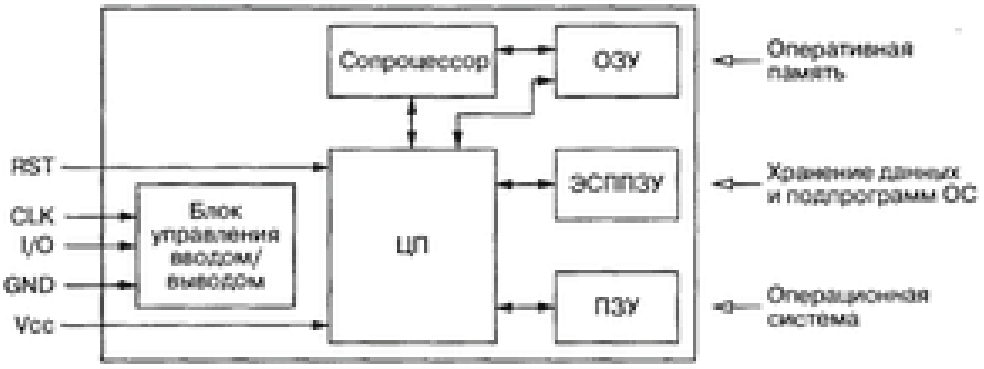


Рис. 6.12. Основные элементы ИС микроконтроллера смарт-карты

Наиболее важными функциональными элементами типичного микроконтроллера для смарт-карты являются ЦП (CPU), шина адресов/данных и три типа памяти: ОЗУ, ПЗУ и ЭСППЗУ. Кроме того, чип содержит простой интерфейсный модуль ввода/вывода, который отвечает за последовательную коммуникацию микроконтроллера с внешней средой. Ряд производителей предлагают чип микроконтроллера с дополнительными специализированными арифметическими блоками. Именно такие смарт-карты с дополнительными блоками были использованы нами при проектировании СКУД для складских помещений компании "Юстех Строй", которая выполняет работы по сносу домов и демонтажу зданий / сооружений, причем эти блоки действуют в качестве математического сопроцессора (NPU). Функции, предоставляемые таким блоком, ограничиваются операциями возведения в степень и модульными операциями над целыми числами. Обе эти операции являются базовыми частями таких процедур шифрования с открытым ключом, как алгоритм RSA и другие.

На рис. 6.13 показано типовое размещение элементов микроконтроллера смарт-карты на полупроводниковом кристалле.

Рис.6.13. типовое размещение функциональных элементов микрокотроллера смарт-карты на полупроводниковом кристалле

Следует отметить, что микроконтроллеры, используемые в смарт-картах, не относятся к числу электронных компонентов, выпускаемых для разных областей применения. Напротив, они являются компонентами, разрабатываемыми специально для применения в смарт-картах с учетом ряда специфических факторов (безопасность, функциональность, размеры чипа, себестоимость).

Безопасность

Смарт-карты предназначены для использования в приложениях, которые предъявляют высокие требования к безопасности хранения и обработки данных в смарт--карте. Общепризнанный и наиболее безопасный подход к интегральной схеме смарт-карты состоит в компоновке всех ее функциональных элементов (ЦП, СП, памяти, схем ввода/вывода и др.) на одном кристалле. Целью такого подхода к компоновке является обеспечение всех функциональных возможностей в малом физическом объеме и укрытие взаимных соединений между элементами микроконтроллера внутри чипа. Если бы микроконтроллер смарт-карты был выполнен на нескольких чипах, то соединения между этими чипами стали бы очевидными местами атак злоумышленников. Объединение элементов микроконтроллера в одном чипе затрудняет внешнему наблюдателю перехват сигналов, передаваемых между элементами, и соответственно распознавание информационного содержания этих сигналов, что повышает безопасность хранимых и обрабатываемых смарт-картой данных.

Функциональность

Из-за необходимости интегрировать все функциональные элементы микроконтроллера в одном кремниевом чипе с ограниченными размерами

возможное количество размещенных на чипе полупроводниковых компонентов очень ограничено. С учетом требований функциональной полноты, минимальной площади чипа и реализации последовательного интерфейса на чипе располагают только те элементы, без которых невозможно обеспечить нормальное функционирование смарт-карты.

Разметы чипа

Размер чипа с интегральной схемой существенно влияет на его хрупкость. Чем больше площадь чипа, тем легче его переломить, когда карта с чипом подвергается сгибанию. Например, смарт-карта с чипом, переносимая в бумажнике, подвержена большим случайным деформациям. Соответственно получаются очень большими напряжения изгиба карты и чипа, вмонтированного в эту карту. Тончайшей трещины в чипе вполне достаточно для того, чтобы сделать чип бесполезным.

Поэтому большинство производителей карт придерживаются верхнего предела площади чипа, равного примерно 25 мм2, причем для минимизации риска перелома чипа его компоновку стремятся приблизить по возможности к квадратной.

Себестоимоть

Площадь, занимаемая интегральной схемой на поверхности кремниевого кристалла, является одним из решающих факторов, определяющих себестоимость чипа. Для сложных и соответственно более дорогих модулей производят чипы, занимающие большую площадь. Многие микроконтроллеры, доступные на рынке, включают функции, которые не нужны в смарт-картах. Поскольку реализация этих функций требует дополнительного пространства на кристалле, они могут быть удалены из чипов, предназначенных для смарт-карт. Хотя такие попытки минимизировать размер чипа лишь немного уменьшают себестоимость одного чипа, однако небольшая экономия на одном чипе дает серьезную прибыль при производстве большого их количества.

6.3.2. Центральный процессор смарт-карты

К процессору смарт-карты предъявляется требование высокой надежности. Поэтому в смарт-картах обычно используются процессоры, которые прошли длительную апробацию в других областях. Производители смарт-карт предпочитают доверять более старым процессорам, которые были проверены на практике, а не экспериментировать с последними разработками производителей полупроводниковых микросхем. Аналогичного подхода придерживается авиакосмическая промышленность, для которой жизненно необходима функциональная безопасность используемых компонентов.

Обычно размер адресуемой памяти в смарт-карте находится между 6 и 30 Кб, поэтому использование 8-битовой шины памяти не накладывает каких-либо значительных ограничений. Сами процессоры основаны на архитектурах RISC (reduced instruction set computer - компьютер с урезанным набором команд) и CISC (complex instruction set computer - компьютер с комплексным набором команд).

Концепция RISC-архитектуры заключается в использовании процессора с сокращенным набором команд, которые выполнялись бы в идеале за один машинный такт. Сокращение набора команд обусловлено простым соображением: чем короче команда, тем быстрее она считывается и выполняется. По техническим характеристикам современные 8-разрядные RISC-микроконтроллеры занимают промежуточную нишу между классическими 8-разрядными и 16-разрядными МК. Они обладают высокой производительностью и меньшей, чем у 16-разрядных МК, ценой. Архитектура CISC обычно имеет большие наборы команд и требует несколько тактовых циклов на машинную команду. Диапазон адресов 8-битовых процессоров чаще всего составляет 16 бит, при котором может быть адресовано максимум 65536 байт.

Наборы команд процессора основаны на архитектуре либо Motorola 6805, либо Intel 8051. Дополнительные команды могут быть добавлены к стандартному набору команд производителем чипа. Эти команды чаще всего связаны с дополнительными опциями для адресации 16-битовой памяти.

Новые процессоры для высококачественных смарт-карт переходят на 32-битовую архитектуру. При использовании мощного процессора может быть достигнута приемлемая скорость исполнения программного кода, в частности такого программно-интерпретируемого кода, как текущие реализации Java. Первый шаг в 32-битовую архитектуру был сделан в 1993 году Европейским проектом CASCADE (Chip Architecture for Smart Card and portable intelligent Devices). Одной из целей этого проекта является создание высокопроизводительного процессора для смарт-карт. В этом проекте был выбран RISC-процессор ARM 7M. Он имеет 32-битовую архитектуру, может работать на частоте до 20 МГц при напряжении питания 3 В и при этих условиях потребляет только 40 мА. При 0,8-микронной технологии ядро ARM 7 занимает площадь 5,9 мм^2 (3,12*1,9 мм), а связанный с ним арифметический процессор для криптографических алгоритмов с открытым ключом (RSA, DSA, EC) занимает дополнительно 2мм^2. Этот процессор имеет как супервизорный, так и пользовательский режимы и соответственно поддерживает разделение кода операционной системы и кода приложения [85,140]

По сравнению с 8-битовыми процессорами 32-битовые процессоры занимают значительно больше места на кристалле из-за более широких шин и сложной внутренней организации, однако в новых приложениях смарт-карт они будут

использоваться в возрастающих количествах. Предлагаемая ими вычислительная мощность необходима для этих приложений, поэтому такие недостатки, как более высокое энергопотребление и увеличенная площадь чипа, являются неизбежными издержками технического прогресса. Естественно, 8-битовые процессоры не исчезнут в обозримом будущем, поскольку они обеспечивают надежную базу для недорогих чипов.

6.3.3. Сопроцессор

Ограниченные ресурсы центрального процессора не могут удовлетворить потребности смарт-карты в надежном шифровании и аутентификации за приемлемое время без ущерба для общей функциональности смарт-карты. Типичная для смарт-карты транзакция должна занимать от 1до 3 с, а шифрование с использованием 1024-битового ключа RSA в процессоре, типичном для смарт-карты, может занять 10 секунд и более. В результате этого в некоторые микроконтроллеры смарт-карт включают сопроцессоры для ускорения вычислений, проводимых при шифровании и аутентификации. Сопроцессор представляет собой специально разработанный арифметический блок для вычислений криптографических алгоритмов. Сопроцессор интегрируется на кремниевой подложке вместе с другими функциональными блоками микроконтроллера.

Сопроцессор может выполнять несколько базовых вычислений целочисленной арифметики, которые необходимы для алгоритмов криптографии с открытым ключом, включая быстрое умножение, возведение в степень и модульные операции больших целых чисел. Быстродействие сопроцессора оптимизируется исключительно для этих операций, поэтому в своей области применения он может превзойти даже очень быстрые ПК по меньшей мере в 6 раз. Использование сопроцессора значительно ускоряет криптографические операции с открытым ключом, однако при этом несколько увеличивается общий размер чипа, а также возрастает его себестоимость. В зависимости от модели микроконтроллера блок шифрования DES и генератор случайных чисел могут либо входить в сопроцессор, либо выполняться в виде самостоятельных узлов микроконтроллера.

Блок шифрования DES

В ряд микроконтроллеров для смарт-карт включается дополнительный блок шифрования DES (Data Encryption Standard). Алгоритм шифрования DES применяется для защиты финансовых транзакций и телекоммуникационных приложений. Алгоритм DES изначально разрабатывался для аппаратной реализации, поэтому включение блока DES в микроконтроллер смарт-карты не является сложной задачей. При тактовой частоте 3,5 МГц выполнение DES занимает всего 154 мкс, а для вычисления тройного DES требуется 236 мкс. Повышение тактовой частоты приводит к соответствующему

уменьшению времени вычисления. Криптографические методы и алгоритмы защиты смарт-карт подробно рассматриваются в главах 8-11 и 13.

Генератор случайных чисел

Случайные числа необходимы для генерации крипто-ключей с целью аутентификации смарт-карты и терминала. По причинам безопасности они должны быть не псевдослучайными, а истинно случайными числами. Существуют микроконтроллеры, которые имеют интегрированы е генераторы случайных чисел для создания действительно случайных чисел. Качество работы подобных генераторов не должно зависеть от таких внешних физических факторов, как изменения температуры и напряжения питания.

Поскольку этого трудно добиться в чипе на одном кремнии, используется другой подход. Генератор случайных чисел использует разные логические состояния процессора, такие как содержимое регистров, и подает их на циклический регистр сдвига с обратными связями, который приводится в действие тактовыми импульсами, генерируемыми на основе ряда различных параметров. Если процессор считает содержимое этого регистра, он получает сравнительно качественное случайное число, которое невозможно детерминировано вычислить вне чипа. Качество получаемых случайных чисел можно улучшить, используя дополнительные процедуры и алгоритмы.

Аппаратная поддержка управления памятью

Последние версии операционных систем смарт-карт допускают загрузку в карту исполняемого машинного кода. Затем его можно вызвать, используя специальную команду, и, например, исполнить криптографическую функцию, известную только эмитенту карты. Однако существует следующая серьезная опасность: после того как исполняемая программа загружена в карту, в принципе, она имеет возможность, используя соответствующую подпрограмму, произвести считывание секретных данных из памяти карты. Открытый доступ к такой конфиденциальной информации, как секретные ключи и алгоритмы в различных приложениях на карте, имел бы фатальные последствия для пользователей.

Приемлемым решением этой проблемы является включение в микроконтроллер смарт-карты *блока управления памятью MMU* (Memory Management Unit). Этот блок контролирует границы памяти текущего приложения параллельно с исполнением программы. Разрешенная область памяти фиксируется подпрограммой операционной системы перед вызовом этого приложения, и она не может быть изменена прикладной программой во время исполнения этой программы. Такой подход гарантирует, что приложение полностью инкапсулировано и не получит доступ в области памяти, запрещенные для него.

Пока лишь немногие микроконтроллеры смарт-карт имеют блоки управления памятью MMU. Следует отметить, что в будущем значимость этого дополнительного аппаратного обеспечения сильно возрастет, поскольку это – единственный практический способ безопасно разделить несколько приложений, которые поддерживаются одной смарт-картой.

6.3.5. Канал ввода/вывода смарт-карт

Для осуществления трафика ввода/вывода между картой и считывателем применяются две линии интерфейса. По одной из них, линии ввода/вывода, передаются биты данных. Эта линия может находиться в одном из двух состояний, одно из которых представляет 0, а другое 1. Вторая линия, тактовая, указывает, когда производить выборку в линии ввода/вывода для получения бита данных.

Канал ввода/вывода смарт-карты является последовательным каналом, работающим в полудуплексном режиме. Это означает, что он передает данные побитно, порциям и по 1 байту за один раз, причем поток данных может идти в любой момент времени только в одном направлении. Если и терминал, и карта будут передавать данные одновременно, то данные будут потеряны. Если они одновременно перейдут в режим приема, то система перейдет в состояние взаимной блокировки. Поэтому действия смарт-карты и терминала должны быть синхронизированы.

Каждая сторона канала связи должна отслеживать, находится ли партнер по обмену в состоянии передачи или в состоянии приема. Если в результате обмена ошибочными сообщениями возникает неопределенная ситуация, тогда на считыватель возлагается ответственность за перезапуск всей последовательности протокола обмена для уст ранения сбоя в канале.

Протокол связи между хост-компьютером и смарт-картой поддерживает отношения ведущего (хост-компьютер) и ведомого (смарт-карта). Хост-компьютер посылает команды карте и слушает ответ. Смарт-карта никогда не посылает данные хост-компьютеру, кроме как в ответ на его команду. Операционные системы смарт-карт обычно поддерживают как посимвольный, так и поблочный обмен.

Аппаратная часть смарт-карты может обрабатывать данные со скоростью 115200 бит/с, но большинство терминалов для смарт-карт обычно поддерживают связь с картами на скоростях значительно ниже этой, в частности управляют контактными смарт-картами на скорости 9600 бит/с, а бесконтактными смарт-картами - на скорости 7800 бит/с.

Продвижение к более совершенным механизмам ввода/вывода является одним из направлений развития технологии смарт-карт. Некоторые новые смарт-карты допускают прямое использование USB-канала. В интерфейсе USB используются две дополнительные линии, образующие второй канал

ввода/вывода. Этим достигается дуплексное соединение. USB-канал работает в дуплексном режиме с повышенной скоростью. При повышенных скоростях могут работать и существующие архитектуры операционных систем смарт-карт.

Важным шагом в совершенствовании канала ввода/вывода смарт-карты является введение аппаратной поддержки передачи данных. До настоящего времени прием и передача данных через интерфейс смарт-карты контролировались исключительно программными средствами операционной системы без какой-либо аппаратной поддержки. Это сильно усложняет программное обеспечение, а также увеличивает возможность программных ошибок. Однако основной проблемой является ограничение скорости программно поддерживаемой передачи данных, поскольку сама скорость работы процессора жестко ограничена.

Для разгрузки процессора и получения более высоких скоростей коммуникации разработан универсальный асинхронный приемо-передающий блок UART (Universal Asynchronous Receiver Transmitter). Этот блок дает возможность выполнять прием и передачу данных без прямого вовлечения процессора в эти действия. Он не ограничивается производительностью процессора и не нуждается в ПО для коммуникации на байтовом уровне. Конечно, более высокие уровни протокола передачи данных должны быть представлены в смарт-карте как программное обеспечение, но самый низкий уровень реализуется как программное обеспечение в UART.

В настоящее время лишь ограниченное количество микроконтроллеров обеспечивают с помощь UART аппаратную поддержку коммуникаций. В будущих микроконтроллерах смарт-карт применение UART, вероятно, станет стандартом.

1.3. Методы и технологий считывания и обработки данных в микропроцессорных картах

Основные системы в радиочастотной идентификации

На сегоднящний день все часще используют радиочастотную идентификацию RFID. Для идентификации объека на нем заклеплены специальные радиочастотные метки (транспондеры), несущая идентификацию и разные информации. Основаня техническая процедура позаимствована из радиолакационной системы. Метод радиолакационной идентификации назвали RFID-технологией и стал основой построения бесконтакных информационных систем., а системы реализующие такую идентификацию, назвали RFID-системами.

Принцип действия похожи к бесконтакным микропроцессорным картам. Хранятся данные в транспондере - носитель электронных данных, также как и в бесконтакных микропроцессорных картах, питание подается к

устройству- носителю данных, а также обмен данных между устройством-носителем и считывателем происходит ез гальванических контактов, вместо него магнитное или электромагнитное поле.

RFID-системы и их состав

Основная схема радиочастотной идентификации на рис.

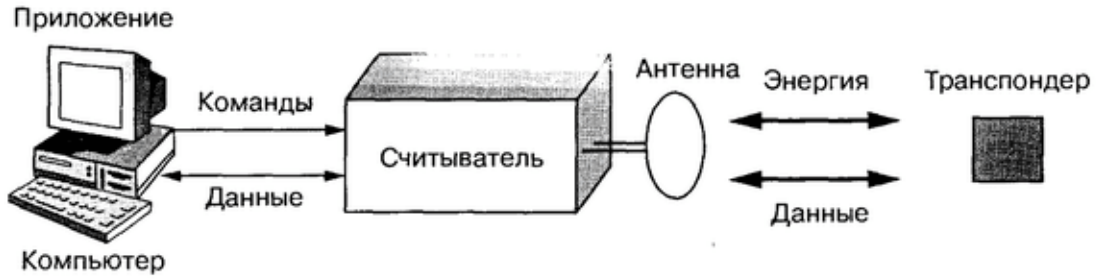


Рис. 1.4. Общая схема системы радиочастотной идентификации RFID

RFID состоит из следующих компонентов:

- считывателя;
- транспондера (радиочастотная метка, тег);
- компьютерная система ораотки данных;

Транспондер и считыватель взаимодействуют между собой с помощью радиочастотным каналом.

В составе считывателя входят:

- приемо-передающее устройство, а также антенну, который посылает и принимает сигнал к транспондеру;
- микропроцессор проверяющий и декодирующий данные;
- память которая сохраняет данные, для следующей передачи;

Компоненты транспондера:

- Чип (интегральная схема), которая управляет связью со считывателем;
- Антенна.

Через антенну передатчик считывателя излучает определенной частоты электромагнитное поле. В зоне действия транспондер обнаруживает сигнал от считывателя и отвечает собственным сигналом на той или другой частоте, содержащей информацию (код товара и т.д.). Антенной считывателя улавливается сигнал, нужная информация расшифровывается и для обработки передается в компьютер.

Не требуется непосредственного контакта между считывателем и транспондером, так как сигнал с легкостью проходит через неметаллические

материалы. Поэтому, транспондеры могут быть внутри тех объектов, которые идентифицируются.

Транспондеры: Активные и пассивные

Активные работают от встроенной или соединенной батареи, нужно меньшей мощности считывателя, но с большой дальностью считывания.

Пассивные не имеющие собственного источника питания. Нужную энергию получают от электромагнитного сигнала, которая поступает от считывателя, а дальность считывания зависит от энергии считывателя.

Система идентификации на микропроцессорных картах.

Микропроцессорная карта - пластиковая карта со встроенным микропроцессором, выполняющие контроль доступа к памяти. Важность то что она не только хранить, а также и обрабатывает содержащую информацию. Содержащее в микросхеме надежно защищено от постороннего доступа.

Классификация микропроцессорной карты можно по следующим признакам:

- тип микросхемы;
- способ считывания информации;
- соответствовать стандартам;
- в области применения.

Микропроцессорные карты делятся на два типа в зависимости от встроенной микросхемы:

- Карты с памятью для хранения информации. Память там свободна для доступа или содержать логику контроля доступа к памяти карты для ограниченных операции чтения и записи на ней данных, а также защищаться PIN-кодом.
- смарт-карты для задач требующие сложной обработки информации. в ней микроконтроллер, центральный процессор соединенный с сопроцессором, оперативное запоминающее устройство (ОЗУ), постоянное запоминающее устройство (ПЗУ) и электрически стираемое программируемым ПЗУ (ЭСППЗУ)- на рис.

В составе микроконтроллера микропроцессорных карт входят:

- центральный микропроцессор с тактовой частотой до 10 МГц;
- (RAM-Random Access Memory) оперативное ЗУ емкость до 2 Кбайт. Память для временного хранения данных, к примеру вычисления, хранимые данные в ОЗУ исчезают при выключении напряжения питания.

- постоянное ЗУ (ROM-Read Only Memory) емкость до 32 Кбайт; в ПЗУ пишется набор программ, являющийся операционной системой микропроцессорной карты;
- Электрический стираемое программируемое постоянное запоминающее устройство-ЭСППЗУ(EEPROM-Electrically Erasable Prom) емкость до 32 Кбайт. Многократно перезаписывается и считывается в ЗУ, а данные при выключении питания не пропадают;
- схема ввода/вывода (I/O). Для обмена данных внешних устройств.
- Встроенная система безопасности для защиты данных хранится и обрабатывается в микропроцессорной карте, в виде криптографического сопроцессора с функцией криптографического преобразования данных.

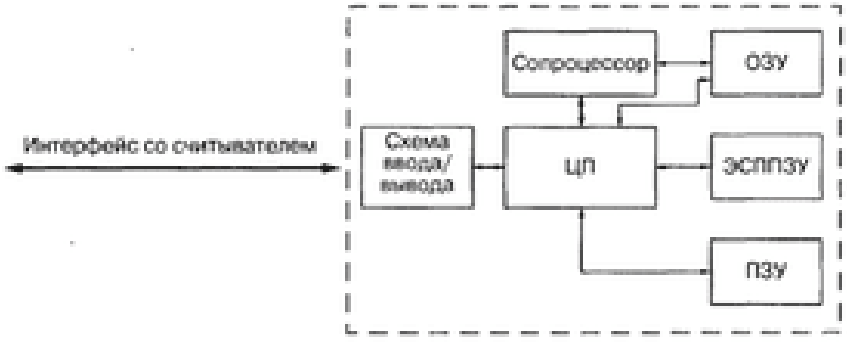


Рис. 1.5. Блок-схема микроконтроллера, встриваемого в микропроцессорную смарт-карту

Операционная система карты COS (Card Operation System) в ПЗУ записан специальный набор программ. Информация в ПЗУ записано при выпуске микропроцессорной карты. В операционной системе есть файловая система на базе ЭСППЗУ обеспечивая доступ к данным и часть данных бывает доступна внешним программам карты, а чтение и запись памяти ЭСППЗУ контролирует операционная система.

В настоящее время в одной смарт-карте бывает интеграция разных приложении (мульти-приложение), а некоторые приложения загружаются в ЭСППЗУ до выпуска карты и инициируются через операционную систему.

С опцией программирования смарт-карт способна к быстрой адаптацией к вновь выпущенным приложениям.

Интеллектуальные карты защищают хранимые на ней информацию при передаче, чтении и записи.

Бывают разновидность интеллектуальных карт-карты с криптографической логикой. Они применяются в системах защиты информациипри процессе шифрования данных или выработке криптографических ключей, электронных цифровых подписей.

Классификация микропроцессорных карт и способы считывания информации с них.

По таким признакам существуют такие типы микропроцессорных карт:

- контактные;
- бесконтактные;
- с двойным интерфейсом.

Контактная состоит из:

- Микроконтроллер каты (чип с интегральной схемой);
- Пластиковая основа;
- контактная область;

Контактная область 6 или 8 контактов квадратной или овальной формы. Размеры и основа карты и расположения контактов определены стандартами ISO-7816.

Взаимодействуют со считывателем физического прикосновения металлических контактов с контактами считывателя. Микропроцессорные карты получают от считывателя через контактные поверхности энергию питания и тактовые импульсы и передает считывателю после аутентификации держателя и терминал спрашивает информацию. Передача данных происходит через двунаправленный последовательный интерфейс (I/O-порт) между считывателем и картой. Метод считывания простой, но быстро изнашиваются контакты карты. Недостаток микропроцессорных карт с контактами ведет к износу, коррозии и грязи, а считыватели дорогие и неправильной работе. Такие считыватели доступны для всех, к примеры в телефонных аппаратах, но не защищены от вандализма.

Бесконтактные микропроцессорные карты самые перспективные на сегодняшний день в ее состав входят :

- встроенный в корпус индуктивная антенна;
- Чип с интегральной схемой;

Для большей защиты чип с интегральной схемой помещается в внутренний модель, который подключен к концам антенны. На рис. показаны конструктивные элементы микропроцессорной карты бесконтактного интерфейса и архитектура с чипом (интегральной схемой).

Встроенная интегральная схема состоит:

- бесконтактного радиочастотного интерфейса (РЧ);
- микроконтроллера;

Схема РЧ интерфейса соединяются с выводами антенны микропроцессорной карты и используют переменное электромагнитное поле, излучаемое считывателем для получения энергии питания для микропроцессорных карт и обмен данных между считывателем и картой.

Считыватель генерируя электромагнитное излучение какой-либо частоты и при поднесении карты в зону действия считывателя это излучение через встроенную антенну в карте и РЧ интерфейса запитывает микросхему карты.

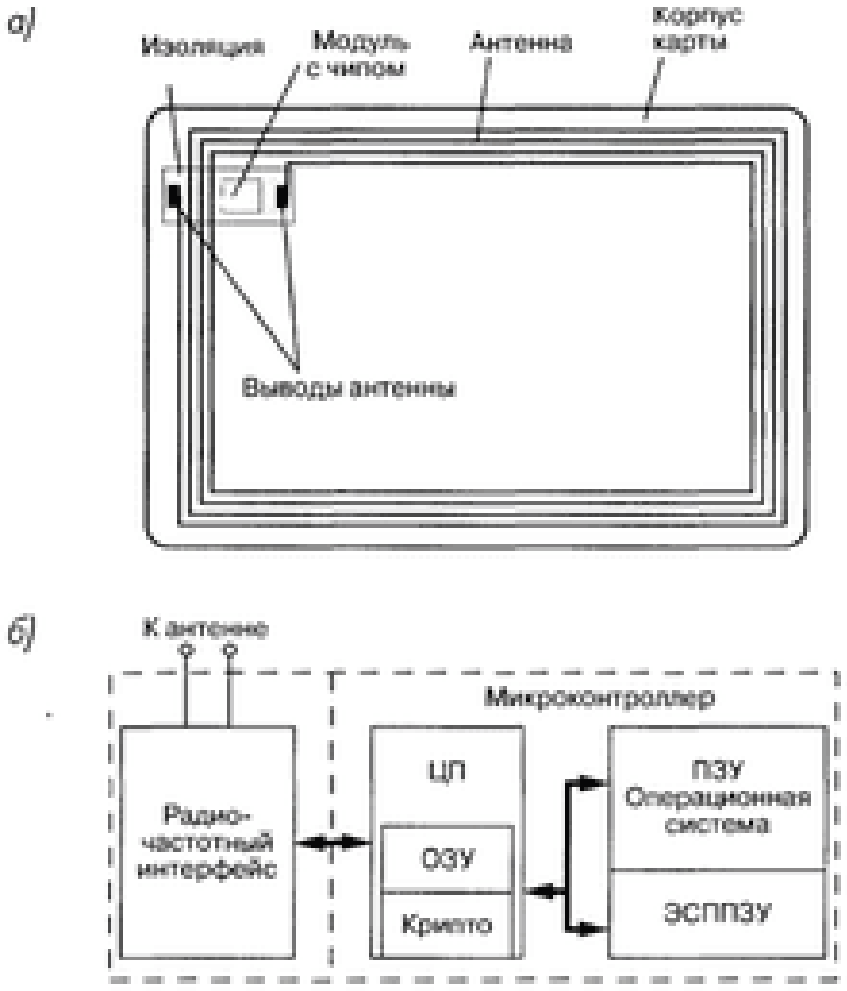


Рис. 1.6. Микропроцессорная смарт-карта с бесконтактным интерфейсом: а) конструктивные элементы бесконтактной смарт-карты; б) архитекртура интегральной схемы (чипа) бесконтактной смарт-карты

Нужную энергию для работы, карта перекидывает на считыватель идентификационный свой номер с помощью электромагнитных импульсов какой-либо частоты и формы.

Они срабатывают на расстоянии от 10 см до 1 метра, зависит от рабочей частоты считывателя и без какого либо позиционирования, обеспечивая устойчивую работу с высоким пропускной способностью и удобна в использовании. Чтобы бесконтактная карта сработала все лишь нужно ее преподнести ее к считывателю.

Также используется термин Hand Free (руки свободны), оставлять в кармане, в бумажнике держателя карты. Считыватель может быть прикреплен на стене или обратной стороны двери повышая уровень защиты от вандализма, чтобы повысить уровень защиты используют PIN код (Personal Identification Number), который известен только держателю карты, чтобы набрать код устанавливают вместе со считывателем PIN-кодовую панель (клавиатуру), но применение кода в качестве основного идентификатора не рекомендуется, такой метод считается низким уровнем секретности.

Карты с двойным интерфейсом в них есть:

- контактная площадка;
- встроенная катушка индуктивности.

Такого типа карты могут осуществлять работу с разными считывателями.

Считыватели

Считыватели системы RFID выполняют роль интерфейса для приложения, которая требует считывание или запись данных на бесконтактный мобильный транспондер, но доступ к транспондеру должен быть аналогичен процессу доступа к другим сопоставимым носителям данных, к примеру микропроцессорным картам с контактным интерфейсом. Взаимосвязь между звеньями информационной цепочки "приложение-считыватель-транспондер" (Рис.)основан на принципе "ведущий-ведомый" (master-slave), что означают все действия считывателя и транспондера инициируют приложения. В иерархической структуре системе приложения являются ведущими, а считыватель-ведомый, активизируется при поступлении команды записи/чтения.

Для того чтобы выполнить команду приложения, считыватель создает соединение с транспондером, но считыватель ведущий по отношению к транспондеру, а транспондер только отвечает на команды считывателя не проявляя активность (за ислючением простых транспондеров "только чтение"). Простая команда считывания, поступившм.я на считыватель от приложения может инициировать рад шагов по обмену сообщениями считывателем и транспондеров, к примеру команда считывания поступает от приложения, вызывает активацию считывателем трансподера, а в следом исполнение транспондером и считывателем аутентифиционной процедуры, после осуществить передачу транспондером считывателю требуемых данных.

Все этапы бесконтакного обмена данными между считывателем и транспондером, соединение, процедуры антиколлизии и аутентификации, передача данных от транспондера, полностью управляется считывателем.

Не зависит отрзличий в диапазоне частот, тип связи (индуктивная/электормагнитная) коммуникационной последовательности особенности процедуры передачи данныхот транспондера к считывателю (нагрузочная модуляция, обратное отражение, применеие субгармоник), но все считывали имеют много общего в принципе действия и в конструкции.

Во всех системах RFID структуру считывателя можно представить в следующих функциональных блоках:

- радичастотного интерфейса (РЧ), состоящего из передатчика и приложения;
- блок управления (рис.).

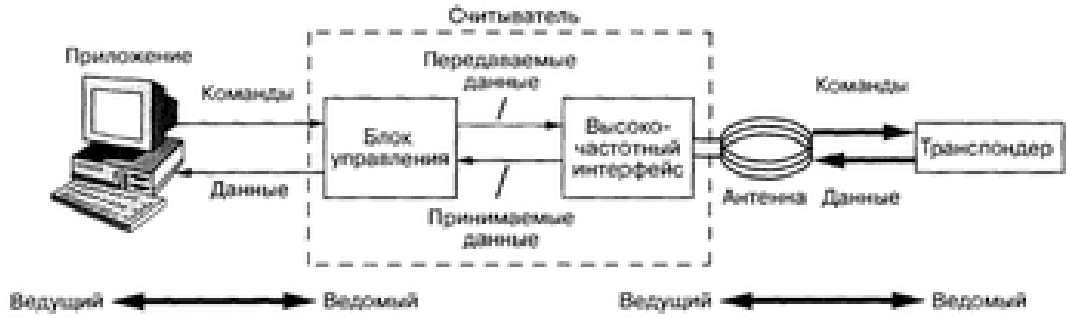


Рис. 4.1. Схема вхаимодействия приложения, считывателя и транспондера

Блок управления считывателя

Блок управления выполняет:

- Связь с ПО приложениями и исполнение команд ПО приложения;
- Управлять связь с транспондером (принцип "ведущий-ведомый" master-slave);
- кодирование и декодирование сигнала.

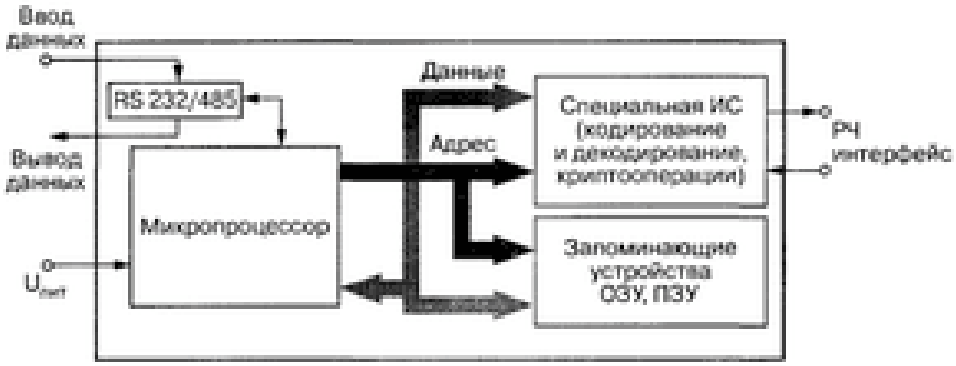


Рис. 4.5. Схема юлока управления считывателя

В более сложных системах доступны следующие функции:

- исполнение антиколлизионного алгоритма;

- шифрование и расшифрование данных, передаваемых между транспондером и считывателем;
- выполнение аутентификации между транспондером и считывателем;

Выполнение таких сложных функции блок управления базируется на микропроцессоре. Такие криптографические процедуры, поточное шифрованиемежду транспондером и считывателем, кодированиесигнала выполняется в специализированной ИС (ASIC-Applications Specific Integrated Circuit)чтобы не загружать процессор сложными вычислениями, а доступ к ИС через шину микропрцессора.

Обмен даннх между блоком управления через интерфейс RS232 или RS485. 8-битное асинхронное NRZ-кодирование, для обычного ПК. Скорость двоичной передачи 1200 бод (4800 бод, 9600 бод). Для протокола связи используются различные, часто самоопределенные протоколы.

Интерфейс между блоком управления и РЧ интерфейсом предостовляется как двоичное число. При ASK-модулированой системе логическая 1 при входе модуляции РЧ интерфейса предостовляет состояние "РЧ сигнал включен", а логическая 0 представляет "РЧ сигнал выключен"

Практическая реализация считывателей.

Приложения которые используют бесконтакные идентификационные системы, где требуется несколько считывателей и большое количество транспондеров, к примеру в системе общественного транспорта применяют несколько десятков тысяч бесконтакных микропроцессорных карт, а в транспорных средствах устанавливаются несколько сотен считывателей. Для идентификации животных, контейнеров имеется разница между количеством применяемых транспондеров и числом считывателей. Существуют много разных систем, так как нет еще стандартов для индуктивных или микроволновых систем RFID. Считыватели производятся только неольшими партиями-по несколько тысяч.

Теперь возможна интеграция всей аналоговой секции считывателей в кремниевую интегральную микросхему, причем требуется несколько внешних компонентов.

Платежная EMV-карта. Механизмы обеспечения безопасности платежа из песочницы

Платежные системы*, Информационная безопасность*

Платежные карты прочно вошли в нашу жизнь. Еще совсем недавно повсеместно использовались только карты с магнитной полосой. Сегодня же никого не удивишь картой с чипом. Всем известно, что чиповая, микропроцессорная или, созвучнее, платежная EMV-карта – современный и надежный способ доступа к расчетному счету. Она безопаснее карты с магнитной полосой и ее практически невозможно подделать. Однако детали реализации «внутренностей» EMV-карты мало известны. Всем кому интересно как работает EMV-карта, почему технология EMV обеспечивает безопасность платежей и насколько стоит всему этому доверять – добро пожаловать под кат.

1. Введение

О каких картах пойдет речь?

Сегодня международные платежные системы (МПС) используют стандарт EMV для проведения операций по банковским картам. Одними из наиболее известных МПС, стоящих у истоков разработки этой технологии, являются компании «VISA Inc» и «MasterCard Worldwide». Поскольку в основе микропроцессорных карт этих компаний лежит общая технология EMV, мы будем рассматривать обобщенную EMV-карту, не вдаваясь в детали реализации той или иной компании.
Стоит сразу отметить, что спецификация EMV достаточно большая, поэтому статья не претендует на полное описание стандарта. Многие вещи будут представлены в упрощенной форме без использования специфической терминологии. Так как стандарт является открытым, при желании всегда можно ознакомиться и разобраться в деталях на сайте EMVCo.

Описывая платежные транзакции и функциональность EMV-карты, мы будем ссылаться на других участников системы. Помимо самой платежной системы в процессе проведения транзакции участвуют:

- банк-эмитент – банк, который выпустил платежную карту, и счет которой находится в этом банке

- банк-эквайер – банк, который обслуживает терминал платежной точки
- платежный терминал – устройство, которое обеспечивает работу с платежной картой

Рассматривая подробнее платежную EMV-карту, мы будем концентрировать внимание не только на возможностях микропроцессора. Технология EMV подвергла изменениям как сами карты, так и сообщения, которыми обмениваются участники системы; расширила функциональность приложений для терминалов, банков-эквайров и эмитентов.

2. Аутентификация магнитной и EMV-карты

Одна из основных задач банка, выпустившего карту – это аутентификация карты в ходе ее использования. В данном случае под аутентификацией понимается процесс доказательства того, что данная карта (или приложение на карте) выпущена банком, авторизированным на это соответствующей платежной системой.

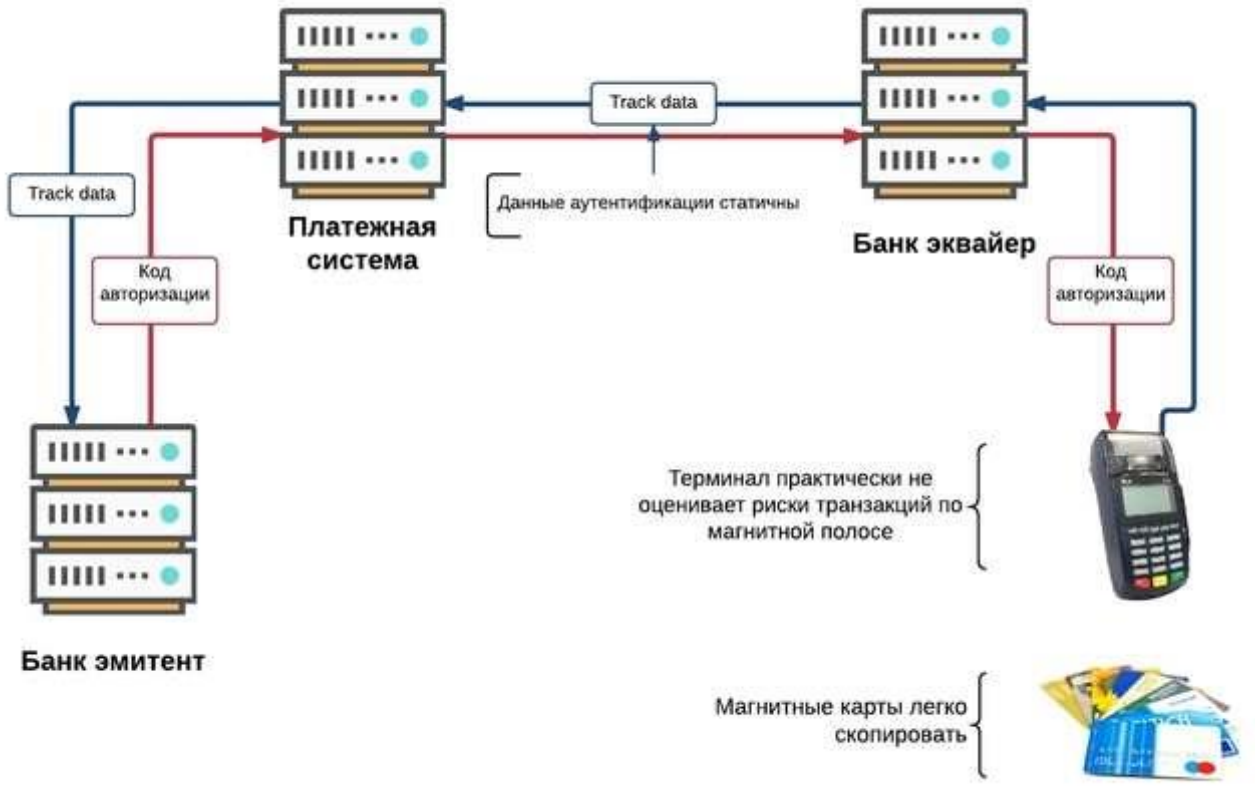
Как происходит процесс аутентификации карты?

В общем случае, прочитав данные карты, терминал отправляет их через банк-эквайер и платежную систему банку-эмитенту. Эмитент на основании данных карты определяет ее подлинность.

В этом процессе заключается одна из основных проблем безопасности платежей по магнитным картам. С одной стороны, целостность данных

магнитной карты надежно защищена кодом CVV/CVC (CVC – Card Verification Code, CVV – Card Verification Value) и модифицировать их бесполезно. С другой стороны, довольно просто скопировать всю карту целиком.

2.1 Аутентификация магнитной карты на основе статических данных



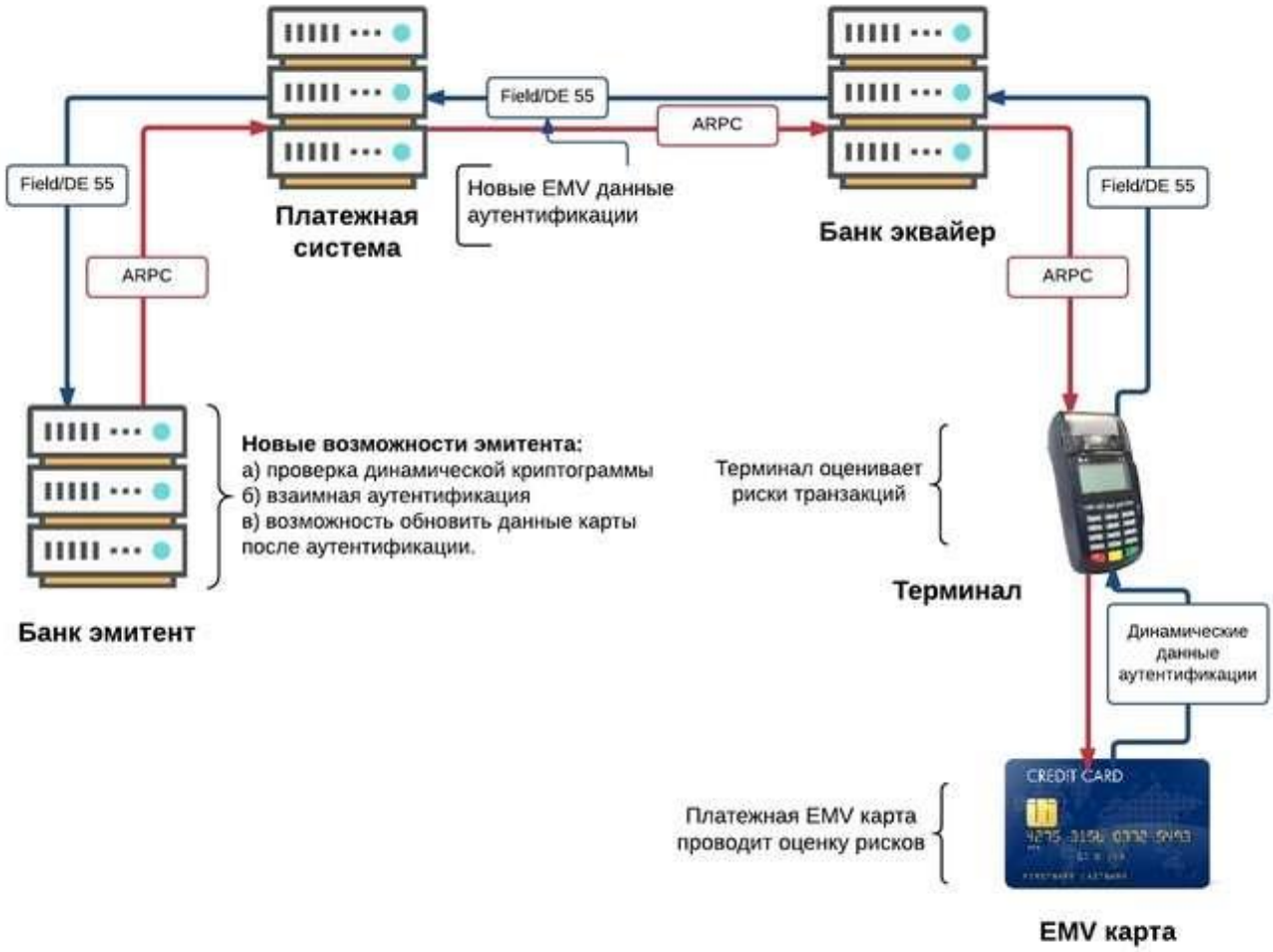
Для аутентификации в транзакциях по магнитной карте используются статические данные карты. Эти данные карты каждый раз передаются в банк-эмитент и не меняются на протяжении всего срока действия карты. Вдобавок, платежный терминал практически не оценивает риски транзакций по картам с магнитной полосой. В итоге – в случае полного копирования карты – банк-эмитент не сможет достоверно определить подлинность такой карты. Соответственно, вероятность проведения мошеннической операции достаточно высока.

2.2 Аутентификация EMV-карты на основе динамических данных

Как этот вопрос решают EMV-карты?

Решением вышеописанной проблемы является цифровая подпись статических данных карты и данных транзакции, которые отправляются эмитенту. Поскольку цифровая подпись является уникальной для каждой

транзакции, подделка или копирование EMV-карты является нетривиальной задачей.



Рассмотрим подробнее, как происходит динамическая аутентификация карты в ходе EMV-транзакции. Процесс транзакции начинается в момент установки карты в терминал. Терминал передает карте данные транзакции (сумма, валюта, страна и т.д.). Затем карта и терминал производят взаимную проверку рисков транзакции. Если оба устройства все «устраивает» то карта подписывает данные транзакции, а терминал заполняет полученными данными поле (таг или тэг) «DE 55» и отправляет его в банк-эквайер. Тот, в свою очередь, отправляет сообщение банку-эмитенту.

Эмитент, получив поле «DE 55», проверяет подлинность подписи (далее криптограммы) карты, которая рассчитана на основании динамических данных текущей транзакции, тем самым проверяя подлинность самой карты.

Описанный выше процесс является сильно упрощенной моделью EVM-транзакции. Однако он раскрывает главный аспект безопасности EVM-платежей – использование для аутентификации карты динамических данных вместо статических.

Стоит отметить, что у эмитента появляются новые возможности:

- проверка динамической криптограммы карты
- взаимная аутентификация: эмитент может выслать свою криптограмму карте
- возможность обновить данные карты после аутентификации (например, заблокировать карту или сменить лимит).

Также в EMV-транзакциях существенная роль отведена терминалу и его системе оценки рисков, согласно которой и терминал, и карта могут принимать решения о возможности проведения транзакции.

3. Внутренняя структура и безопасность EMV-карты

По большему счету, микропроцессорная карта стандарта EMV является обычной смарт-картой (почитать раз, два, три), в основе которой лежат стандарты ISO/IEC 7816 или ISO/IEC 14443 (для бесконтактной).

Реализация EMV-карты может быть выполнена как на базе JavaCard и GlobalPlatform, так и с помощью нативных методов смарт-карты. Аналогично обычными операционными системами (ОС), карточные ОС также имеют файловую структуру и приложения. В контексте этой статьи, наиболее интересны именно платежные приложения EMV-карты. Поэтому будем рассматривать именно их.

Что представляет собой платежное EMV-приложение?

С точки зрения пользователя (терминала или банкомата), платежное EMV-приложение – это программный продукт с интерфейсом, детально описанным в стандарте EMV.

Интерфейс представляет собой серию команд для проведения транзакций и управления EMV-приложениями. Подробную информацию можно найти в «EMV Book 3 Application Specification». Несмотря на существование стандарта, платежные приложения компаний Visa и MasterCard имеют отличия в реализации. Также могут отличаться и разные приложения одной компании. Например, «M/Chip 4» и «M/Chip Advance» компании MasterCard.

Вне зависимости от реализации, каждое приложение имеет свой собственный идентификатор, так называемый AID (Application Identifier). Он указывает к какому типу платежной системы относится приложение. По идентификатору

приложения AID терминал определяет возможность проведения транзакции или, в случае нескольких приложений строит список поддерживаемых приложений и предлагает выбрать одно из них.

Если на карте реализована файловая структура и управление приложениями, какие же механизмы обеспечивают безопасность данных от доступа извне?

Тут стоит разделить время жизни карты до момента выпуска банком, и после.

Первичный доступ к чистой карте обычно регламентируется производителем чипов. Чаще всего каждая партия карт имеет свой ключ карты, с помощью которого необходимо аутентифицироваться с картой в ходе ее прошивки.

На следующем этапе доступ к файловой системе и приложениям обычно регулируется операционной системой. Она также имеет свой собственный ключ, и, соответственно, для доступа требуется аутентификация.

Далее установленное приложение проходит процесс персонализации карты. Персонализация представляет собой загрузку параметров и ключей приложения, которые определяют безопасность EMV-транзакций. Для доступа к этому процессу также требуется аутентификация с помощью ключа приложения.

После установки приложения и его персонализации вышеперечисленные доступы обычно закрываются навсегда. Что исключает возможность проникновения «внутрь» после выпуска карты.

Итого: ключ карты, ключ ОС и ключ приложения защищают карту от стороннего вмешательства на различных стадиях ее производства. В случае если в ходе изготовления часть карт будет дискредитирована (например, украдена), эти ключи защитят карты от вмешательства извне. А без знания ключей карты становится практически полностью бесполезными.

Некоторые данные приложения могут быть модифицированы и после выпуска карты. Изменения могут быть выполнены так называемыми скриптовыми командами. Исключительные права на внедрение изменений принадлежат эмитенту. Такая возможность предусмотрена, чтобы в любой момент времени, эмитент мог заблокировать или разблокировать карту, обновить лимиты или настройки карты. Обновление данных производится терминалом или банкоматом только после успешной онлайн транзакции (аутентификации с банком). Данные приходят на карту от эмитента в чистом виде, однако имеют в себе аналог цифровой подписи – MAC, который

гарантирует целостность данных. Для расчета MAC используется соответствующий ключ приложения (один из трех DES ключей загружаемых в приложение).

Отдельными пунктами являются модификация оффлайн пин-кода (offline PIN) и счетчика лимита неудачных вводов пин-кода (PinTryLimit). Эти изменения также выполняются скриптовой командой с MAC-подписью. Однако, при смене пин-кода эти команды дополнительно шифруются с помощью специального ключа, предназначенного исключительно для выполнения описанного процесса.

4. Данные EMV-приложения

Аналогично картам с магнитной полосой, EMV-приложения также имеют открытые данные доступные для чтения. И хотя само приложение прочитать невозможно, как невозможно добраться и до ключей и пин-кода – доступ к открытым данным приложения всегда открыт.

EMV Tag	Chip Data	EMV Tag	Chip Data
42	Issuer Identification Number (IIN)	9F 07	Application Usage Control
4F	Application Dedicated File (ADF) Name	9F 08	Application Version Number (CARD)
50	Application Label	9F 0B	Cardholder Name Extended
57	Track2 Equivalent Data	9F 0D	Issuer Action Code (IAC) - Default
5A	Application Primary Account Number (PAN)	9F 0E	Issuer Action Code (IAC) – Denial
5F 20	Cardholder Name	9F 0F	Issuer Action Code (IAC) – Online
5F 24	Application Expiration Date	9F 11	Issuer Code Table Index
5F 25	Application Effective Date	9F 12	Application Preferred Name
5F 28	Issuer Country Code	9F 1F	Track1 Discretionary Data
5F 2D	Language Preference	9F 20	Track2 Discretionary Data
5F 30	Service Code	9F 2D	ICC PIN Encipherment Public Key Certificate
5F 34	Application PAN Sequence Number	9F 2E	ICC PIN Encipherment Public Key Exponent
5F 50	Issuer URL	9F 2F	ICC PIN Encipherment Public Key Remainder
5F 53	International Bank Account Number (IBAN)	9F 32	Issuer Public Key Exponent
5F 54	Bank Identifier Code (BIC)	9F 38	Processing Options Data Object List (PDOL)
5F 55	Issuer Country Code (Alpha2 Format)	9F 3B	Application Reference Currency
5F 56	Issuer Country Code (Alpha3 Format)	9F 42	Application Currency Code
82	Application Interchange Profile (AIP)	9F 43	Application Reference Currency Exponent
84	Dedicated File (DF) Name	9F 44	Application Currency Exponent
87	Application Priority Indicator	9F 45	Data Authentication Code (DAC)
88	Short File Identifier	9F 46	ICC Public Key Certificate
8C	Card Risk Management Data Object List (CDOL) 1	9F 47	ICC Public Key Exponent
8D	Card Risk Management Data Object List (CDOL) 2	9F 48	ICC Public Key Remainder
8E	Cardholder Verification Method (CVM) List	9F 49	Dynamic Data Object List (DDOL)
8F	Certificate Authority (CA) Public Key Index	9F 4A	Static Data Authentication (SDA) Tag List
90	Issuer Public Key Certificate (IPKC)	9F 4B	Signed Dynamic Application Data
92	Issuer Public Key Remainder	9F 4D	Log Entry
93	*Signed Static Application Data*	Key	MDK_{AC}
94	Application File Locator (AFL)	Key	MDK_{SMI}
97	Transaction Certificate Data Object List (TDOL)	Key	MDK_{SMC}
9F 05	Application Discretionary Data	Key	MDK_{ION}
		Key	MDK_{CVC3}

Данные EMV приложения

О каких данных идет речь?

На картинке выше приведен ориентировочный список данных, хранящихся внутри EMV-приложения. Конечно, для каждого конкретного приложения он может несколько отличаться. На данном этапе важно отметить, что персональная информация клиента не хранится в EMV-приложении. Действительно, больший объем памяти чипа позволяет платежным системам и банкам хранить на карте больше информации – однако персональной информации клиента там нет.

Предыдущая картинка наглядно иллюстрирует факт того, что на карте хранится множество технических данных, необходимых для эффективного проведения операций и доступа к счету. Данные EMV-приложения размещаются в записях (рекордах или треках). Их список можно получить в ответ на команду «Get Processing Options». Конкретную запись можно прочитать с помощью команды «Read Record». Внутри могут находиться: сертификаты ключей, номер карты (PAN – Primary Account Number), списки методов проверки карты (CVM list– Card Verification Methods list) и множество другой информации. Чтение этих записей очень похоже на чтение треков с магнитной полосы. Данные технических настроек карты, счетчики и лимиты можно получить командой «Get Data», указав требуемый тип.

Интересно, что практически все данные о счете держателя карты и настройках приложения можно вычитать из карты без каких либо трудностей. Единственное до чего не добраться – это ключи приложения и значение пин-кода.

Можно ли скопировать данные на с одной чиповой карты на другую?

Если у вас есть карта с «чистым» (не персонализированным) приложением, то технически это реализуемо. Однако за счет отсутствия возможности сделать копию ключей карты – приложение будет генерировать неверные подписи транзакции. В результате – эмитент будет отклонять любые онлайн-операции. Также отсутствие ключей не позволит провести CDA /DDA аутентификацию. Единственная брешь — это SDA офлайн. Однако на данный момент этот метод в виде единственного метода аутентификации считается устаревшим. Далее будет детально рассмотрено, как защищена EMV-транзакция.

Можно ли скопировать данные EMV-приложения на магнитную полосу?

Из данных EMV-приложения можно составить треки для карты с магнитной полосой, за исключением одного небольшого параметра – кода обслуживания (Service Code). В качестве данных для EMV-приложения, код обслуживания указывает терминалу, что транзакция должна быть проведена с использованием приложения карты. Если взять этот код «как есть» и скопировать на магнитную дорожку – терминал будет пытаться выполнить

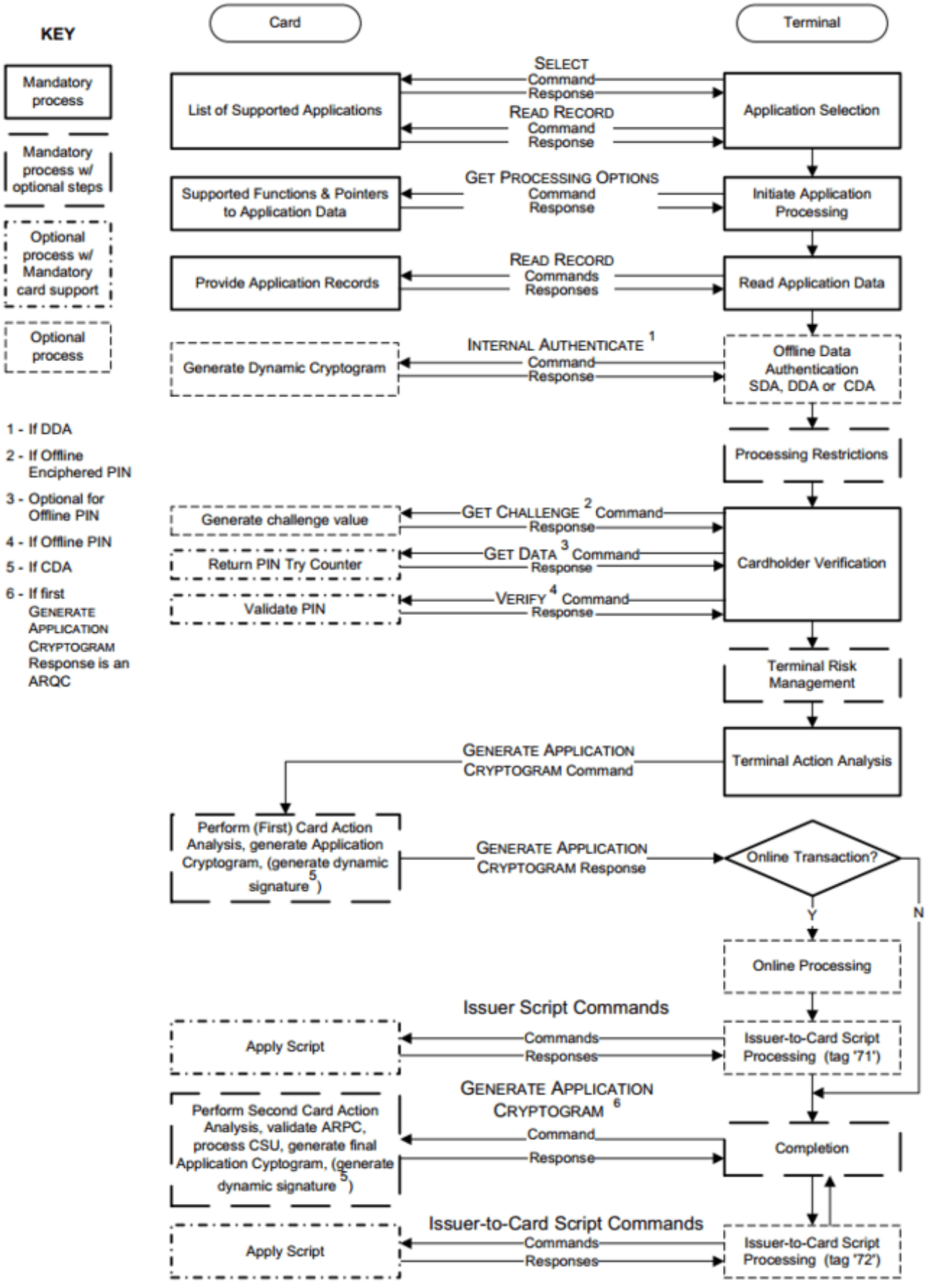
транзакцию с помощью приложения. Казалось бы, можно отредактировать код обслуживания, но целостность данных защищена кодом CVV/CVC кодом. Он является ближайшим аналогом цифровой подписи.

Создается ощущение, что EMV-карта защищена от копирования со всех сторон. Хотя все-таки известна одна тривиальная возможность. Для режима совместимости производители выпускают EMV-карты комбинированного типа – то есть с микропроцессором и магнитной полосой. Существует возможность скопировать данные магнитной полосы на другую комбинированную карту с нерабочим чипом (чистым или сожженным) и попытаться провести так называемый fallback (при невозможности считать чип, терминал проводит операцию по магнитной полосе). В данный момент такие операции не приветствуется платежными системами, а риск по этим операциям ложится на эквайра или эмитента.

5. Безопасность EMV-транзакции

Существует два разных (хотя и выполняющих одну и ту же функцию) варианта проведения платежной транзакции – онлайн и офлайн. Выше мы в общих чертах рассматривали онлайн-транзакцию, которую эмитент подтверждает в режиме реального времени. Офлайн-транзакция проводится терминалом без моментального подтверждения банком. Такие транзакции используются для операций с низким уровнем риска или в случае, например, отсутствия связи с банком-эмитентом.

Для этих двух видов транзакций существует соответственно два вида аутентификаций – онлайн и офлайн. В случае выполнения онлайн-аутентификации, операция производится с участием эмитента, а офлайн-аутентификация подтверждается платежным терминалом. Стоит уточнить, что во время проведения онлайн- транзакции может выполняться как онлайн, так и офлайн-аутентификация одновременно (если и карта, и терминал это поддерживают). Несмотря на избыточность схемы, на этапе аутентификации не всегда понятно в каком режиме будет проходить транзакция.

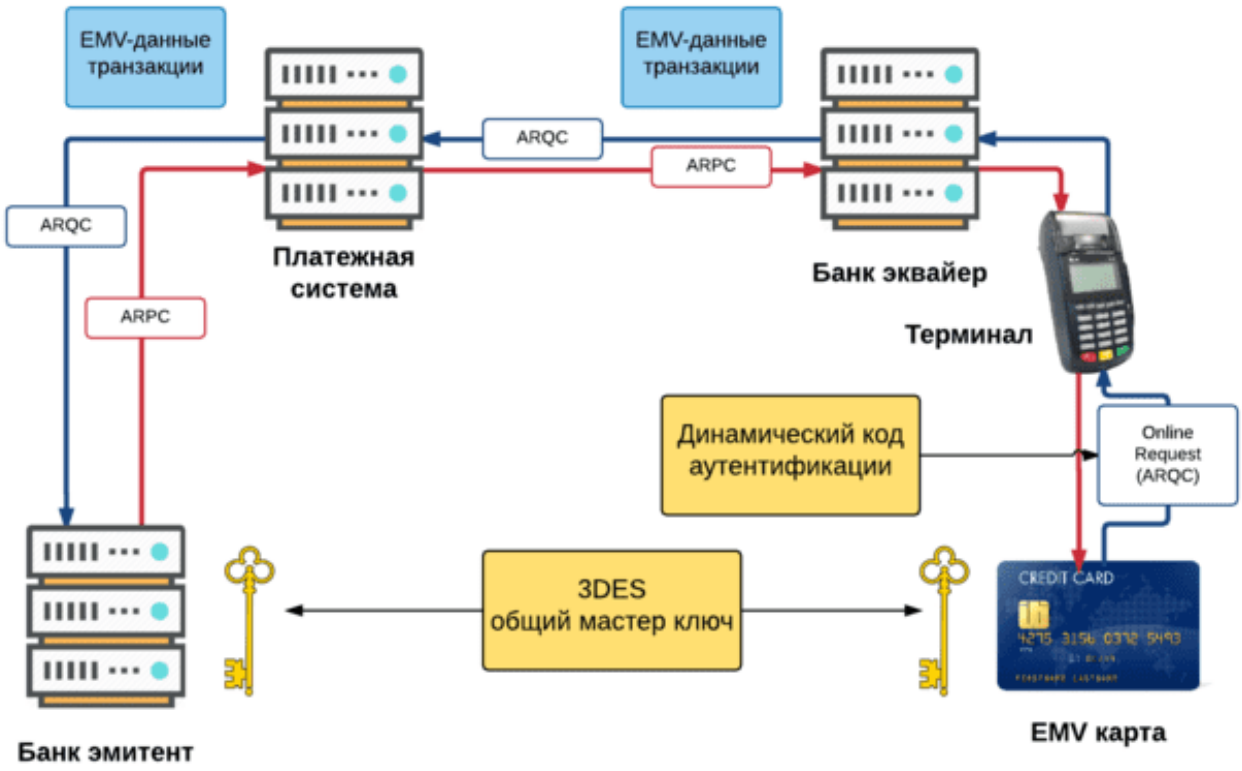


Порядок выполнения транзакции карта – терминал

Функции безопасности, рассматриваемые ниже, являются только частью EMV-транзакции. Помимо аутентификации, к функциям безопасности можно отнести: оценку рисков проведения транзакции и верификацию держателя карты (онлайн и офлайн-пин, размер суммы транзакции, страна, валюта, прочее).

5.1 Онлайн EMV-транзакция

Основным методом подтверждения подлинности карты в онлайн-транзакциях является аутентификация карты онлайн. В основе данного метода лежит генерация картой криптограммы ARQC (Authorisation Request Cryptogram) для каждой платежной операции. Давайте рассмотрим этот процесс подробнее.



Онлайн EMV-транзакция

В основе генераций и проверок криптограмм лежит алгоритм 3DES. Эмитент и карта владеют общим секретным ключом MKac (Application Cryptogram Master Key). В начале транзакции карта генерирует на основе MKac сессионный ключ SKac (Application Cryptogram Session Key). Криптограмма

ARQC длинной 8 байт генерируется картой с помощью алгоритма MAC, на сессионном ключе SKac с использованием данных транзакции.

В процессе транзакции, сгенерированная картой криптограмма ARQC, отправляется в банк-эмитент, Банк сверят пришедшую ARQC с криптограммой которую, рассчитал самостоятельно. Для этой операции банком генерируется сессионный ключ, затем на основании пришедших данных транзакции, рассчитывается собственный ARQC. Если собственный (сгенерированный эмитентом) ARQC и ARQC карты сходятся – карта подлинная.

Далее эмитент по похожему алгоритму на основе динамических данных транзакции и данных ответа генерирует ARPC (Authorisation Response Cryptogram) и отсылает эту криптограмму назад карте. В тот момент, когда карта подтвердит пришедший ARPC, взаимная аутентификация карты и эмитента – выполнена.

Выше описан основной механизм аутентификации карты, который используется для онлайн-транзакций. Как уже было сказано, в онлайн-транзакции может присутствовать офлайн-аутентификация. Однако, чтобы не усложнять, рассмотрим детальное описание офлайн-аутентификации в контексте офлайн-транзакции.

Следующим методом безопасности являются расширенные данные в Field/DE 55 которые передаются в банк-эмитент. Field/DE 55 содержит результаты работы карты и терминала, оценки рисков и анализа транзакции.

Как показано на изображении выше, в Field/DE 55 содержится важная информация. Например, Terminal Verification Result, Card Verification Result, которые в сумме с остальными данными помогают понять эмитенту и платежной системе как происходит транзакция и предоставляют множество дополнительных деталей для оценки рисков транзакции.

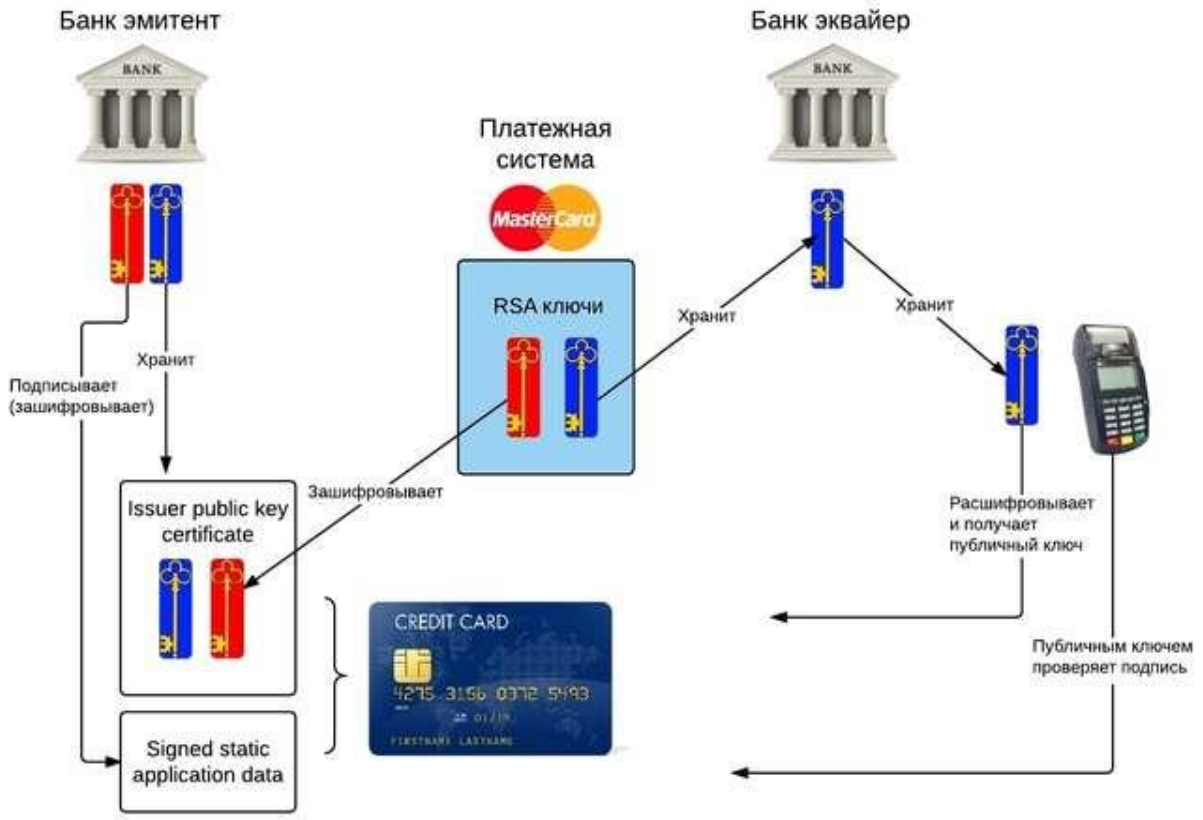
5.2 Офлайн EMV-транзакция

Особенность офлайн-транзакции заключается в том, что транзакция проводится картой и терминалом без обращения к банку и платежной системе. В процессе такой транзакции карта может одобрить транзакцию в пределах установленного лимита, а терминал, в свою очередь, отправляет информацию в банк позже по расписанию, либо когда появится связь с банком. Такие офлайн-транзакции предоставляют дополнительные преимущества как банку-эмитенту, так и владельцу карты. Например, владелец может расплатиться даже, если связи с банком нет. Либо же, если сумма небольшая – операция пройдет намного быстрее.

Как происходит аутентификация карты при офлайн-транзакции?

Ранее упоминалось, что онлайн- и офлайн-аутентификации используют разные технологии. Если онлайн использует криптографический алгоритм 3DES, то в случае с офлайн используется RSA с ассиметричными ключами. Зачем же использовать такие разные технологии? Все дело в том, что при онлайн-аутентификации, ключи хранят только карта и банк. В случае же офлайна – ключ нужно доверить терминалу. Учитывая наличие большого количества терминалов, существует вероятность, что секретный ключ доверенный терминалам недолго останется секретным.

Т.к. детальное описание офлайн-аутентификации карты достаточно большое, рассмотрим упрощенную модель.



Static Data Authentication

Во главе всего стоит платежная система (точнее центр сертификации), которая выпускает пару ключей: приватный ключ (красный) и публичный ключ (синий). Банк-эмитент также имеет свою пару ключей. Для своих ключей эмитент специальным образом генерирует сертификат (Issuer Public Key Certificate), который содержит в себе публичный ключ эмитента. Этот сертификат подписан (зашифрован) приватным ключом платежной системы. В процессе персонализации этот сертификат загружается на карту.

Когда платежный терминал устанавливают в торговую точку и подключают к системе, публичный ключ платежной системы через банк-эквайер загружается в терминал.

В процессе офлайн-транзакции терминал производит офлайн-аутентификацию карты. Сначала терминал вычитывает из карты Issuer Public Key Certificate, и с помощью публичного ключа платежной системы проверяет правильность подписи сертификата (т.е. расшифровывает). Если подпись верна – извлекается публичный ключ эмитента. Далее, с помощью публичного ключа эмитента, проверяется подпись критических данных карты, чем и подтверждается ее подлинность.

Описанный выше метод относится к статической аутентификации SDA (Static Data Authentication). В настоящее время чаще используются динамические аутентификации: DDA (Dynamic Data Authentication) и CDA

(Combined Data Authentication), которые включают в себя SDA и дополнительно, по аналогии с онлайн, подписывают данные, которые курсируют между терминалом и картой. Данные подписываются приватным ключом карты, который загружается на карту в процессе персонализации. Подпись проверяется терминалом с помощью публичного ключа, восстановленного из соответствующего сертификата.

Технология SDA позволяет терминалу проверить, что данные на карте не модифицированы. Однако, она не позволяет полностью идентифицировать подлинность карты (существует возможность скопировать SDA-данные). В свою очередь, технологии DDA и CDA позволяют подтвердить подлинность карты, потому что карта является носителем уникального приватного ключа, чей сертификат (публичный ключ) подписан приватным ключом эмитента (сертификат эмитента (его публичный ключ) подписан приватным ключом платежной системы).

Диаграммы SDA,DDA и CDA, EMV Book 2

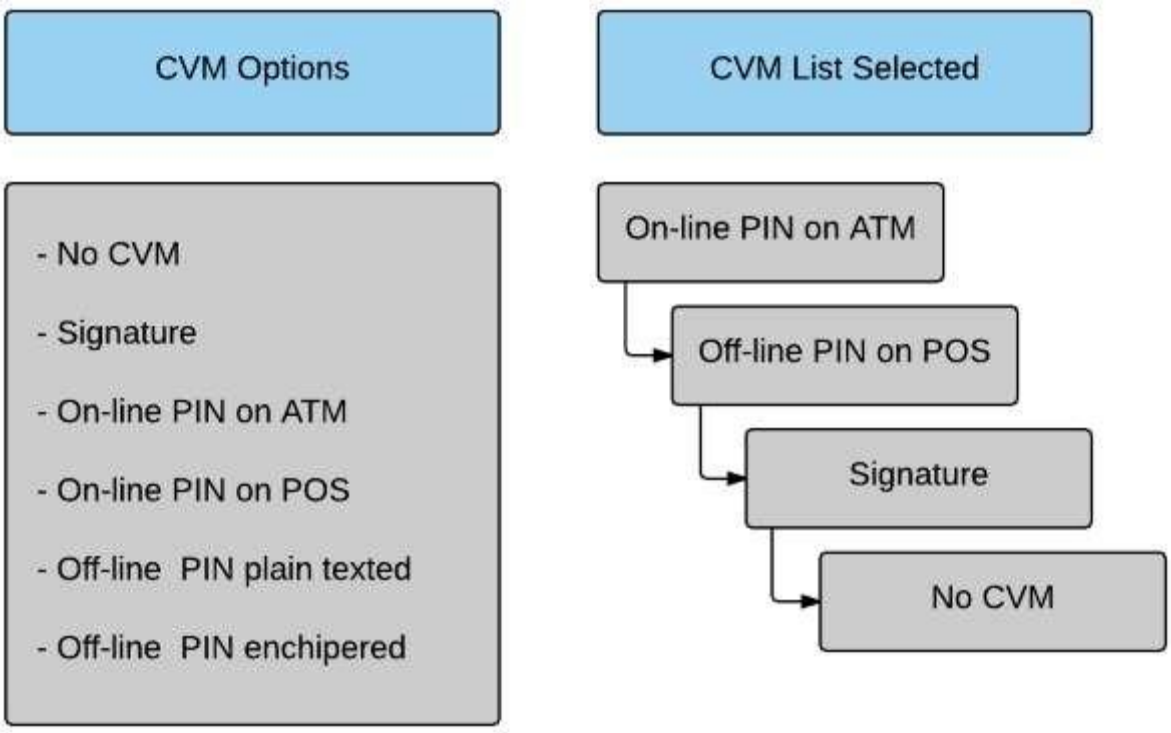
Технологии DDA и CDA уже содержат в себе SDA и в целом сходны. Оба алгоритма используют уникальный ключ карты и динамические данные. DDA-аутентификация является отдельной операцией и выполняется до основного цикла процесса транзакции. CDA выполняется в основном цикле транзакции, а в качестве подписываемых данных дополнительно используется криптограмма карты. В целом, сегодня, технология DDA более распространена, хотя CDA является более предпочтительной в использовании.

Помимо цифровой подписи, терминал и карта умеют оценивать риски транзакции. Для офлайн-транзакции карта может оперировать несколькими видами счетчиков транзакций и аккумуляторов офлайн-сумм, валютами и странами, офлайн-пином и его лимитами, а также дополнительными правилами. В процессе персонализации карты эмитент имеет возможность ограничить максимальное количество последовательных офлайн-транзакций и/ или максимальную сумму транзакции (нижними и верхними лимитами), таким образом определяя уровень риска.

Для каждой из реализаций приложения конкретной платежной системы существует свой набор правил, на основании которых карта может принимать решения проводить офлайн, онлайн или отклонять транзакцию. Список этих правил достаточно гибкий и может по-разному настраиваться эмитентом для каждого карточного продукта. В процессе решения могут участвовать результаты предыдущих транзакций, офлайн-счетчики, результаты проверки пина и т.д.

6. Проверка держателя карты CVM (Cardholder verification method)

Практически вся статья была посвящена транзакциям и процессу аутентификации карты, а пользователю карты уделялось мало внимания. С появлением технологии EMV проверка держателя карты не слишком видоизменились. В данный момент наиболее популярными методами проверки являются: проверка пин-кода (онлайнового и/или офлайнового) и подпись владельца карты. Так сложилось, что с приходом EMV не все платежные терминалы обладают одинаковыми возможностями проверки держателя карты (например, по причине возраста оборудования). В свою очередь, разные EMV приложения также могут быть ограничены в возможностях. Поэтому терминалу и карте приходится выбирать подходящий метод проверки держателя карты. Для этого используются так называемые CVM-списки. В CVM-списке определены методы проверки держателя карты и их приоритеты. И платежное приложение, и терминал имеют свои собственные списки. Итоговый список определяется путем объединения списков терминала и приложения. Из полученного итогового списка терминал выбирает общий CVM- метод с наибольшим приоритетом и осуществляет проверку держателя карты.



Пример такого списка представлен на картинке выше. Например, если карта вставлена в банкомат – будет запрошен онлайн-пин, если в терминал – офлайн-пин. В случае, если устройство не имеет пин-пада – будет запрошена проверка подписи. Во всех остальных случаях проверка держателя карты производиться не будет.

Заключение

В данной статье были поверхностно рассмотрены платежное EMV-приложение и хранимые в нем данные, описаны основные отличия в процессах проведения транзакций по магнитным и EMV-картам. Также были рассмотрены процедуры проведения онлайн и офлайн-транзакций и механизмы обеспечения их безопасности. Конечно же, каждый аспект технологии EMV, имеет гораздо большую глубину и степень сложности. Однако, надеюсь, что статья дала общее понимание принципа работы платежных EMV-карт и проведения платежей с их помощью.

В заключении можно сказать, что платежная EMV-карта сложный и высокотехнологичный продукт, надежно защищающий доступ к вашему счету в банке. Микропроцессорную EMV-карту практически невозможно скопировать, а каждая транзакция защищена уникальной цифровой подписью. Любые действия, происходящие внутри карты, регламентируются строгим набором правил с указаниями как поступать в каждом конкретном случае. В процессе создания платежные EMV-приложения проходят обязательную многоуровневую сертификацию и получают разрешение от платежной системы на их использование. Программировать такие карты сложно и интересно. Впрочем, описание этого процесса может растянуться еще не на одну статью.

Системы контроля и управления доступом

Назначение и классификация систем контроля и управления доступом

Согласно [28], комплексное обеспечение безопасности объекта определяется как деятельность по созданию условий и обеспечению ресурсами для предотвращения и уменьшения последствий от угроз различного характера. Для формирования комплексной системы защиты объекта в первую очередь необходимо разработать концепцию безопасности. В [29] подробно рассмотрены вопросы разработки и реализации концепции безопасности и показано, что в общем случае система защиты объекта должна включать в себя следующие элементы:

- физическую охрану;
- организационные мероприятия;
- технические средства обеспечения безопасности (ТСОБ).

Физическая охрана обеспечивается наличием стационарных и мобильных постов, пунктов диспетчерского наблюдения, групп оперативного реагирования.

К организационным мероприятиям в первую очередь относятся действия, направленные на обеспечение контрольно-пропускного режима (КПР), который необходим для обеспечения упорядоченного доступа сотрудников, посетителей и транспорта на территорию охраняемого объекта. Под КПР понимается комплекс организационно-правовых ограничений и

правил, инженерно-технических решений, устанавливающих порядок прохода посетителей через контрольно-пропускные пункты [30].

Структура и количественный состав комплекса ТСОБ могут варьироваться в зависимости от условий функционирования объекта и количества рубежей защиты. Важным элементом каждого рубежа являются СКУД, которые позволяют обеспечивать безопасность персонала и посетителей, а также сохранность материальных и информационных ресурсов предприятия посредством организации КПР. Подобные системы успешно применяются как на промышленных объектах, так и в жилых помещениях, офисных центрах, магазинах, на автостоянках и т.д. При этом наличие СКУД не только позволяет повысить уровень безопасности путем предотвращения несанкционированного доступа на охраняемую территорию, но и оперативно реагировать на поведение персонала и посетителей.

В соответствии с [26, 31] основными задачами СКУД являются:

- предотвращение несанкционированного доступа в контролируемые зоны с ограниченным доступом, в том числе регистрация таких событий;
- организация возможности беспрепятственного прохода (проезда) в зоны со свободным доступом;
- контроль и учет доступа посетителей на объект;
- обеспечение условий для соблюдения внутриобъектового режима и выполнения соответствующих обязанностей персоналом объекта;
- интеграция с иными системами безопасности (например, автоматическая разблокировка замков на дверях при срабатывании пожарной сигнализации).

СКУД представляет собой совокупность программно-аппаратных средств, обладающих технической, информационной, программной и эксплуатационной совместимостью. Входящие в состав подобных систем устройства можно разделить на следующие группы:

- устройства считывающие (УС): считыватели карт доступа, биометрические сканеры, устройства распознавания автомобильных номеров, кодонаборные устройства и др.;
- средства обнаружения различных материалов: металлодетекторы, обнаружители взрывчатых веществ и радиационных материалов;
- устройства обработки информации: контроллеры, панели управления;
- устройства исполнительные (УИ): турникеты, калитки, шлагбаумы, электромеханические, электромагнитные и механические замки и др.;
- вспомогательные устройства: модули связи между компонентами СКУД, инжекторы питания, адаптеры и др.

Пример реализации современной СКУД представлен на рис. 1.1.

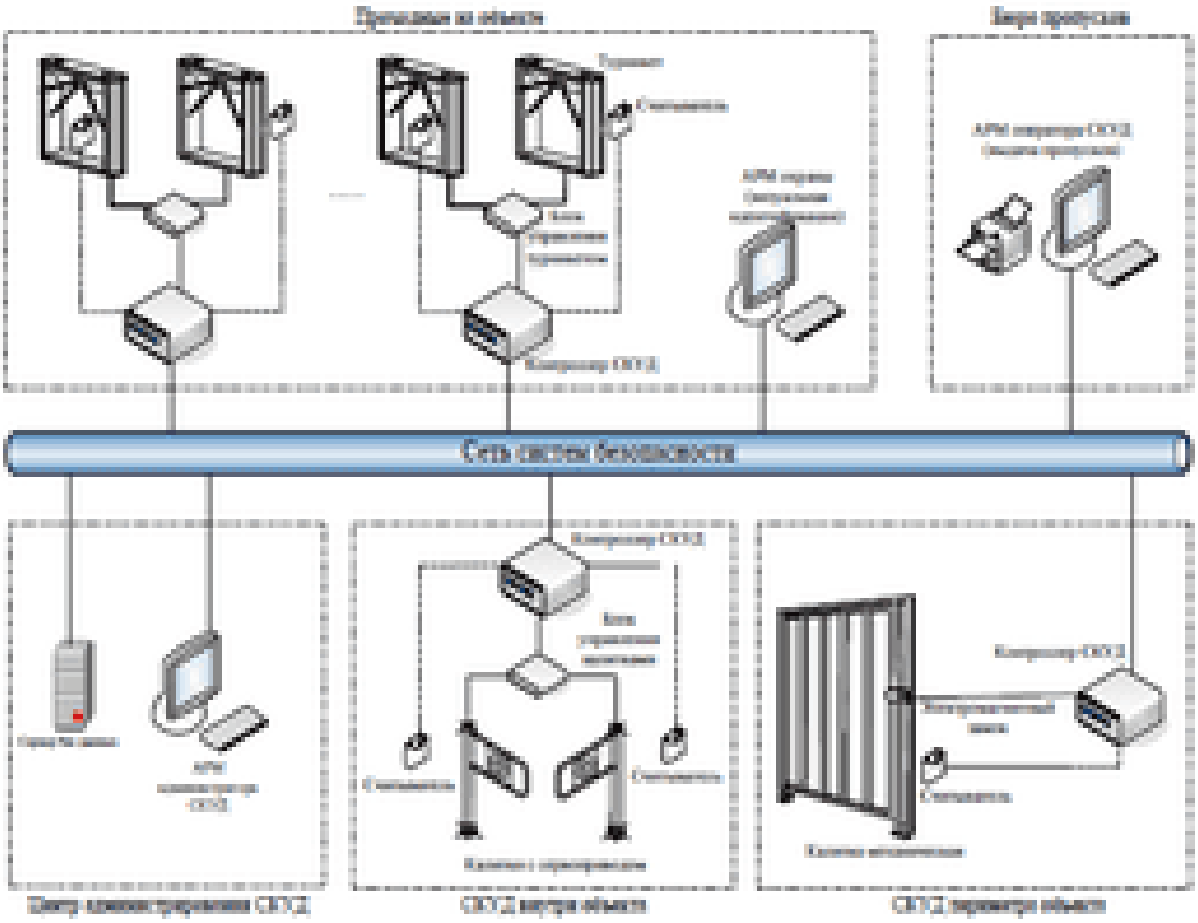


Рис. 1.1 – Пример реализации современной СКУД

Как было сказано ранее, СКУД может быть реализована различными способами в зависимости от условий функционирования конкретного объекта и решаемых задач. В некоторых случаях могут применяться автономные СКУД для каждой точки прохода. В последнее время наиболее популярны распределенные системы, поддерживающие масштабирование при расширении контролируемой зоны. В таких случаях контроллеры доступа являются самостоятельными устройствами, осуществляющими процесс управления с использованием специализированных удаленных интерфейсных модулей.

Деление средств и систем КУД на классы может выполняться на основе сравнительного анализа ряда функциональных возможностей. В ГОСТ Р 51241-2008 «Средства и системы контроля и управления доступом» [23] наряду с общими техническими требованиями и методами испытаний представлена классификация подобных систем по:

- способу управления;
- числу контролируемых точек доступа;
- функциональным характеристикам;
- виду объектов контроля;
- уровню защищенности системы от несанкционированного доступа.

Использование современных идентификационных признаков в СКУД
Мобильные устройства как способ идентификации

Появление смартфонов – мобильных телефонов, вычислительная мощность которых сегодня сопоставима со стационарными компьютерами, – стало символом одного из важнейших прорывных шагов в развитии инфокоммуникаций, качественно повлиявших на жизнь человека. На базе смартфонов за последние несколько лет появились новые виды электронной техники (планшеты, смарт-браслеты, смарт-часы и прочие гаджеты [74]), которые активно используются практически во всех видах жизнедеятельности человека. При этом в большинстве случаев подобные устройства используются для обеспечения возможности человека быть всегда «на связи» посредством различных технологий (GSM, LTE, Wi-Fi, Bluetooth и т.д. [75]). Данный факт позволяет объединить подобные устройства в единый класс – мобильные устройства связи.

Согласно статистике [76], представленной Международным Союзом Электросвязи (США), на 2016 г. в мире используется около 7 377 млн мобильных устройств связи, учитывая, что население Земли составляет около 7 448 млн человек. При этом, согласно [77], мобильные устройства связи составляют более 50% продаж всей потребительской электроники в мире. Очевидно, что при таком состоянии рынка доступность данного рода гаджетов увеличивается, в том числе из-за снижения их стоимости. На сегодняшний день у многих людей есть несколько подобных устройств, например ноутбук, планшет, смартфон и др.

С одной стороны, при таком положении дел вполне логичным представляется усиление потребительского отношения к мобильным устройствами связи и их значительному понижению в списке материальных благ человека. С другой стороны, несмотря на эффект массовости и усиления потребительского отношения к гаджетам, в последнее время наблюдается обратный эффект – персонализация гаджетов. Появление социальных сетей, мессенджеров, развитие бесконтактных технологий оплаты приводит к тому, что человеку не хочется часто менять устройство связи. Для него важна возможность постоянной коммуникации. Именно поэтому люди все чаще ассоциируют конкретные мобильные устройства как неотъемлемый аксессуар, «привязывая» их к различным учетным записям, банковским картам и т.д. Потеря или кража гаджетов нередко вызывает у современного человека серьезный дискомфорт, на время «выбивая его из жизненной колеи». Таким образом, в обществе все чаще заметна тенденция – тщательнее и бережнее относиться к обеспечению сохранности личного мобильного устройства связи, нередко позволяющего идентифицировать себя в среде инфокоммуникаций.

Издательство Nature.com опубликовало отчет [78] об исследовании американских и бельгийских ученых, в котором оценивалась возможность идентификации человека в обществе на основе данных о его местонахождении во времени и в пространстве. При этом основным

инструментом исследования являлись мобильные устройства связи. На протяжении 15 месяцев накапливалась и анализировалась анонимизированная информация о времени и месте звонков и SMS-сообщений 1,5 млн абонентов. На основе полученных данных строились индивидуальные «следы» (траектории) передвижения людей.

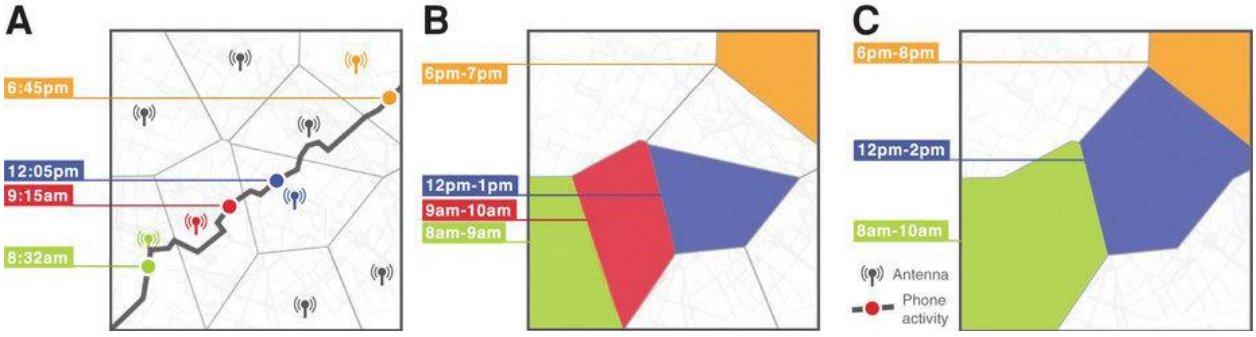


Рисунок 3.1 – Пример вычисления траектории движения анонимного пользователя, построенной путем исследования активности использования мобильного устройства связи в течение дня Результаты исследования показали, что в 95% случаев достаточно знать всего 4 пространственно-временные точки из открытых источников (Facebook, Twitter, Foursquare, LinkedIn и т.п.) для идентификации и деанонимизации человека среди миллионов других членов общества. Одним из наиболее значимых результатов вышеуказанного исследования является подтверждение факта персонализации мобильных устройств связи в современном мире и возможности рассмотрения их в качестве современных идентификационных признаков.

По мнению автора, данное предположение не является субъективным в связи с наличием множества других подтверждений. Например, одним из направлений развития современных ИТ-компаний является концепция BYOD («Bring your own device») [79]. Принцип, заложенный в эту концепцию, подразумевает, что для доступа к работе с ресурсами компании сотрудники используют личные ноутбуки, планшеты, смартфоны и т.д. Подобная стратегия активно поддерживается такими крупными корпорациями, как Microsoft [80], VMware [81], SAP [82], IBM [83]. Несмотря на то что для подобных компаний затраты на приобретение и сопровождение корпоративных мобильных устройств являются приемлемыми, они охотно поддерживают данную концепцию, предоставляя возможности своим сотрудникам комфортно работать удаленно, вне зависимости от места и времени. При этом современный взгляд на обеспечение безопасности объектов информатизации [84-86] лишь подтверждает приведенные выше слова о персонализации мобильных устройств: в большинстве случаев сотрудники намного тщательнее заботятся о сохранности своего гаджета, чем о безопасности выданного им отдельного физического идентификатора (например, proximity карты доступа).

Все это еще раз подтверждает, что на сегодняшний день мобильные устройства связи настолько прочно вошли в нашу жизнь, что без них сложно представить будни каждого человека.

В настоящее время на рынке систем безопасности наблюдается повышение спроса на СКУД, обладающие функционалом «мобильного доступа». Происхождение этого термина связано с продуктом «Mobile Access» [87], представленным компанией HID Global в 2014 г. и позволяющим смартфонам взаимодействовать со специализированными считывателями iClass SE [88] и multiClass SE [88]. Продукт подразумевает возможность построения инфраструктуры, в которой для получения доступа к материальным или информационным ресурсам в качестве идентификатора используются мобильные устройства.

Широкая вариативность встроенных модулей и предлагаемых функций современных гаджетов на сегодняшний день позволяет организовать взаимодействие со СКУД несколькими способами.

Использование GSM-модуля

Поскольку подавляющее большинство рассматриваемых устройств может быть использовано в качестве сотового телефона, одним из наиболее распространненых подходов является использование GSM-модуля. При этом управляющее воздействие на ИУ или УПУ передается в случае поступления телефонного звонка с определенного номера, закрепленного в БД СКУД за субъектом доступа. Длительность процедуры звонка, а также возможные сбои в работе операторов связи ограничивает область применения подобных систем объектами, характеризующимися невысокой проходимостью посетителей. Примером реализации может выступать шлагбаум, установленный на автомобильной парковке и управляемый по звонку с телефонов субъектов доступа, зарегистрированных в БД. Помимо низкой скорости функционирования подобная схема обеспечивает низкий уровень защищенности, поскольку единственным защитным механизмом выступает «белый список» номеров абонентов.

Использование мобильных приложений

Поскольку современные мобильные устройства функционируют на базе ОС, предоставляющих возможности установки различного ПО, то одним из вариантов применения их в СКУД является разработка специализированных клиентских приложений, которые позволяют субъекту доступа самостоятельно передавать команды на контроллер. Очевидное преимущество данной схемы заключается в возможности исключения УС и как следствие, упрощении архитектуры СКУД. В то же время необходимо гарантировать наличие связи между смартфоном и контроллером. В целях обеспечения масштабируемости и снижения стоимости системы в подобных решениях зачастую используются облачные сервисы в сети Интернет. Учитывая, что предоставление возможности субъекту доступа самостоятельно передавать команды контроллеру по сути приводит к наделению пользователей правами операторов системы, вышеуказанные преимущества такой схемы взаимодействия обусловливают возникновение целого спектра угроз безопасности для СКУД в целом.

Использование технологий штрих-кодирования

Еще одним вариантом взаимодействия смартфонов и ПИАФ является применение линейных или двумерных штрихкодов, которые отображаются на экране мобильного устройства [89]. Для реализации такой схемы в СКУД должны применяться УС видеозаписывающего класса, способные фиксировать изображение, преобразовывать его и передавать информацию на контроллер в некотором формате. Примером таких СКУД являются системы регистрации билетов. Учитывая, что такие системы характеризуются низкой степенью защищенности ключевой информации, считываемой с экрана мобильного устройства, применение подобных решений весьма ограничено.

Использование технологии Near Field Communication

Появившаяся более 10 лет назад технология беспроводной высокочастотной связи малого радиуса действия NFC (Near Field Communication) [90, 91] является одним из наиболее популярных современных трендов в области использования использования мобильных устройств в качестве идентификатора. Среди основных характеристик этой технологии следует отметить:

- бесконтактную передача данных;
- минимальное возможное расстояние для передачи данных;
- возможность обмена информацией с другими устройствами или пассивными метками;
- низкую стоимость решения;
- низкое энергопотребление.

Подробные технические характеристики приведены в Приложении 3 [91, 92]. Спецификация интерфейса NFC не предоставляет практических сценариев использования. В отличие, например, от Bluetooth, NFC является только базой, а непосредственные решения обеспечиваются дополнительным ПО. С одной стороны, это открывает широкие возможности для разработчиков, а с другой – является для них же проблемой при обеспечении взаимодействия разных приложений и устройств. Существенным недостатком, ограничивающим массовое применение данной технологии, является отсутствие соответствующего модуля у большого количества существующих смартфонов.

Использование Bluetooth Low Energy

Bluetooth с низким энергопотреблением (Bluetooth Smart, BLE) [93] представляет собой спецификацию ядра популярной беспроводной технологии, наиболее существенными достоинствами которой при использовании в СКУД являются:

- сверхмалое энергопотребление (среднее, пиковое и в режиме простоя);
- отсутствие необходимости предварительного сопряжения устройств;

- использование алгоритма шифрования данных AES [94] с ключом размером в 128 бит, что является достоинством при использовании в СКУД.

Подробные технические характеристики приведены в приложении 3 [95]. Кроме того, в отличие от NFC, BLE полноценно поддерживается как в мобильных ОС Android (с версии 4.4), так и Apple iOS.

Представленный выше анализ существующих решений и подходов по использованию мобильных средств связи в качестве идентификаторов доступа позволяет сделать вывод, что существенными недостатками предлагаемых на рынке продуктов являются:

- необходимость применения конкретного оборудования, поддерживающего фиксированный набор технологий передачи данных;

- отсутствие в их составе документации, регламентирующей организацию ПИАФ в СКУД ММПЛ;

- проприетарность алгоритмов и используемого ПО.

Другими словами, в настоящее время отсутствует методическое обеспечение, которое позволяло бы организовывать процедуры идентификации и аутентификации в ММПЛ, исходя из требований политики безопасности объекта, а не из особенностей применения конкретного оборудования.

Автором предлагается подход, который заключается в использовании мобильных устройств в качестве идентификаторов для ПИАФ электронных проходных. При этом отличительной особенностью подхода является отсутствие зависимости от конкретных протоколов и технологий передачи данных.

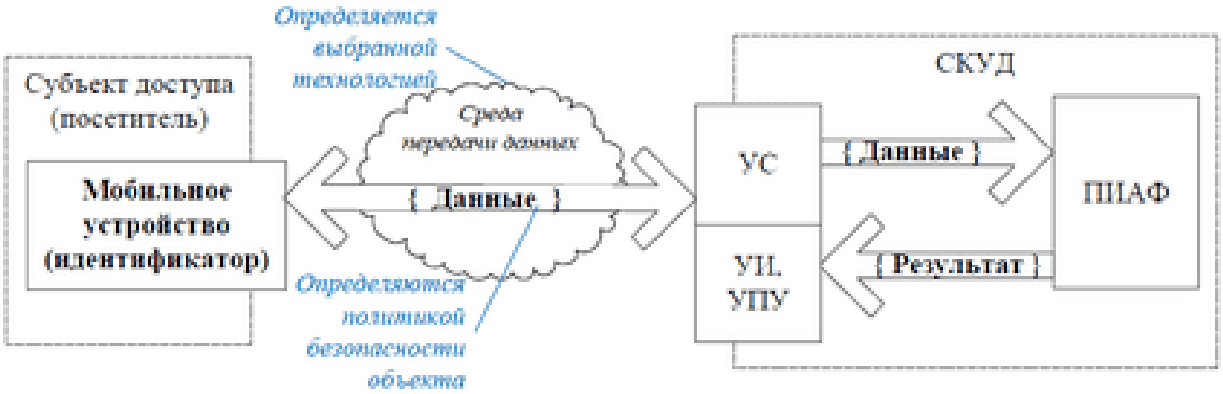


Рисунок 3.2 – Концепция предлагаемого подхода

Концепция подхода (рис. 3.2) предполагает использование мобильных устройств в качестве идентификаторов доступа. При этом владельцу объекта предоставляется возможность самостоятельного определения набора идентификационных данных, а также технологии их передачи в СКУД, что обеспечивает гибкость подхода и возможность масштабирования ПИАФ.

В соответствии с этапами, представленными на рис. 3.3, для организации ПИАФ в СКУД ММПЛ необходимо проанализировать бизнес-процессы и ТСОБ, уже имеющиеся на объекте. На основе полученных сведений следует определить механизмы аутентификации (если они требуются), технологии передачи данных и алгоритмы шифрования. Полученные данные представляют собой техническое задание, которое используется для построения (реализации) ПИАФ.

Рисунок 3.3 – Использование подхода

Как упоминалось ранее, большинство современных смартфонов характеризуются существенными вычислительными возможностями и оборудованы высокоскоростными модулями связи для работы в сети Интернет. Мобильные ОС предоставляют широкий спектр инструментов для разработки и установки специализированных прикладных программ, а также возможность использования различных средств шифрования. Благодаря этому современное мобильное устройство связи может содержать в себе несколько идентификаторов («виртуальных пропусков»), используемых для доступа на различные объекты.

Данные факты являлись предпосылками теоретических изысканий автора и легли в основу вышеизложенного подхода к идентификации и аутентификации в СКУД ММПЛ. Ниже перечислены основные преимущества применения мобильных устройств в качестве идентификаторов доступа.

Во-первых, в современном мире мобильные устройства связи нередко рассматриваются в качестве идентификационных признаков, поэтому вероятность обнаружения субъектом доступа потери или кражи такого идентификатора значительно выше, чем в случае с классическими электронными пропусками. Также вероятность возникновения случаев передачи смартфона другим лицам для прохода на объект значительно ниже, чем в случае с картами доступа. Кроме того, современные телефоны нередко обладают функционалом по информированию владельца о координатах устройства.

Во-вторых, одно мобильное устройство может быть использовано в качестве идентификатора для нескольких объектов. Многие организации имеют офисы по всему миру, в которых могут использоваться различные СКУД. В случае командирования сотрудников зачастую требуются гостевые пропуска. Применение мобильных устройств, помимо очевидного снижения финансовых затрат на обслуживание идентификаторов, позволяет обеспечить единую политику безопасности на всех объектах компании.

В-третьих, наличие в мобильных устройствах высокоскоростных модулей подключения к сетям передачи данных обусловливает возможность оперативного удаленного управления идентификаторами, включая их выдачу, изменение конфигурации и отзыв в режиме реального времени. Не менее важной является возможность удаленной регистрации субъектов доступа в БД СКУД (без предварительного посещения объекта) с применением механизмов верификации личности.

В-четвертых, используемые в мобильных устройствах технологии транспорта идентификационных данных обеспечивают возможность использования в местах, требующих значительного расстояния между УС и идентификатором. Например, шлагбаум на въезде в паркинг, гараж и т.д. Кроме того, в большинстве случаев возможна скрытая установка УС (без вмешательства в интерьер помещения) для защиты от вандализма.

В-пятых, современные мобильные ОС позволяют использовать алгоритмы шифрования и средства ЭП, что обеспечивает высокий уровень безопасности самого устройства, выступающего в качестве идентификатора. Как правило, приложения работают в изолированной программной среде (Sandbox) [96], исключающей возможность доступа или изменения данных другими программами, а базовые функции автоматической блокировки экрана обеспечивают возможность введения дополнительного фактора аутентификации посредством применения паролей различных видов.

Помимо вышеперечисленных преимуществ предложенный подход обусловливает широкие конвергенционные возможности, поскольку мобильное устройство, применяемое в качестве идентификатора, может быть использовано как для разграничения доступа к физическим объектам, так и для доступа к электронным ресурсам предприятия. Примером может являться аутентификация при доступе к беспроводным сетям связи, VPN [97], корпоративным порталам и т.д.

Алгоритм аутентификации на основе применения QR-кодов

Применение метода генерации OTP «запрос-ответ», предполагающего асинхронный режим работы с двусторонним обменом данными между клиентом и сервером, не находит практической реализации в случае использования технологий штрихового кодирования.

Недостатком всех остальных методов, работающих в синхронном режиме, является возможность рассинхронизации OTP-токена [57] и сервера. Несмотря на то что в режиме «синхронизация по времени» используется не дискретное значение, а некоторый допустимый запас времени для процедуры аутентификации, часы программного генератора одноразовых паролей могут уйти вперед или отстать. Это может быть вызвано умышленным изменением времени устройства либо сбросом настроек по причине разряда аккумулятора или сбоя ПО.

Умышленная смена временных параметров является той процедурой, которую пользователи обычно выполняют очень редко, тем более что современные средства связи обладают функционалом синхронизации времени с NTP серверами. В случае полной разрядки телефона внутренние часы мобильного устройства обычно поддерживаются резервным элементом питания.

В режимах «только ответ» или «синхронизация по событию» сбой аутентификации может привести к «отставанию» сервера от мобильного устройства субъекта. В качестве примера можно рассмотреть случай, когда пользователь запустил приложение-аутентификатор, но по каким-то причинам не воспользовался им и завершил работу. В качестве решения подобных проблем можно генерировать варианты OTP сразу для нескольких

событий. Приведенный ниже алгоритм предполагает формирование одноразовых паролей в режиме «синхронизация по времени».

Регистрация мобильного устройства субъекта доступа

Для регистрации идентификатора необходимо перейти в режим синхронизации времени и поднести мобильное устройство к камере-считывателю (в случае удаленной регистрации – к web-камере субъекта доступа). ПО, установленное на мобильном устройстве, отобразит на дисплее QR код с текущим временем. После этого сервер вычислит и сохранит значение смещения между своими часами и временем, переданным мобильным устройством. Как и в случае с синхронизацией времени, процедура согласования ключа *b* происходит путем считывания строки с дисплея устройства QR кода. В случае использования удаленной регистрации посетителей необходимо обеспечить шифрование передаваемых данных между клиентом и сервером.

Аутентификация

Процедура аутентификации в такой системе будет выполняться согласно схеме, представленной на рис. 3.5 [99].

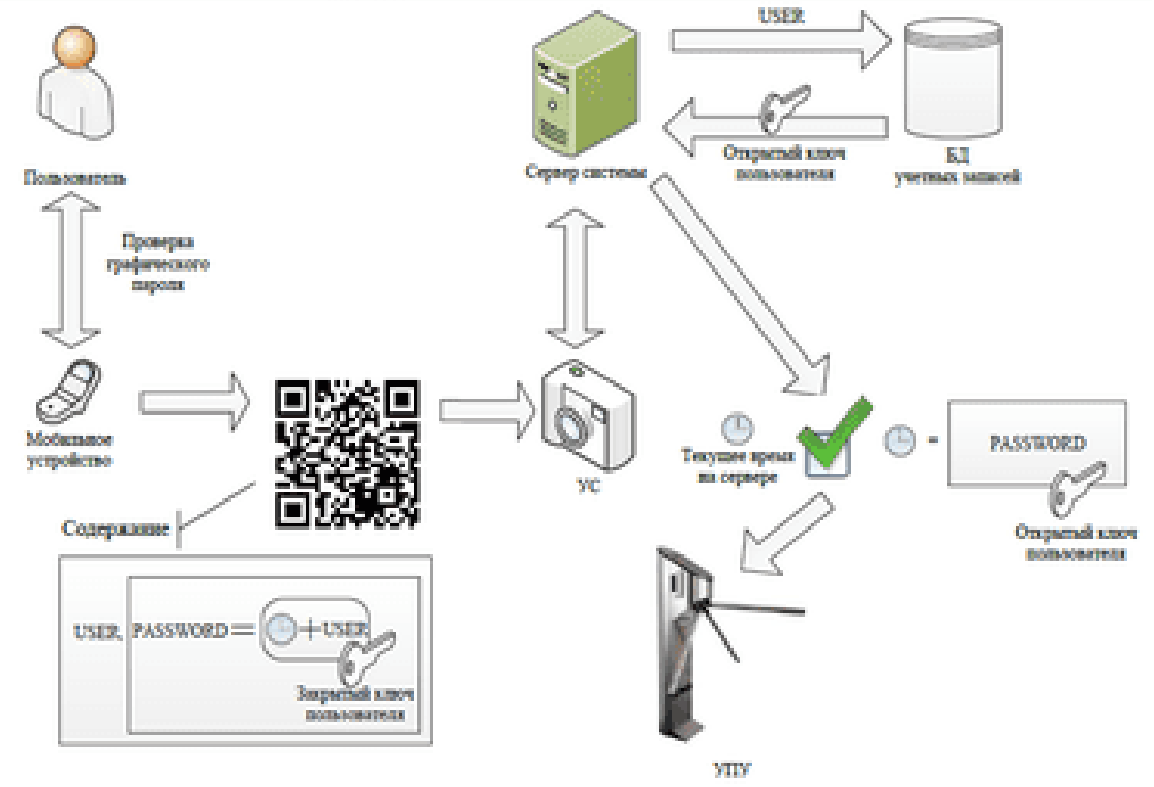


Рисунок 3.5 – Схема аутентификации в режиме передачи QR кодов

Получение одноразового QR кода для аутентификации можно описать выражением *F*:

$$F(u,g,b,n)= Q(u^{\times}E_{R[g*b*n),l]}(u \oplus t)) \qquad (3.1)$$

где *u* – идентификатор пользователя; *g* – значение графического пароля; *b* – ключ, хранящийся в ПЗУ мобильного устройства; *n* – набор значений аппаратных характеристик мобильного устройства; *t* – текущая временная метка; *l* – набор требований, предъявляемых к секретному ключу в используемом алгоритме шифрования; $^{\times, *, \oplus \vdash}$ операции конкатенации строк с использованием разделителей; *Q(x)* – операция преобразования сообщения *x*

в QR код; *Ek*(*x*) – операция шифрования сообщения *x* на ключе *k*; *R*(*x*,*y*) – операция формирования секретного ключа из входной строки *x* в соответствии с набором требований *y*, предъявляемых к секретному ключу используемым алгоритмом шифрования; *H*(*x*) – операция хэширования строки *x*.

Алгоритм, определяющий порядок формирования одноразового QR кода для аутентификации, представлен в виде блок-схемы на рис. 3.6.

Перед началом операции формирования нового QR кода счетчик *tint*, принимает значение 0. В качестве условия, свидетельствующего о необходимости генерации нового QR кода, выступает процедура сравнения счетчика с параметром «временя жизни одноразового пароля *tint-live*», определяемого оператором системы.

УС системы после считывания QR кода передает его контроллеру для декодирования (операция *Q`*) и дальнейшего разбиения (операция *Array*) на массив *m*, состоящий из двух элементов – идентификатора пользователя *u* и значениея OTP (выражение (3.2)):

F`( m`) = Array [Q`(m`)] (3.2)

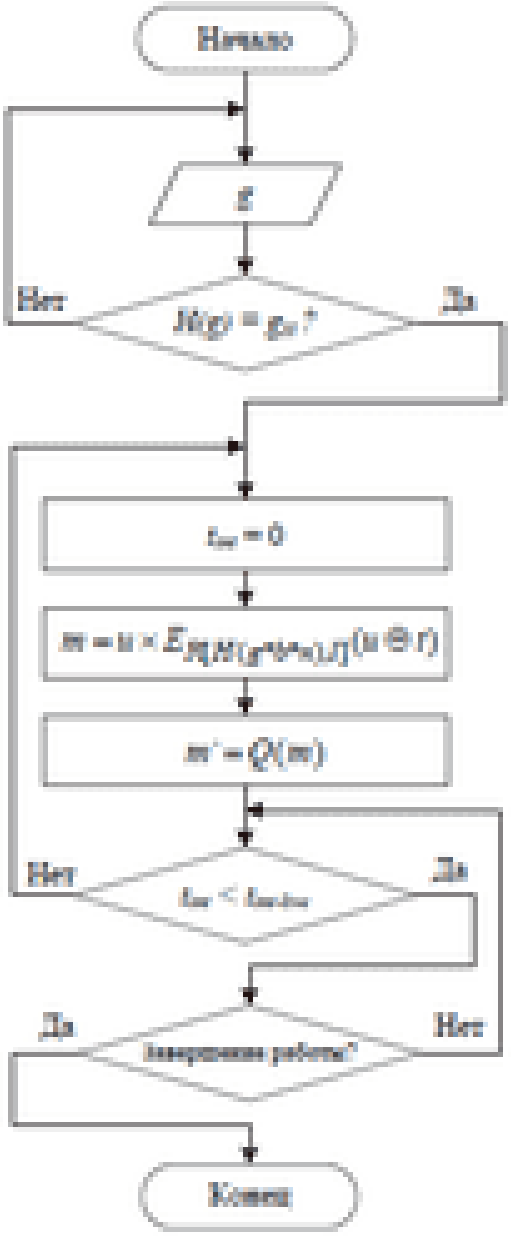


Рисунок 3.6 – Блок схема формирования QR кода для аутентификации

Далее выполняется процедура расшифрования и последующее выделение значение времени генерации пароля:

$$P(D_{S(m[0])}(m[1]))=t_2, \quad (3.3)$$

где *t2* – полученное значение времени генерации одноразового пароля; *m*[0] – идентификатор пользователя; *m*[1] – значение пароля; *P(x)* – операция выделения значения времени генерации OTP; *Dk(x)* – операция расшифрования сообщения *x* на ключе *k*; *S(x)* – операция получения ключа *k* по пользовательскому идентификатору *x* из БД.

В случае успешного расшифрования значения OTP система сверяет текущее время *t* со временем генерации одноразового пароля *t2*. Если разница не превышает установленную границу *tint-live*, то в УИ или УПУ

передается управляющее воздействие на пропуск субъекта доступа (рисунок 3.7).

Рисунок 3.7 – Блок –схема выполнения процедуры аутентификации контроллером СКУД

Алгоритм аутентификации на основе применения NFC-меток

Технология NFC предусматривает три режима работы [100]:

1) Card Emulation mode – эмуляция бесконтактной смарт-карты;

2) Peer-to-Peer – пиринговый режим обмена между двумя активными устройствами;

3) Reader-Writer mode – режим чтение / запись.

До 2013 года информацию для осуществления NFC-транзакций в режиме эмуляции бесконтактной карты в мобильных устройствах можно было хранить только на специальном защищенном чипе (Secure element, SE [101]). Данный элемент может быть размещен либо конструктивно в корпусе телефона, либо на специальной карте формата MicroSD, либо в SIM-карте формата (принцип SIM centric NFC [91, 101]). С выпуском мобильной ОС

Android 4.4 KitKat [102] компания Google предоставила возможность взаимодействия NFC-контроллера не только с SE, но и с обычным приложением в телефоне. Данное решение под названием Host Card Emulation (HCE) оказалось очень популярным среди приложений для мобильных устройств (карты лояльности, бонусные системы, скидочные купоны и т.д.). Все приведенные в п. 3.1.1 решения по реализации функционала мобильного доступа в СКУД в случае применения технологии NFC основаны на использовании HCE.

В данной работе автором предложен алгоритм для использования технологии NFC в СКУД, использующий альтернативный режим Peer-to-Peer и основанный на применении OTP с методом генерации одноразовых паролей «Запрос-ответ» (рис. 3.8). Применение подобного алгоритма существенно расширяет возможность практической реализации функционала мобильного доступа в СКУД, не требуя от мобильного устройства поддержки технологии HCE.

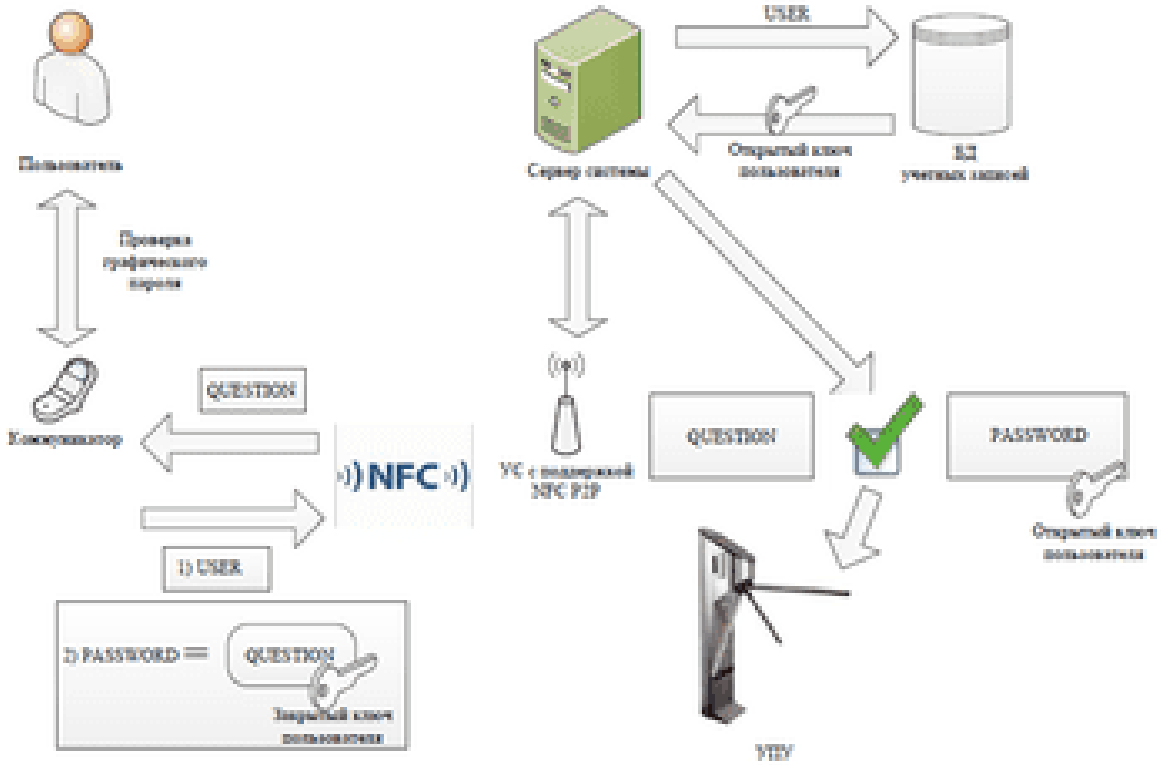


Рисунок 3.8 – Схема аутентификации в режиме NFC

В данной схеме динамическим составляющим является случайный вопрос *a*, который сервер генерирует на каждое обращение пользователя. Именно этот вопрос подвергается шифрованию пользовательским устройством, в дальнейшем позволяя осуществить проверку аутентичности субъекта доступа.

Алгоритм процедуры аутентификации можно представить в виде блок-схем, представленных на рис. 3.9, 3.10.

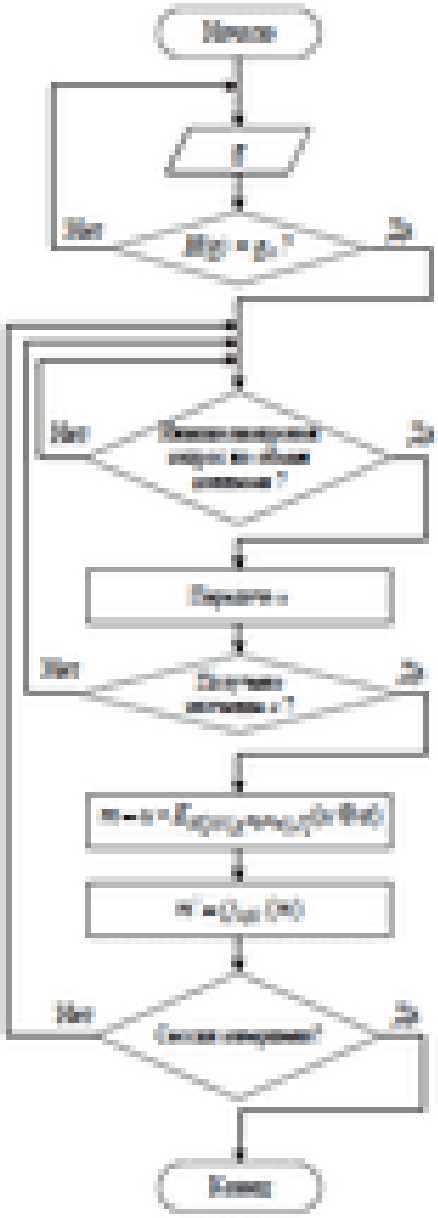


Рисунок 3.9 – Блок – схема аутентификации в режиме NFC для мобильного средства субъекта доступа

1.4. Программное и информационное обеспечение микропроцессорных карт

Программное и информационное обеспечение Микропроцессорных карт. Программное и информационное обеспечение микропроцессорных карт включает
операционную систему карты, прикладные программы (приложения) и идентификационную информацию.

1. Операционная система смарт-карт (COS - Card Operating System) - упрощённый аналог «настоящих» операционных систем. На самом деле представляет собой набор микрокодов, записанных в ROM, которые принято разделять на ядро COS и команды COS. Термин «операционная система»

взят по аналогии, чтобы сохранить строгость терминологии в отношении аналогичных функций компьютеров и смарт-карт, хотя операционные системы современных смарт-карт, и особенно многофункциональных, по своей сложности всё больше напоминают полноценные компьютерные операционные системы.

Ядро COS - это набор микрокоманд, выполняемых при установлении сеанса связи с ридером, обеспечивающих работу с аппаратурой карты и интерфейс с внешним устройством. Ядро COS является аналогом загрузочных программ обычной ОС.

Команды COS - это специфицированный набор процедур, также, как правило, для повышения быстродействия реализованных аппаратно, предназначенный для работы с прикладными программами и данными, обеспечения механизма безопасности карты. Выделяют следующие группы команд COS:

- команды работы с данными прикладных программ: создание и удаление файлов, чтение данных из файлов различных форматов: структурированных и бесструктурных, выбор директории (для иерархической файловой системы), позиционирование указателя, обновление записей в
файлах различных типов, добавление записи к файлу;
- команды обеспечения безопасности - их часто выделяют в отдельную группу, называемую “системой безопасности” смарт-карты (SF - Security Features). Это довольно большая группа команд, предназначенных для обеспечения безопасности данных, хранящихся и обрабатываемых картой, защиты файлов, операций с ключевой и идентификационной информацией. В SF входят следующие команды: выработка случайного числа, смена PIN владельца, аутентификация внешнего устройства, аутентификация смарт-карты для внешнего устройства, загрузка файла ключей, закрытие файла, объявление файла недействительным и снятие этой блокировки, аутентификация пользователя, блокировка аутентификации пользователя, изменение условий доступа к файлам и др. Если карта поддерживает работу с асимметричными криптографическими алгоритмами, то к перечисленным командам добавляется группа команд для работы с открытыми ключами: внутренняя и внешняя аутентификация с открытым ключом, загрузка ключа, чтение ключа, вычисление хэш-функции, проверка ключа, проверка цифровой подписи;
- команды обслуживания карты предназначены для выполнения вспомогательных операций:
- блокировка карты при неоднократных попытках несанкционированного доступа, транспортная блокировка, форматирование файловой системы и др.;
- специфические команды, зависящие от назначения карты. Например, если карта используется для электронной торговли, в ОС могут быть включены специальные команды увеличения/уменьшения величины, хранящейся в структуре данных, представляющей счёт владельца карты, простановка криптографических отметок («штампов», зависящих от времени). Это

помогает значительно повысить скорость выполнения часто повторяющихся операций;

- некоторые другие команды: чтение файла статистики транзакций, выбор коммуникационного протокола, изменение скорости обмена с внешним устройством и т.п.

2. Прикладное ПО (AS - Application Software), как правило, записывается на смарт-карту процессинговым центром (см. гл. 4). Частично прикладное ПО может быть записано в PROM-память, частично - в EEPROM. ПО смарт-карты должно разрабатываться в связи с ПО устройств, осуществляющих работу с ними, работа всех компонентов ПО должна быть согласована между собой и с форматами файлов, хранящихся на смарт-карте. Отдельное ПО необходимо для авторизации карты в центре авторизации, т.е. записи на неё информации о владельце, эмитенте, порядкового номера, ключей, идентификаторов и т.п.

Программирование памяти смарт-карт осуществляется с помощью специальных

инструментальных пакетов, включающих транслятор текстов программ с языков высокого уровня и ассемблера в коды микропроцессора смарт-карты, а также программный эмулятор микропроцессорной системы смарт-карты для отладки и тестирования ПО.

В качестве примера можно привести интегрированное программное средство ASE Soft для разработки приложений для смарт-карт, разработанное фирмой ALADDIN Software Security R.D. ASE Soft включает в себя библиотеку функций API высокого уровня, API нижнего уровня, криптографического API, примеры приложений для смарт-карт на языках C++, Visual Basic, Java. Пакет обеспечивает безопасный обмен данными между смарт-картой и приложением, работающим на терминале, использование различных типов смарт-карт и протоколов в одном приложении с использованием как высокоуровневых процедур программных библиотек, так и набора команд ISO 7816, возможность работы на одном компьютере с несколькими кард-ридерами и др. возможности. ASE Soft API содержит функции, реализующие алгоритмы шифрования, алгоритмы работы со случайными числами, антиотладочные и антитрассировочные механизмы. Пакет поддерживает многие известные языки программирования высокого уровня: C, C++, Pascal, Clipper, Visual Basic и язык Assembler. Интегрированная программная среда и эмулятор МП-системы позволяют наблюдать заполнение ячеек памяти, пароли доступа к данным, последовательность микрокоманд процессора. Полный комплект разработчика включает в себя также специальный кард-ридер ASE Drive с программируемой EEPROM-памятью и набор смарт-карт с различным объёмом памяти, протоколами обмена, встроенными специальными функциями и аппаратными алгоритмами шифрования.

3. Идентификационная информация записывается на карту в центре авторизации. В банковских системах запись осуществляется на рабочем месте оператора банка, кассира или другого лица, выдающего клиенту карту.

С помощью специального кард-ридера на карту записывается PIN-код владельца, который в запечатанном конверте передаётся клиенту. Клиент впоследствии по своему желанию может сменить его.

6.4. Элементы программного обеспечения смарт-карт

В настоящее время в широком обращении находятся два вида смарт-карт:

- Смарт-карты с фиксированным набором команд, построенном на основе набора команд стандарта ISO 7816-4;
- Смарт-карты с возможностью послеэмиссионного программирования

Смарт-карты первого вида создаются для применения в определенной сфере. Набор команд , соответствующий рекомендациям стандарта ISO 7816-4, предназначен обеспечить внутреннюю файловую систему карты функциями хранения и передачи данных, а также моделью безопасности, в которой элементы приложения внутри и вне карты могли бы аутентифицировать друг друга. Модель безопасности также позволяет владельцу карты аутентифицировать себя в ней и подтвердить, что карта действует от имени подлинного владельца.

Смарт-карты второго вида позволяют проектировать наборы команд специально для приложения, которое будет загружено в карту. Добавляя все больше таких приложений, одну и ту же смарт-карту можно использовать для широкого круга задач.

6.4.1. Структура программного обеспечения смарт-карт

Программное обеспечение смарт-карт можно разделить по месту расположения на две взаимосвязанные части:

- Программное обеспечение основного (хост-) компьютера, связанного со смарт-картой;
- « внутреннее» программное обеспечение самой смарт-карты

Программное обеспечение хост - компьтера составляет наибольшую часть программного обеспечения смарт-карт. Иногда это программное обеспечение называют «внешним» (по отношению к смарт-карте) программным обеспечением. Оно написано для персональных компьютеров и рабочих станций, работающих с существующими смарт-картами, обеспечивает доступ к этим картам и объединяет эти карты в более масштабные системы. В состав ПО хост компьютера входят:

- Прикладные программы;
- Программы системного уровня, поддерживающие подключение считывателей (ридеров) смарт-карт к хост-платформе;
- Программы системного уровня, обеспечивающие применение конкретной смарт-карты, нуждающейся в поддержке прикладных программ.

Кроме того, программное обеспечение хост- компьютера содержит приложения и утилиты, необходимые для поддержки управления инфраструктурной смарт-карт.

Программное обеспечение самой смарт-карты, часто называемое «внутренним» ПО, включает в себя программы, которые выполняются на самой смарт-карте. В зависимости от масштабов решаемой прикладной задачи в качестве этого ПО выступает операционная система смарт-карты, утилита или приложения.

Широкое распространение получили серийные смарт-карты, поставляемые производителями, и смарт-карты, создаваемые главными их эмитентами, такими как банковские ассоциации, телекоммуникационные компании и национальные правительства. Операционные системы таких распространенных смарт-карт реализует характерный набор команд (обычно 20 или 30), на которые откликается смарт-карта. Программное обеспечение хост-компьютера посылает команды операционной системы смарт-карты, которая исполняет их на своем процессоре и возвращает результаты.

Для многих приложений достаточно использовать смарт-карты общего типа с их общепринятым внутренним программным обеспечением, при этом для карты не нужны специальные программы. Там, где требуется программное обеспечение специально для данного приложения, его обычно пишут либо на ассемблере, зависящем от архитектуры чипа микропроцессора, размещенного на смарт-карте, либо на языке более высокого уровня с последующей интерпретацией непосредственно на карте или компилированием в ассемблер карты и загрузкой в нее.

Программное обеспечение смарт-карты полезно подразделять по назначению на прикладные и системные программы.

Прикладные программы используют вычислительные возможности и емкость памяти смарт-карт так же, как и любого другого компьютера, и не занимаются защитой данных смарт-карт. *Системные программы,* напротив,

используются непосредственно для улучшения свойств смарт-карты по сохранению данных и обеспечению защиты.

Прикладные программы хост-компьютера используют смарт-карту для альтернативного выполнения одних и тех же функций (например, когда ключ шифрования или медицинскую запись предпочтительнее хранить на смарт-карте, чем в файле жесткого диска локального компьютера или в базе данных где-либо на сервере).

Программное обеспечение хост-компьютера использует уникальные и присущие только данной смарт-карте способности обработки и хранения информации, посылая карте данные и команды и получая от нее данные и результаты вычислений.

Прикладные программы карты обычно используются для настройки существующей серийной смарт-карты для конкретного применения и обрабатываемой информации и для переноса прикладных программ с хост-компьютера в саму карту.

Это может делаться в интересах производительности для ускорения взаимодействия между хост-компьютером и картой в интересах безопасности для защиты внутренней части системы.

Системные программы смарт-карты пишутся на машинном языке низкого уровня конкретного чипа смарт-карты и приминяются для расширения или замены основных функции смарт-карты.

. . Программное обеспечение хост-компьютера обычно бывает написано на одном из языков программирования высокого уровня, имеющихся на ПК и рабочих станциях (например, C, C++, Java), и поддерживает связь с коммерчески доступными библиотеками и драйверами устройств для организации доступа к считывателям смарт-карт и самим картам, вставляемым в них. Программное обеспечение карт обычно пишется на таком языке программирования, как Java, или на языке более низкого уровня, например на Forth или на языке ассемблера.

Перед началом работы со смарт-картой программа хост-компьютера должна выполнить две задачи:
1. во-первых, она должна убедиться, что смарт-карта, с которой она работает, является подлинной;
2. во-вторых, она должна убедить смарт-карту в своей подлинности. Пока не будет установлено взаимное доверие, смарт-карта не должна выполнять никаких команд хост-компьютера.

Следует отметить, что выполнение приложения, для которого создавалась смарт-карта (хранение электронных денег или создание цифровой подписи), обычно составляет лишь малую часть всего взаимодействия между операционной системой смарт-карты и программой хост-компьютера на командном языке.

Взаимодействие «внешнего» и «внутреннего» программного обеспечения смарт-карты

Оба вида программного обеспечения смарт-карты - внешнее программное обеспечение хост-компьютера и внутреннее программное обеспечение самой карты фундаментально различны по своему назначению и ориентации. Программное обеспечение хост-компьютера может применяться для многих типов карт и обычно использует сведения о картах различных типов, о владельцах и эмитентах карт.

Программное обеспечение карты сосредоточено на содержимом конкретной карты, обеспечивает вычислительное обслуживание приложений по доступу к этому содержимому и защищает это содержимое от множества приложений, способных некорректно обратиться к нему.

Программное обеспечение хост-компьютера соединяет смарт-карты и пользователей в единую систему. Например, программа, работающая в банкомате, использует смарт-карты, вставляемые клиентами банка, для идентификации клиентов и затем представляют доступ клиентам к их банковским счетам. Программное обеспечение хост-компьютера приспосабливает свой отклик, основываясь на конкретном виде представленной ему карты.

Программное обеспечение смарт-карты реализует свойства защиты данных и их обработки, а также политику безопасности определенной смарт-карты. Например, программа, выполняющаяся на смарт-карте, не может выдать номер счета, хранящийся на карте, пока ей не будет предоставлен правильный PIN-код. Программа, выполняющаяся на смарт-карте, может рассчитать цифровую подпись с применением закрытого ключа, хранимого на карте, но не может создать закрытый ключ сама. Программы, работающие в смарт-карте, обеспечивают безопасный, авторизованный доступ к данным, хранящимся в ней. Этим программам известны лишь содержимое конкретной карты и объекты за ее пределами (например, люди, компьютеры, терминалы и т.п.), пытающиеся получить доступ к этому содержимому.

В отличие от большинства обычных компьютерных программ, полагающихся на поддержку служб из окружения ПК и рабочих станций,программы смарт-карты изначально предполагают, что окружение, в котором они находятся, враждебно и не заслуживает доверия. Программа смарт-карты доверяет только самой себе. Все, что находится снаружи программы, должно доказать, что оно заслуживает доверия, перед тем, как программа смарт-карты начнет взаимодействовать с ним. Смарт-карты не доверяют хост-компьютерам, с которыми они связываются, пока они не получат убедительные доказательства, при этом хост-компьютеры также изначально не доверяют картам, взаимодействующим с их считывателями.
Безопасность программного обеспечения смарт-карт основана на использовании криптографической защиты информации. Криптография используется для аутентификации объектов системы, таких как пользователи, карты и терминалы, и для шифрования связи смарт-карты с внешним миром. Криптографические ключи хранятся в файлах карты, а криптографические алгоритмы и протоколы выполняются в программном обеспечении карты. Криптографические функции, встроенные в смарт-карту для удовлетворения собственных требований безопасности, могут быть также использованы для выполнения функций безопасности в других системах. Защита, обеспечиваемая смарт-картой, значительно увеличивает безопасность системы.
Одной из первых выполняемых смарт-картой задач при ее активации является аутентификация внешних по отношению к ней объектов, прежде всего личности того, кто вставил карту в терминал, и того терминала, в который ее вставили, а также в аутентификации самой себя по отношению ко всем этим объектам. Процедура аутентификации может быть просто демонстрацией владения общим секретом, подобным четырехзначному PIN-коду, или же представлять собой более сложный процесс демонстрации способности шифровать предложенное сообщение, называемое запросом, при помощи заранее согласованного алгоритма шифрования и секретного ключа.

Если на любом этапе процесса смарт-карта обнаруживает, что имеет дело не с тем объектом, за который он себя выдавал, то вся дальнейшая связь с таким объектом блокируется. Регистрация этих неудавшихся попыток может вестись на смарт-карте, и после определенного количества безуспешных попыток аутентификации

Глава 2. Легковесная криптография.

2.1. Криптографические примитивы.

Развитие криптографии, начиная с последней четверти XX века, в основном определялось и в ближайшем будущем будет определяться развитием Интернета и интернет технологий. При этом определяющим на ближайшие годы направлением развития Интернета будет так называемый *Интернет Вещей* (Internet of Things, IoT), принятый комиссиями Европарламента и Совета Европы в качестве магистрального пути развития информационных и интернет технологий 1.

1 www.internet-of-things-research.eu/documents.htm

Это направление характеризуется переходом от Интернета Персональных Компьютеров к Интернету Вещей.

В последние годы наряду с традиционными интернет-устройствами, такими как персональные компьютеры, ноутбуки, смартфоны, стали появляться устройства бытовой техники, транспорта, а также различные датчики, имеющие доступ в Интернет. Это явление получило название «Интернет вещей»3. Интернет вещей представляет собой беспроводную самоконфигурирующуюся сеть между объектами типа бытовых приборов, транспортных средств, различных сенсоров и датчиков, а так же меток радиочастотной идентификации (**R**adio **F**requency **ID**entification, RFID).

3 Термин «Интернет вещей» был, по-видимому, впервые введен в обращение в 1999 г. Кельвином Эштоном (Kelvin Ashton), в то время главным технологом Массачусетского Технологического Института.

Стремительное развитие указанных технологий делает чрезвычайно актуальными вопросы, связанные с их информационной безопасностью. Особенно актуальной становится задача эффективной реализации алгоритмов защиты информации, обеспечивающих конфиденциальность и целостность данных.

Основу такой безопасности должны образовывать криптографические методы защиты информации. И основным средством обеспечения информационной безопасности в мире Интернета Вещей является так называемая «*легковесная криптография*» (lightweight cryptography, LWC). Аксель Пошманн, известный специалист по Lightweight Cryptography, характеризует ее словами «As light as a feather, and as hard as dragon-scales» – «Легка, как пух и прочна как чешуя дракона» 7.

7 см. Толкиен, «Властелин Колец»

«Как свет, как перо, и так же тяжело, как дракон-весы».

Было описание Бильбо Бэггинса для Мифрила, легендарного материала в J.R.R. Толковский знаменитый роман «Властелин колец» [228]. Однако это также подходит к описанию легкой криптографий. С одной стороны, легкая криптография нацелена на очень легкие реализации, которые практически «легки как перо», но, с другой стороны, не слишком уступают уровню безопасности. Фактически, один из основных аспектов легкой криптографии заключается в использовании компромиссов эффективности безопасности, присущих реализации криптографических алгоритмов. «Жесткие, как драконские шкалы» - хороший парафраз для этого аспекта, потому что он подчеркивает, что существуют достаточные уровни безопасности (размер ключа 80 бит).

Массовое развертывание повсеместных устройств имеют много преимуществ, таких как снижение логистических издержек, более высокая степень детализации процесса, оптимизированные цепи поставок или услуги, основанные на местоположении, например, технология RFID[1] считается способствующей технологией для Интернета вещей. В основном, метки RFID состоят из транспондера и антенны и могут удаленно получать данные с устройства RFID или устройства считывания. Метки RFID можно разделить на пассивные и активные устройства: активные метки обеспечивают собственный источник питания (батареи), тогда как пассивные метки исключительно полагаются на энергию сигнала несущей, передаваемого считывателем устройство. В результате пассивные устройства RFID не только значительно дешевле, но также требуют меньше размера микросхемы и имеют более длительный жизненный цикл [74].

Существует множество рисков, присущих широко распространенным вычислениям:

Многие предусмотренные приложения чувствительны к безопасности, такие как беспроводные сенсорные сети для военных, финансовых или автомобильных приложений. Благодаря широкому присутствию встроенных компьютеров, безопасность - это проблема, потому что потенциальный ущерб вредоносных атаки также увеличиваются. Стаджано видит эти три основные проблемы, которые определяют, ubicomp будет успешным или нет [210]:

(1) недостаточная безопасность

(2) решение неправильных проблем (глобальная инфраструктура с открытым ключом)

(3) предотвратить ubicomp, чтобы стать повсеместным наблюдением.

Еще одной активной областью исследований является предотвращение подделки товаров.

Для этой цели желательно использовать метки RFID как криптографические жетоны, т.е. в протоколе ответа на вызов. В этом случае тег должен быть в состоянии для выполнения защищенного криптографического примитива. Бесконтактные микропроцессорные карты [192], которые способные выполнять криптографические алгоритмы,

являются не только дорогостоящими и, следовательно, не обязательно подходит для массового производства, но также привлекает много тока. Высокая, неоптимальная мощность потребление микропроцессора обычно может быть обеспечено только системами с близкой связью, т.е. должно быть обеспечено короткое расстояние между считывающим устройством и устройством RFID [74]. Лучший подход заключается в использовании микросхемы RFID на заказ, которая состоит из схемы приемника, блока управления, то есть конечный автомат, некоторая энергозависимая или энергонезависимая память и криптографический примитивный. Эти криптографические примитивы должны быть оптимизированы для жесткой мощности и области ограничения, с которыми сталкиваются недорогие пассивные RFID-метки.

Термины «*низкоресурсная*», «*малоресурсная криптография*» или «легкая крипторафия», точно отражает суть дела.

Легкая криптография - относительно молодая научная подполе, расположенная на пересечении электротехники, криптографии и информатики и фокусируется на новых проектах, адаптациями или эффективными реализациями криптографических примитивов и протоколов.
Типичными ограничениями, встречающимися в низкоресурсной криптографии, являются: для аппаратной реализации – размер микросхемы, потребляемая энергия, время, затраченное на исполнение программы; для программной реализации – размер программного кода, размер оперативной памяти, время, затраченное на исполнение программы. В зависимости от конкретных условий применения разрабатываемого средства важной может оказаться такая характеристика, как ширина полосы рабочих частот канала связи.

Каждый проектировщик в области низкоресурсной криптографии должен стремиться найти баланс между безопасностью, ценой и производительностью (рис. 1). Обычно легко оптимизировать любые две из трёх целей разработки – безопасность и стоимость, безопасность и производительность, или стоимость и производительность; однако, очень тяжело оптимизировать эти три параметра одновременно.

Рис.1

Для реализации надежного и высокопроизводительного оборудования может быть достигнута с помощью конвейерной, устойчивой к боковым каналам архитектуры, что приводит к высокой площади и, следовательно, высокие затраты. С другой стороны, можно разработать безопасную, недорогую аппаратную реализацию с недостатком ограниченной производительности.

Метрика оценки отличается: для программных реализаций, мы сравниваем требования к ОЗУ и ПЗУ и требуемое количество тактовых циклов. Для аппаратных реализаций, мы фокусируемся на требуемом размере микросхемы и количестве тактовых циклов.

Другое различие между симметричными и асимметричными шифрами, поскольку последние предлагают больше функциональной безопасности и, следовательно, сценариев приложений. Симметричные шифры служат главным образом для проверки целостности сообщений, аутентификации объекта, и шифрование, а асимметричные шифры дополнительно предоставляют преимущества управления ключами и неотказуемость. Асимметричные шифры являются вычислительно гораздо более требовательным, как на аппаратном, так и на программном обеспечении.

Криптографические примитивы (cryptographic primitives) являются основным криптографическим инструментарием, который обеспечивает выполнение тех или иных криптографических сервисов. Они подразделяются на примитивы с секретным ключом (симметричные примитивы), примитивы с открытым ключом (асимметричные примитивы) и бесключевые примитивы.(рис. 2.)

Рис.2

Для примитивов с секретным ключем криптографическая стойкость обеспечивается секретностью ключа, который должен оставаться неизвестным для всех участников информационного процесса, не имеющих соответствующих полномочий. В этот класс входят такие примитивы, как симметричные шифры (которые, в свою очередь, подразделяются на блочные шифры и поточные шифры), ключевые хэш-функции, называемые также кодами проверки подлинности сообщения или имитовставками (keyed hash function, **M**essage **A**uthentication **C**ode – MAC) а также криптографические генераторы псевдослучайных последовательностей.

Примитивы с открытым ключом используют пары ключей (*e,d*) – (ключ шифрования, ключ расшифрования) или (открытый ключ, секретный ключ). Криптографическая стойкость обеспечивается секретностью, секретного ключа, который должен оставаться неизвестным для всех участников информационного процесса, не имеющих соответствующих полномочий. В этот класс входят такие примитивы, как протоколы выработки и согласования ключей, асимметричные шифры (шифры с открытым ключом), схемы цифровой подписи и некоторые другие.

Бесключевые примитивы. В этот класс входят примитивы, не использующие ключей. Такие примитивы используются в таких криптографических сервисах, как аутентификация и обеспечение целостности информации. В этот класс входят такие примитивы, как однонаправленные подстановки

(one-way permutation), хэш-функции, бесключевые хэш-функции или коды обнаружения модификации информации (unkeyed hash function, **M**odifi cation **D**etection **C**ode – MDC), генераторы случайных последовательностей и проч.
Безусловно важнейшей характеристикой криптоалгоритма является его криптостойкость. Однако анализ стойкости и «легковесных» вариантов классических алгоритмов и специально разработанных легковесных алго ритмов принципиально ничем не отличается от анализа криптоалгоритмов общего вида. Он не имеет какой-то особой специфики и проводится обычным для современного криптоанализа образом – осуществляется проверка стойкости алгоритма относительно известных на сегодняшний день методов криптоанализа (линейный криптоанализ, разностный криптоанализ, корреляционный анализ и т.д.). Для средств низкоресурсной криптографии очень важна их стойкость по отношению к анализу по побочным каналам (Side Channel Attacks), что объясняется условиями их эксплуатации. Большинство алгоритмов (и облегченные варианты классических и специально разработанные «легковесные» алгоритмы) демонстрируют практическую стойкость.

Основным характеристиками реализации криптографического алгоритма являются сложность реализации и скорость работы. Скорость – очень важная характеристика во многих применениях – в свою очередь зависит не только от частоты работы процессора, но и от размеров микросхемы (в случае аппаратной реализации) поскольку криптографические примитивы, как правило, весьма удобны для распараллеливания. В свою очередь сложность характеризуется размером микросхемы в GE [9] (в случае аппаратной реализации) или размером программного кода в байтах и размером требуемой оперативной памяти (в случае программной реализации).

---

9 Площадь микросхемы обычно измеряется в μm2, однако этот параметр сильно зависит от используемых технологий и библиотек стандартных ячеек (standard cell library). Для того, чтобы иметь возможность сравнивать микросхемы, изготовленные по разным технологиям, размеры принято измерять в условных логических элементах (Gate Equivalent – GE). За один условный логический элемент принимается площадь, занимаемая элементом NAND с двумя входами.
При этом «ультралегкой» (ultra-lightweight) называется реализация, требующая менее 1000 GE, «низкостоимостной» (low-cost) – требующая не более 2000

Симметричные алгоритмы, в основном, служат для шифрования, проверки целостности сообщений, аутентификации, в то время как асимметричные алгоритмы используются в основном для управления ключами и обеспечения неотказуемости. Асимметричные алгоритмы требуют значительно большего объема вычислений по сравнению с симметричными как при аппаратной так и при программной реализации.
Так оптимизированный асимметричный алгоритм на эллиптических кривых (ECC) выполняется от 100 до 1000 раз медленнее, чем стандартный

симметричный шифр и имеет на два-три порядка более высокое энергопотребление.

Среди алгоритмов с открытым ключом есть три семьи, имеющие практическое значение: ECC, RSA (Ривст-Шамир-Адлеман) и дискретных логарифмов.

Table 2. Comparison of software implementations of ciphers.

Cipher	Key size (bits)	Block size (bits)	Encryption (cycles/ block)	Throughput at 4 MHz (Kbps)	Decryption (cycles/ block)	Relative throughput (% of AES)	Code size (bytes)	SRAM size (bytes)	Relative code size (% of AES)
Hardware-oriented block ciphers									
DES	56	64	8,633	29.6	8,154	38.4	4,314	0	152.4
DESXL	184	64	8,531	30.4	7,961	39.4	3,192	0	112.8
Hight	128	64	2,964	80.3	2,964	104.2	5,672	0	200.4
Present	80	64	10,723	23.7	11,239	30.7	936	0	33.1
Software-oriented block ciphers									
AES	128	128	6,637	77.1	7,429	100.0	2,606	224	100.0
IDEA	128	64	2,700	94.8	15,393	123.0	596	0	21.1
TEA	128	64	6,271	40.8	6,299	53.0	1,140	0	40.3
SEA	96	96	9,654	39.7	9,654	51.5	2,132	0	75.3
Software-oriented stream ciphers									
Salsa20	128	512	18,400	111.3	NA	144.4	1,452	280	61.2
LEX	128	320	5,963	214.6	NA	287.3	1,598	304	67.2

* IDEA: международный алгоритм шифрования данных; TEA: крошечный алгоритм шифрования; SEA: Масштабируемый алгоритм шифрования.

ECC считается наиболее привлекательным для из-за меньшей длины операндов и относительно меньших вычислительных требовании. ECC принимается на коммерческой основе и также был принят стандартизирующими органами таких как Американский национальный институт стандартов (ANSI), IEEE, Международной организацией по Стандартизация (ISO), Стандарты для эффективного Криптографическая группа (SECG) и Национальный институт стандартов и технологий (NIST).

2.2. Шифрование с низкой задержкой.

Наиболее активная и наиболее продуктивная деятельность по разработке низкоресурсных криптоалгоритмов происходила в области алгоритмов блочного шифрования.

Развитие этой области низкоресурсной криптографии шло по двум направлениям:

- эффективная реализации известных алгоритмов блочного шифрования
- разработка новых блочных шифров, ориентированных на оптимальную реализацию на микропрограммном или аппаратном уровне.

В качестве «точки отсчета» для сравнения тех или иных реализаций блочных шифров приведем различные реализации «эталонного» алгоритма блочного шифра AES.

Идентифицирую несколько известных облегченных блочных шифров, чтобы обеспечить реализацию с малой задержкой. Изучая этот набор шифров в контексте шифрования с малой задержкой, наша работа дает первые

результаты в поле. Разрабатываем структуру, которая анализирует поведение криптографических примитивов с низкой задержкой по следующим аспектам:

• Минимальная достижимая латентность.
• Его влияние на размер схемы.
• Его влияние на потребление энергии.

Облегченные блочные шифры дают хорошие результаты с точки зрения стоимости реализации даже в полностью развернутой реализации. Помимо латентности в качестве нашей основной цели, мы рассматриваем область кремния как очень важный фактор в практических реализациях алгоритмов шифрования, и поэтому мы ограничиваемся легкими блочными шифрами, но включают в себя эталонный шифр. Также ограничиваем набор шифрами хорошо изученной структурой SPN.

Это приводит к следующему списку из семи легких блочных шифров SPN: aes [10, 28], klein [14], led [16], mcrypton [25], mini-aes [8], noekeon [9], present [5]. Приведем краткое описание каждого шифра.

AES [10,28], разработанный Daemen и Rijmen в 1997 году, стал не только стандартом NIST, но и самым используемым блочным шифром в настоящее время.

Aes - это итегрированный блочный шифр с размером блока 128 бит и тремя возможными длинными ключами 128, 192 и 256 бит.

Рассмотрим 128-битную ключевую версию, состоящую из 10 раундов. Размер слова составляет 8 бит, т. е. элементы данных рассматриваются как элементы поля GF (28). Каждый раунд aes состоит из следующих операций: SubBytes, ShiftRows, MixColumns и AddRoundKey. Операция SubBytes (S-слой) определяется как одновременное применение S-Box (инверсия в GF (28)) для каждого элемента состояния. Перестановочный слой (P-слой) состоит из операций ShiftRows и MixColumns. Операция ShiftRows определяется как одновременное левое вращение строки i состояния по i позициям. Операция MixColumns предварительно умножает каждый столбец состояния на матрицу MDS, определенную над GF (28). KeySchedule выводит круглый ключ из секретного ключа, применяя один раз S-Box и некоторые простые линейные операции. AddRoundKey XOR - круглый ключ к текущему состоянию.

NOEKEON [9] - это 128-битный блочный шифр с 128-битным ключом, Noekeon - это самоинверсивный шифр с битовой средой и может считаться предшественником современных легких блок-шифров. Он имеет 16 раундов, и каждый из них состоит из следующих операций: тета, Pi1, Gamma и Pi2. Операция Gamma представляет собой инволютивное нелинейное отображение (S-слой), в котором S-боксы работают независимо от 32 4-битных кортежей. Pi1 и Pi2 выполняют простые циклические сдвиги. Theta - линейное отображение, которое первым XOR является рабочим ключом к состоянию, а затем выполняет простое линейное преобразование состояния. Поэтому Theta действует частично как AddRoundKey и вместе с Pi1 и Pi2

формирует P-слой шифра. KeySchedule очень прост - так называемый рабочий ключ выводится из секретного ключа, а затем XORed в состояние в каждом раунде. Для процедуры шифрования рабочий ключ просто равен секретному ключу. Самонапряженное свойство шифра имеет большие преимущества, когда шифрование и дешифрование необходимо реализовать в одной и той же схеме.

MINI-AES [8], или малый вариант aes. Рассмотрим 10-раундовый мини-айс с размером блока 64 бит, длиной ключа 64 бита и размером слова 4 бита. Основная разница между аэсом и версией мини-ай, заключается в том, что S-поле и матрица MDS определены над полем GF (24). Поэтому выбранный экземпляр mini-aes можно рассматривать как облегченную версию шифрования aes.

MCRYPTON [25] - это 64-разрядный блочный шифр, поддерживающий три разных длины ключа (64, 96 и 128 бит). Каждый раунд mcrypton состоит из следующих операций: NonLinear Substitution γ, Колонка-бит-бит Перестановка π, Транзакция столбца-строка τ и добавление ключа σ. Операция γ (S-слой) состоит из 16 подстановочных подстановок, использующих четыре 4-битных S-блока (S0, S1, S2, S3, все эквивалентные для инверсии в GF (24) и такие, что S2 = S-1 0 и S3 = S-1 1). P-слой состоит из операций π и τ. Операция π является инволюционным битовым умножением матрицы. Операция τ просто переносит состояние и, следовательно, является инволюцией. KeySchedule прост и состоит из двух этапов: генерация круглого ключа посредством нелинейного преобразования S-box и обновление ключевой переменной через простое вращение. Операция σ XOR включает круглый ключ к состоянию. Независимо от длины ключа, mcrypton всегда использует 12 раундов со слегка дифференцированным KeySchedule. Дешифрование и шифрование могут совместно использовать большую часть круглых операций и что KeySchedule допускает прямой вывод последнего круглого ключа.

ПРИСУТСТВУЮЩИЙ [5], разработанный Богдановым и др., один из самых известных легких блок-шифров. Он был принят в качестве стандарт в ИСО / МЭК 29192-2. 31-раундовый шифр имеет размер блока 64 бит и поставляется с 80-битным или 128-битным ключом. Каждый раунд настоящего состоит из следующих операций: sBoxLayer, pLayer и AddRoundKey. SBoxLayer определяется как одновременное применение очень легкого 4-битного S-Box для каждого полубайта состояния. PLayer - простая побитовая перестановка. KeySchedule вращает ключевую переменную, XOR - константу и применяет Sbox к ключевой переменной. AddRoundKey XOR - это 64 наиболее значимых бита ключевой переменной для состояния. PLayer обеспечивает довольно медленный уровень шифрования, что приводит к значительному количеству раундов.

KLEIN [14] - блок-шифр с фиксированным 64-битным блочным размером и переменной длиной ключа 64, 80 или 96 бит. Каждый раунд шифрования состоит из следующих операций: SubNibbles, RotateNibbles,

MixNibbles и AddRoundKey. Операция SubNibbles (S-слой) определяется как одновременное применение инволютивного 4-битного S-Box для каждого элемента состояния. P-слой состоит из операций RotateNibbles и MixNibbles. Операция RotateNibbles поворачивает состояние на два байта влево. MixNibbles совпадает с операцией aes MixColumns, т.е. Предварительно умножает каждый столбец состояния на матрицу MDS, определенную над GF (28). KeySchedule выводит круглый ключ из секретного ключа, применяя два S-Box и некоторые простые линейные операции. AddRoundKey XOR - это круглый ключ к состоянию. klein-64/80/96 использует 12/16/20 раундов соответственно.

Светодиод [16], разработанный Го, Пейрином, Пошманном и Робшоу в 2011 году, является одним из самых последних легких шифров. Это 64-битный блочный шифр на основе nibble с двумя вариантами, использующими 64-битные и 128-битные ключи. Каждый раунд состоит из следующих операций: AddConstants, SubCells, ShiftRows и MixColumnsSerial. Каждые 4 раунда применяется операция AddRoundKey. SubCells (S-layer) повторно использует настоящий S-box и применяет его к каждому 4-битовому элементу состояния. MixColumnsSerial использует матрицу MDS, определенную над GF (24) для линейного дифференциала, которая подходит для компактной последовательной реализации, поскольку она может быть представлена как мощь очень простой двоичной матрицы. AddConstants XOR - константа состояния в каждом раунде. ShiftRows работает, вращая строку i состояния массива по положению i ячейки слева. AddConstants, ShiftRows и MixColumnsSerial образуют P-слой шифрования. 64-битный ключевой вариант состоит из 32 раундов, в то время как 128-битный ключевой вариант состоит из 48 раундов. Шифр не имеет KeySchedule, то есть один и тот же ключ XORed в состоянии с использованием AddRoundKey, один раз каждые 4 раунда.

Полученный набор блок-шифров представляет собой широкий спектр строительных блоков для S-слоя, P-слоя и ключевого графика. Таким образом, aes является (единственным) байт-ориентированным блочным шифрованием (то есть байтом S- и P-слоев) с P-слоем MDS; noekeon имеет S-слой на основе nibble, основанный на битах P-слой, и он является самоинверсивным битовым шифрам; mini-aes - это нитевидный шифр (т. е. S- и P-слои на основе Nibble) с P-слоем MDS; mcrypton имеет S-слой на основе nibble, бит-мутную матрицу для P-слоя с конкретным ключевым расписанием; присутствует S-слой на основе nibble и очень простая битовая перестановка для P-слоя; klein имеет S-слой на основе nibble и байт-основанный P-слой MDS (эквивалент aes); в конечном счете, приведено блочное шифрование, ориентированное на nibble (то есть S- и P-слои на основе Nibble) с P-слоем MDS и без ключевого графика.

Klein и aes имеют одну и ту же матрицу MDS; имеют один и тот же S-слой; мини-аэи и светодиоды имеют различные матричные матрицы MDS; и

S-слой mini-aes и mcrypton близки (эквивалентные эквиваленты) друг к другу.

Наша цель - оценивать проекты с наименьшей достижимой задержкой, мы в основном фокусируемся на архитектуре с 1 циклом и 2 циклами. Более конкретно, архитектура с 1 циклом представляет собой полностью развернутую архитектуру, для которой требуется один тактовый цикл для ее выполнения. Аналогично для архитектуры с 2 циклами требуется два тактовых цикла для выполнения своих вычислений. Поскольку термин lowlatency подразумевает небольшое количество тактовых циклов для выполнения алгоритма не оцениваем архитектуры, требующие трех или более тактовых циклов.

Затем мы различаем только архитектуры шифрования (ENC) и архитектуры шифрования / дешифрования (ENC / DEC). Некоторые из реализованных шифров выигрывают от присущих им сходств между данными шифрования и расшифровки данных. Предоставляем диаграммы для более компактной, но все же немного более медленной реализации, которая разделяет данные. На рисунке 2 показаны все оцениваемые архитектуры, сообщаем результаты только для архитектур (ENC / DEC). Результаты для архитектур, поддерживающих только шифрование, приведены в Приложении B.

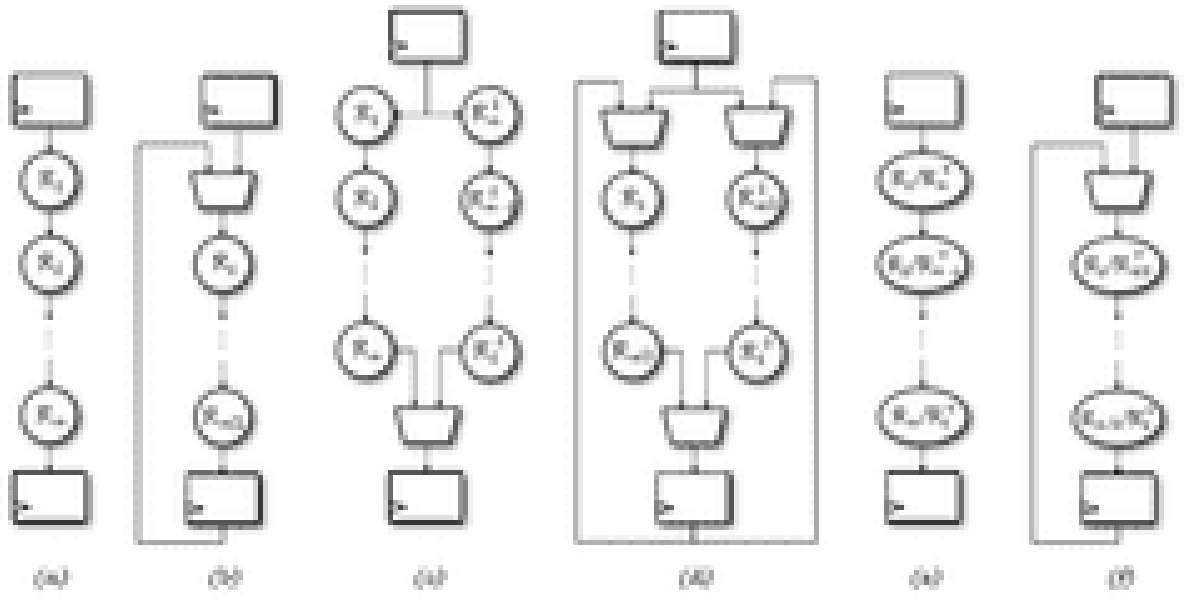


Рис.2: Шесть оцениваемых архитектур: (a) основанный только на 1 цикл, только ENC. (b) 2-циклов, только ENC. (c) 1-циклов, ENC / DEC. (d) основанный на 2 циклах, ENC / DEC. (e) основанный на 1 цикл, ENC / DEC, общий дата тат. (f) двух цикличный, ENC / DEC, общий путь данных.

Все оцененные шифры имеют размер блока 64 бит, за исключением aes и noekeon, которые имеют 128-битный размер блока. Некоторые шифры поддерживают разные длины ключей, поэтому мы оцениваем 128-битный

ключ; 64-битный, 80-битный и 96-битный ключ klein; 64-битный и 128-битный светодиоды; 64-битный, 96-битный и 128-битный ключ mcrypton; 64-разрядные мини-айы; 128-битный ключ noekeon; и 80-битный и 128-битный ключ. Большинство полученных результатов будет сильно коррелировано с количеством раундов шифрования, рисунок 3 визуализирует эту метрику

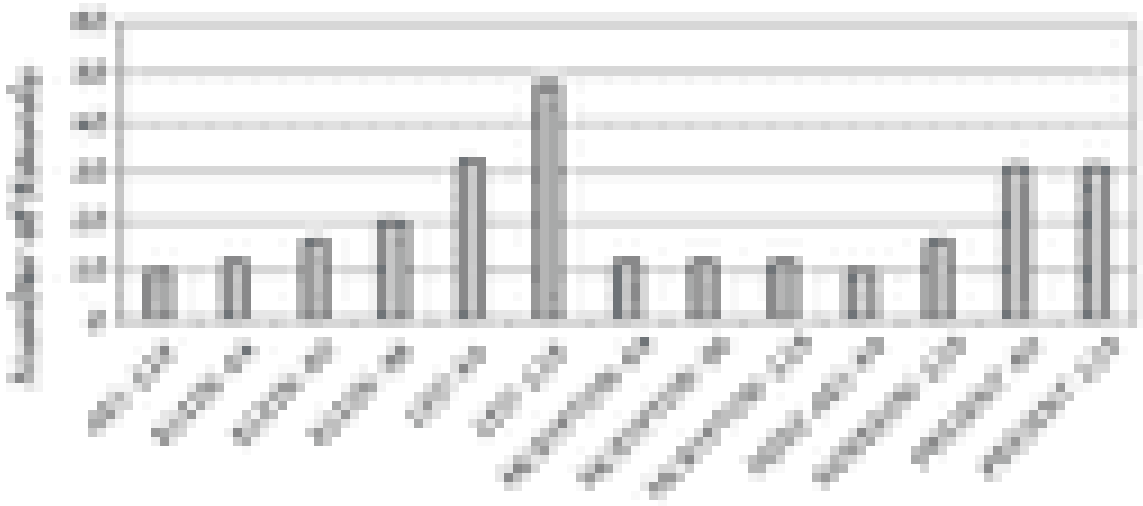


Рисунок 3: Количество раундов проверенных шифров.

Определяем латентность как меру времени, необходимого для того, чтобы определенный проект завершил определенную (вычислительную) задачу. Вычислительная задача определяется как шифрование одного блока сообщений, а задержка вычисляется как:

$$\text{Задержка} = N \cdot tcp,$$

где N - количество тактовых циклов, необходимых для шифрования одного блока сообщений, а tcp - критический путь схемы. Чтобы подчеркнуть разницу между задержкой и пропускной способностью, указываем, что задержка действительно зависит от присущих свойств криптографического алгоритма, в то время как пропускная способность - нет, ее можно просто увеличить, используя общие методы обработки сигналов, такие как конвейерные и параллельные вычисления, на рисунке 4 показана минимальная достижимая латентность для модуля ENC / DEC всех оцененных шифров. noekeons-128 обозначает реализацию noekeon с общим datapath для шифрования и дешифрования, и он четко обозначен серым цветом, чтобы отделить его от других конструкций. На рисунке показано, что в целом существует лишь незначительное преимущество архитектуры с 1 циклом, основанной на двух циклах, но минимальная латентность получается с использованием архитектуры с 1 циклом. Конструкции, которые показывают наивысшую производительность, мини-ай и mcrypton. Клейн-64, который на 30% медленнее, является третьим лучшим кандидатом. Самая низкая производительность - от led-128, что более чем в 5 раз медленнее, чем

у mini-aes. aes, достигает 70% более медленного критического пути, чем мини-ай.

Латентность некоторых конструкций, т. е. Klein и led, зависит от длины ключа, тогда как mcrypton, это не так ? Это напрямую связано с количеством раундов, которое в случае увеличения клеев и светодиодов увеличивается для больших длин ключей, в то время как оно остается постоянным для mcrypton (рисунок 3).

В Приложении В приведены результаты для архитектур, основанных на ENC (рисунок 12). Результаты показывают, что производительность некоторых конструкций, например, klein-64 и mini-aes, безусловно, ухудшается, когда путь дешифрования встроен в дизайн. Предоставляем рисунок 5, где показано средняя задержка за раунд каждого шифра как для архитектуры ENC, так и для ENC / DEC. Легко видеть, что расшифровка datapath для aes, klein и mini-aes значительно медленнее, чем шифрование. В меньшей степени это также относится к led, mcrypton и present. noekeon - единственный шифр, который не обладает этим свойством. Также наблюдаем четкую корреляцию между средней латентностью за раунд и сложностью раунда.

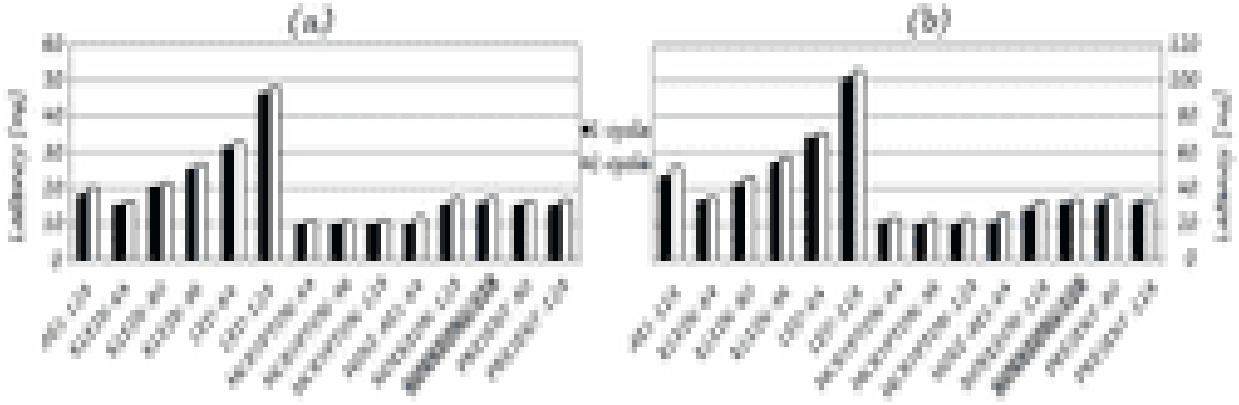


Рис.4: Минимальная задержка [ns] для модуля ENC / DEC: (a) Ограничение по времени. (b) Без ограничений.

С другой стороны, безусловный дизайн показывает несколько неожиданный результат. Его неограниченная архитектура ENC (рис. 12, приложение B), кажется, медленнее, чем ее архитектура ENC / DEC. Этот результат объясняется тем, что при отсутствии ограничений инструмент синтеза оптимизирует конструкции для области, в то время как время менее важно. Тем не менее, когда timeconstreined, инструмент синтеза делает значительным e ff ort для оптимизации времени, и поэтому существующая архитектура ENC / DEC становится медленнее, чем ее архитектура с поддержкой ENC.

Отношение латентности неограниченного и временного конструкций варьируется от 2,63 для aes (ENC / DEC) до 1,30 для ноэкеона (ENConly, рис.12), что иллюстрирует эластичность латентности конструкции

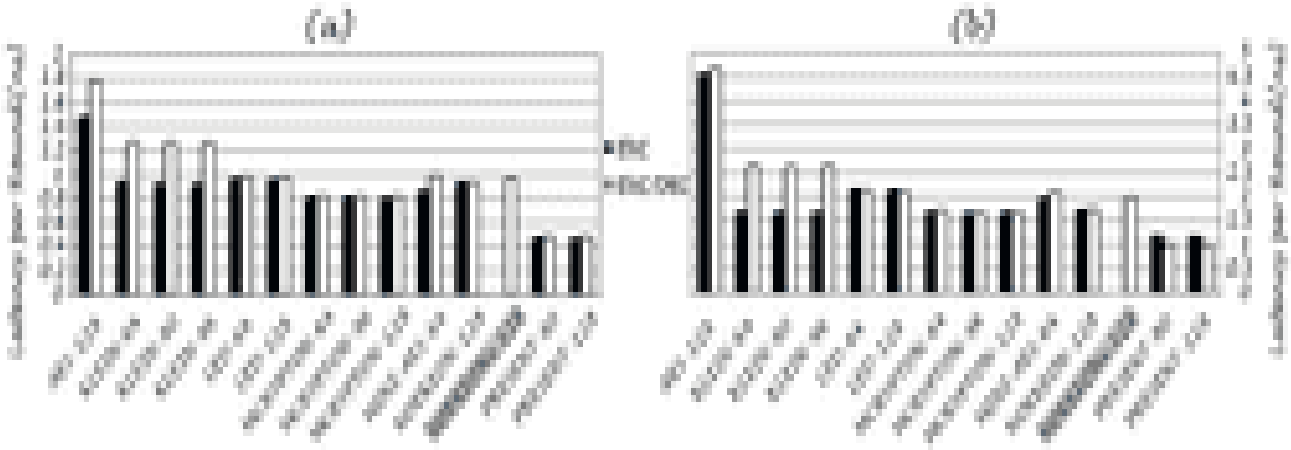


Рисунок 5: Средняя задержка [ns] за раунд: (a) Ограничение по времени. (b) Без ограничений.

Предоставляем результаты для размера схемы всех оцененных вариантов шифрования. Во-вторых, расскажем о распределении площади за раунд, где наблюдаем несколько интересных результатов. Область выражается в единицах эквивалентности затвора (GE), представляющих относительный размер схемы по сравнению с простым 2-входным логическим номером NAND.

На рисунке 6 показана область для архитектур ENC / DEC. В отличие от графиков задержки, преимущество для архитектуры с 2 циклами ясно: архитектуры на основе двух циклов потребляют примерно половину площади архитектур на основе 1 цикла. Также наблюдаем значительную корреляцию между количеством шифрованных раундов и размером схемы. mini-aes и mcrypton снова показывают лучший результат, за которым следует примерно 25% -ная реализация klein-64. присутствующий приходит как следующий с около 60% накладными расходами. Наибольший размер схемы показан aes, что более чем в 9 раз больше, чем мини-айс. Из легких шифров led-128 потребляет самую большую площадь и более чем в 4 раза больше, чем мини-айс.

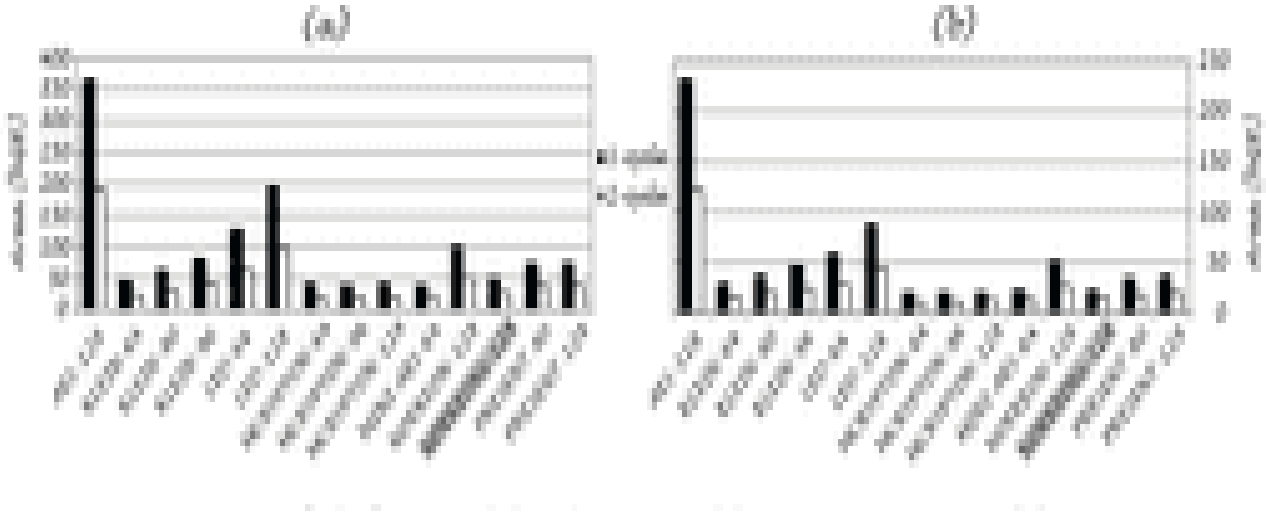


Рис.6: Минимальная площадь [kGE] для модуля ENC / DEC: (a) Ограничение по времени. (b) Без ограничений

В noekeon, где из-за присущего им сходства пути передачи данных для шифрования и дешифрования могут использоваться совместно. Видно, что реализация с общим каналом данных имеет значительную экономию площади (около 50%), но не влияет на латентность так много, на 5% увеличивается (Фиг.4). Подобное наблюдение, все еще в меньшей степени, справедливо для mcrypton. Когда реализован общий путь передачи данных, это приводит к 30% экономии области с примерно 20% накладных расходов по сравнению с результатами, изображенными на рисунке 4.

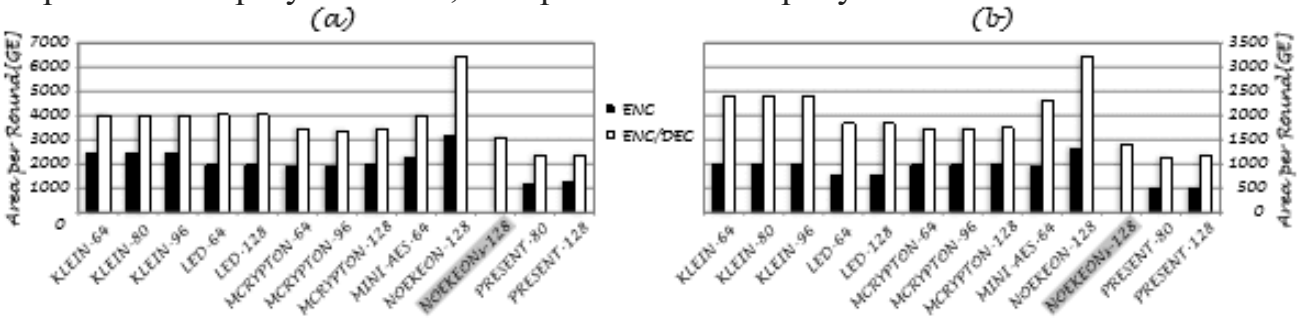


Рис.7: Средняя площадь [GE] за раунд: (a) Ограничение по времени. (b) Без ограничений,

Рисунок 7, который иллюстрирует среднюю площадь за раунд для каждого шифра (за исключением aes), показывает, что у настоящего есть самый маленький раунд среди всех шифров, что неудивительно, так как его круг состоит из S-слоя и очень легкого P-слоя. P-слои других шифров включают более сложные операции, такие как умножение с матрицей MDS для мини-аэсов. Средняя площадь за раунд ноэкеона относительно велика. Это связано с размером блока 128 бит; в два раза больше, чем у других шифров. Из этого следует как количество раундов шифрования, так и их

сложность оказывают существенное влияние на производительность аппаратного обеспечения.

Существует ряд наблюдений за распределением площади за круг, используем пример klein-80 (рис.8); хотя одно и то же наблюдение относится к большей или меньшей степени для большинства оцененных шифров. Во-первых, из-за более высокой сложности дешифрования критический путь проходит через datapath дешифрования, поэтому он становится значительно большим, чем дата-шифрование шифрования при ограничении времени. noekeon - единственный шифр, освобожденный от этого eff ect, в то время как eff ect едва заметен в случае led. Когда ограничения расслаблены, этот эффект естественным образом исчезает.

Еще одно замечание, как для ограниченных по времени, так и для неограниченных реализаций, а также для всех оцененных шифров, - это значительно меньшая площадь, занятая в последние несколько раундов развернутого дизайна. Например, при ограниченных по времени реализации klein-80 последний раунд более чем на 25% меньше по размеру, чем самый большой раунд. Для всех других шифров эта разница всегда остается выше 20%. Это явление объясняется тем, что логические ворота, используемые в последних раундах, требуют значительно более низкой прочности, поскольку они приводят меньше логики, чем средние раунды, и поэтому могут быть меньше.

Третье замечание (рисунка 8) - это заметное колебание в области в первых 13 раундах временной реализации klein-80. Это, эффект, введенный инструментом синтеза и вызвано введением значительного количества ячеек памяти для усиления распространения сигнала по всей комбинационной сети схемы, которая происходит периодически, несколько раундов друг за другом.

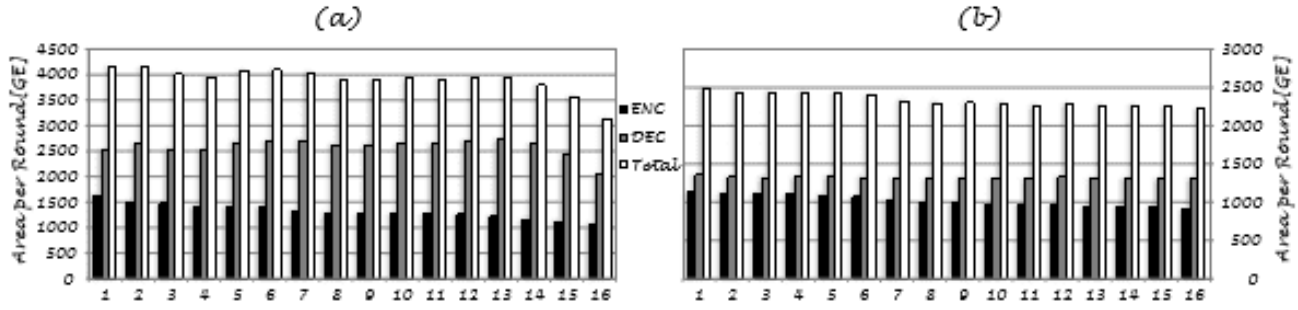


Рис.8: Распределение площадей [GE] за раунд klein-80: (a) Ограничение по времени. (b) Без ограничений.

Соотношение между размером ограниченного временем и неограниченным рисунками охватывает диапазон от 1,66 для klein-80 до 2,22 для noekeon. Это соотношение определяет эластичность области дизайна и указывает на накладные расходы в области, необходимой для достижения наименьшего возможного критического пути конструкции.

Предоставляем графики для продукта временной области, поскольку это часто используемый критерий для выбора окончательной реализации. На рисунке 9 показан продукт временной области для архитектуры ENC / DEC.

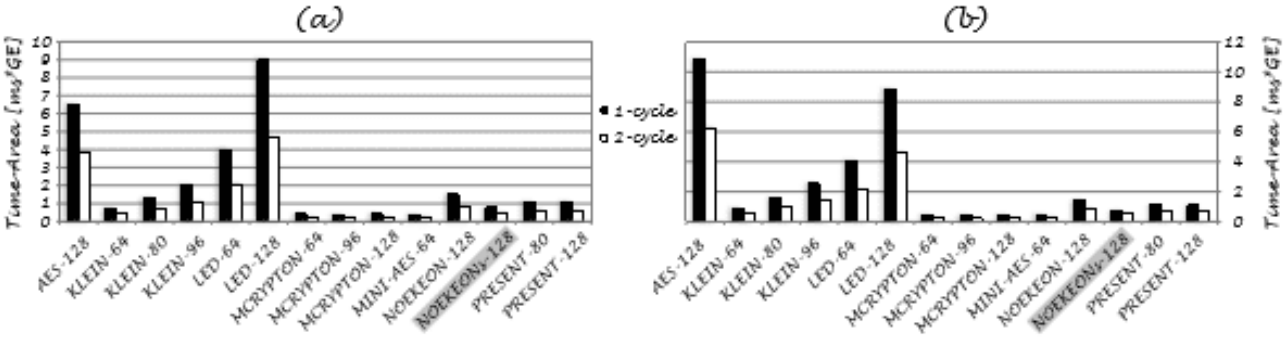


Рис.9: Минимальный продукт временной области [мс · GE] для модуля ENC / DEC: (a) Ограничение по времени. (b) Без ограничений.

Самая высокая производительность по этому критерию показана мини-ай и всеми версиями mcrypton. С более чем 60% накладными расходами, klein-64 занимает третье место, а наименьшая производительность снова показана led-128. Для всех проверенных шифров он утверждает, что архитектура на основе двух циклов обеспечивает от 40% до 45% более высокую эффективность по отношению к этой метрике.

При переходе от неограниченных конструкций с ограничением по времени наибольший выигрыш показан AES с 40% -ным уменьшением продукта временной области, а ноэкеоны достигают даже отрицательного усиления с 8% -ным увеличением продукта временной области. В общем, это отношение колеблется между 0,85 и 1,00, что отражается в довольно небольшом общем улучшении.

Результаты для среднего потребления энергии получены с учетом коммутационной активности схемы и основаны на результатах синтеза. При рассмотрении полностью развернутой конструкции (1 цикл) средняя мощность измеряется, следовательно, усредняется за один такт, что фактически отражает мгновенное потребление энергии. Для двухцилиндровых схем мощность усредняется в течение двух тактов. Чтобы исключить зависимость данных, мы усредняем потребление энергии по 100 случайным входным векторам для каждого измерения.

Потребляемая мощность линейно связана с рабочей частотой, эта метрика непосредственно влияет на значение измеряемой мощности. Устанавливаем частоту как обратную критическую траекторию. Поэтому потребляемая мощность каждой конструкции измеряется в течение ее кратчайшего возможного времени выполнения. Потребление энергии нормализуется по количеству обработанных битов, то есть размер блока сообщения, и рассчитывается как:

$$E = \frac{P \cdot \text{Latency}}{B} = \frac{P \cdot N \cdot t_{cp}}{B} \quad ,$$

где P - средняя мощность, N - количество тактовых циклов, необходимых для шифрования одного блока сообщений, tcp - критический путь схемы, а B - размер блока сообщения.

Рисунки 10-11 иллюстрируют потребление энергии и энергии соответственно. Самые мощные и энергоэффективные проекты снова являются мини-аэрами, mcrypton и klein-64, в то время как ведущие потребляют больше всего. Большой дизайн, такой как aes, потребляет гораздо меньше энергии, чем большинство легких шифров. Это фактически относится к числу раундов, которые в случае aes составляют всего 10, а также размер его блока 128 бит (энергия нормализуется по размеру блока).

Шифры, в основном разработаны для легких приложений. Они не были предназначены для удовлетворения требований с низкой задержкой, налагаемых новыми приложениями. Поэтому некоторые из шифров, которые обеспечивают очень хорошие легкие свойства, демонстрируют довольно низкую аппаратную производительность, когда дело доходит до поведения с низкой задержкой. В основном рассматриваем аппаратные свойства алгоритмов.

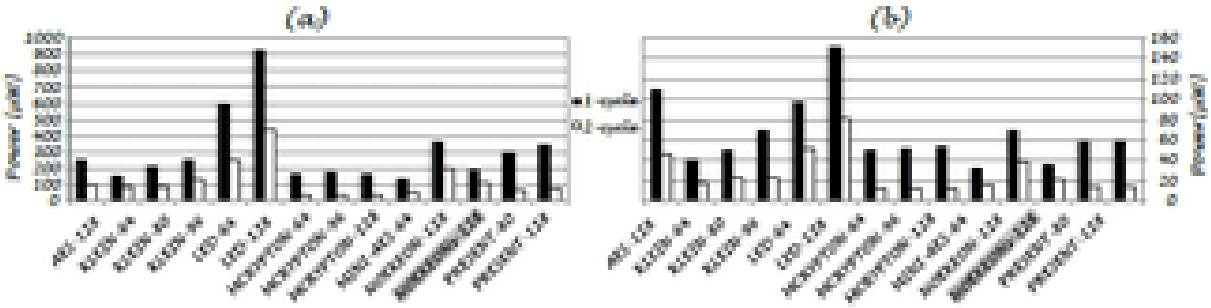


Fig. 10: Power consumption [μW] for ENC/DEC module: (a) Time-constrained. (b) Unconstrained.

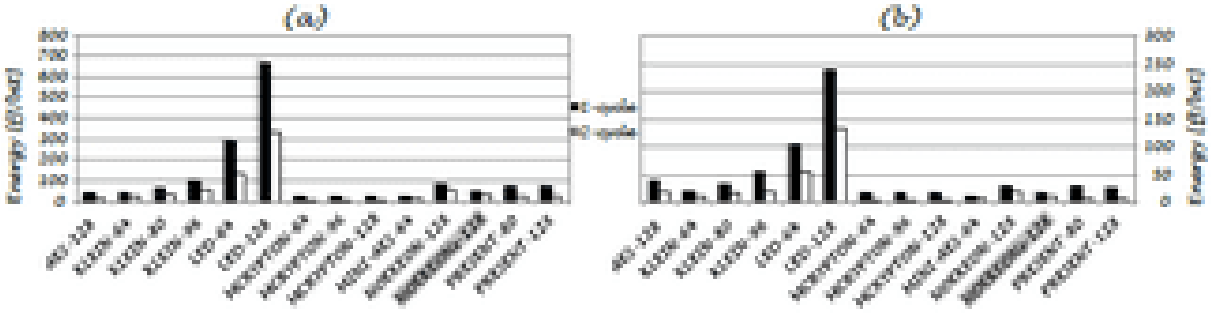


Рис.11: Потребление энергии [fJ / бит] для модуля ENC / DEC: (a) Время ожидания. (b) Без ограничений.

S-бокс. aes - единственный шифр с 8-битным S-ящиком, который значительно больше, чем 4-битные S-боксы, используемые другими шифрами. Теоретически криптографически сильный 8-бит S-box в среднем в 32 раза больше криптографически сильного 4-битного S-бокса. На практике из-за характеристик стандартных библиотек ячеек это соотношение меньше, но остается около 20. Этот факт сильно поощряет использование криптографически сильных 4-битных (или даже 3-битных) S-боксов, где это возможно. Здесь мы подчеркиваем, что даже среди 4-битных (или 3-битных) S-боксов существуют существенные различия в размере схемы [24].

Несмотря на то, что оба пилотных и настоящих используют 4-битные S-боксы, имея относительно легкий круг, количество раундов, из которых они состоят, значительно велико (рис.3). Когда дизайн разворачивается, количество раундов становится значимым фактором в работе алгоритма. Хотя это очевидно в контексте задержки схемы, как только мы нацеливаемся на дизайн с малой задержкой, также становится заметным превышение области. Это подразумевает также более высокую мощность и энергопотребление. Поэтому предлагаем минимизировать количество раундов криптографического алгоритма.

На рисунке 8 показаны только результаты для klein, это иллюстрирует тенденцию, общую для всех шифров (кроме ноэкеона). Фигура ясно показывает, что существует существенный дисбаланс между шифрованием и расшифровкой данных для большинства проверенных шифров. Большинство шифров разработаны с учетом требований шифрования. Поэтому S-box и P-слой часто выбираются так, что их сложность меньше, чем их обратные. Этот факт действительно способствует подходу noekeon, где одни и те же аппаратные ресурсы могут использоваться повторно для шифрования и дешифрования. Этот подход не только экономит значительную площадь, но также уменьшает латентность реализации. Хотя у mcrypton есть инволюционные уровни, существует немалая стоимость для повторного использования их для обоих шифрования дешифрования (из-за необходимости вставки мультиплексоров).

Шесть хорошо известных блочных шифров SPN, включая aes, были выбраны на основе их свойств и идентифицированы как возможные кандидаты для обеспечения хорошего поведения с низкой задержкой. Оценили их аппаратную производительность в контексте шифрования с малой задержкой, тем самым обеспечив первые результаты в поле. Было показано, что полученные результаты (то есть латентность, площадь, мощность и потребление энергии) сильно зависят от конструктивных свойств, таких как количество раундов, сложность раунда и сходство между процедурами шифрования и дешифрования.

В Производительность оборудования для модулей с поддержкой ENC

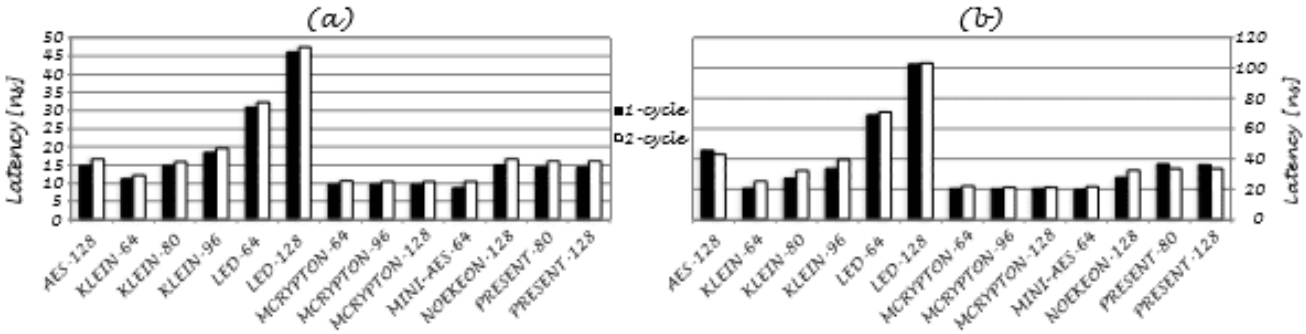


Рис.12: Минимальная задержка [ns] для модуля только для ENC: (a) Ограничение по времени. (b) Без ограничений.

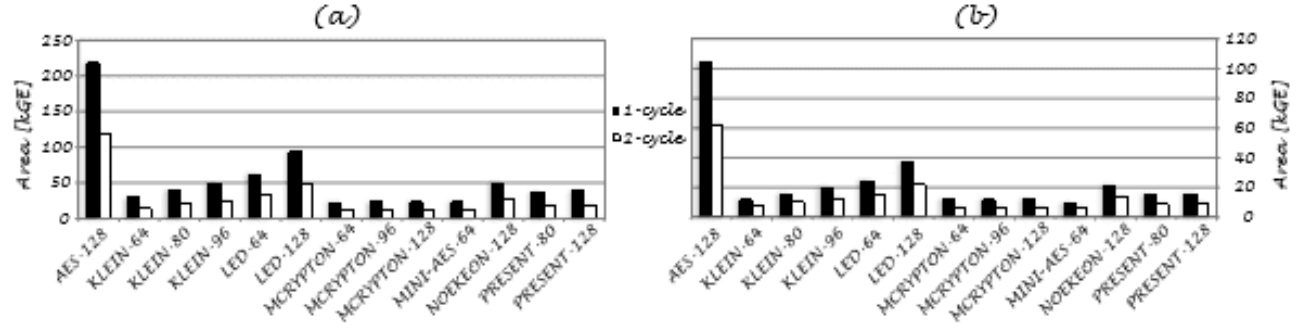


Рис.13: Минимальная площадь [kGE] для модуля только для ENC: (a) Ограничение по времени. (b) Без ограничений.

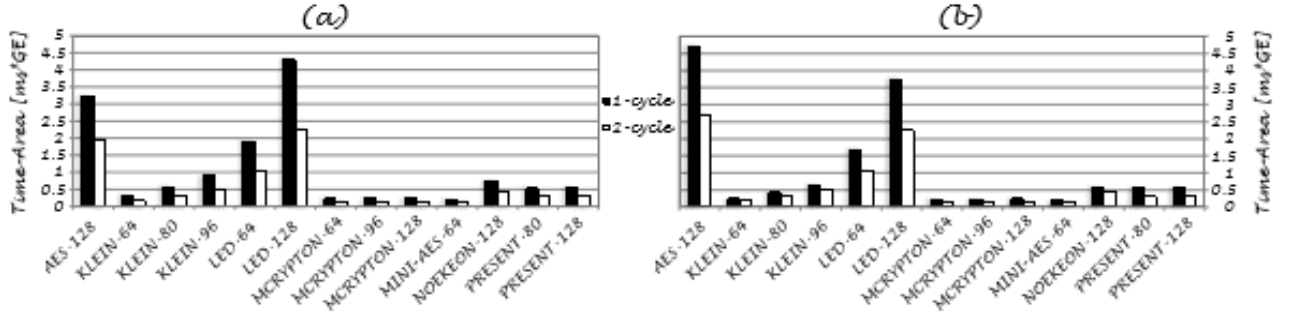


Рис.14: Минимальный продукт временной области [мс · GE] для модуля только для ENC: (a) Ограничение по времени. (b) Без ограничений

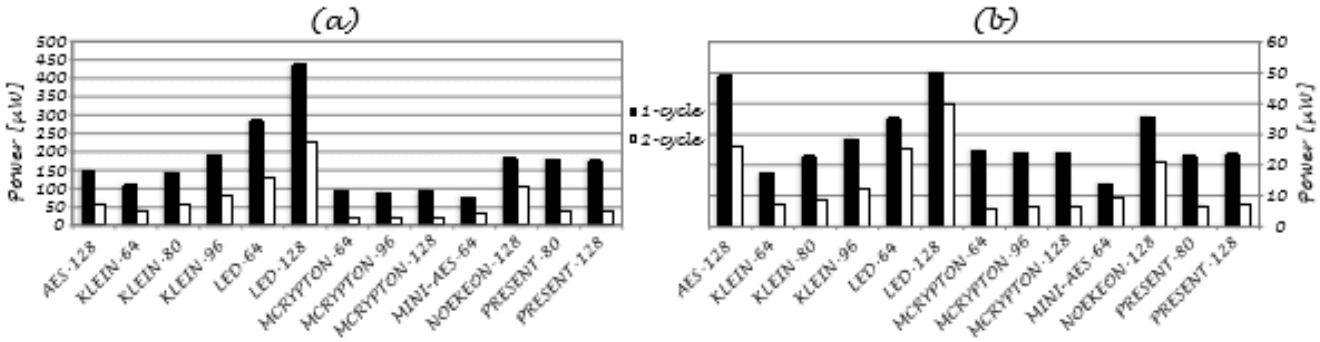


Рис.15: Потребляемая мощность [μW] для модуля только для ENC: (a) Ограничение по времени. (b) Без ограничений.

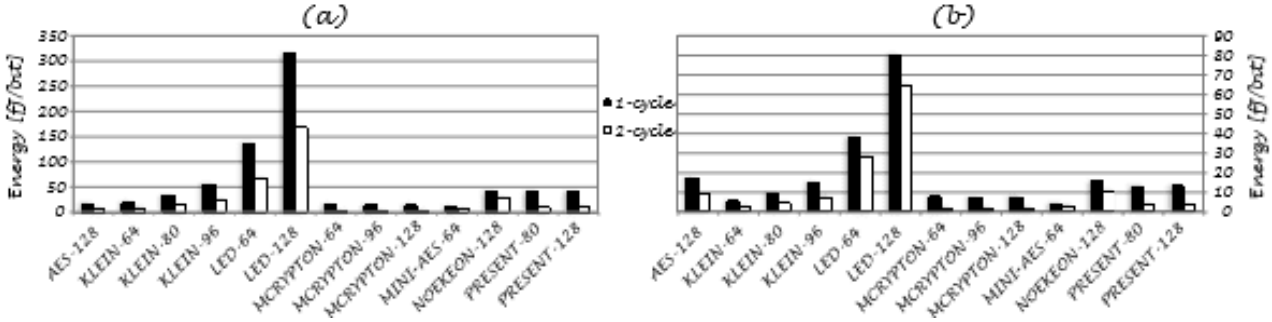


Рис.16: Потребление энергии [fJ / бит] для модуля только для ENC: (a) Ограничение по времени. (b) Без ограничений.

Наиболее скоростная реализация алгоритма AES демонстрирует скорость до 70 Гбит/сек [8]. Такая реализация использует конвейерную архитектуру процессора и требует более 250,000 GE. В то же время наиболее компактная реализация этого алгоритма требует порядка 2,400 GE [9].

С созданием AES потребность в новых блочных шифрах значительно уменьшилась; для почти всех блоков шифрования приложений AES является отличным и предпочтительным выбором. Однако, AES не подходит для чрезвычайно ограниченных сред, таких как RFID-метки и сенсорные сети.

Предлагаем новый аппаратный оптимизированный блок-шифр, который был тщательно разработан с учетом ограничений по площади и мощности.

Рассмотрим компактный блочный шифр. Во-первых, блочный шифр является универсальным примитивом и запускает блочный шифр в счетчике, мы получаем шифр потока. Во-вторых, искусство блочного шифрования кажется немного лучше понятнее, чем потоки шифров.

Блочный шифр, требующий подобных аппаратных ресурсов, как компактный потоковый шифр, может представлять значительный интерес.

Дизайн и реализация шифрования идут рука об руку, и это выявило несколько фундаментальных ограничений и присущих противоречий. Данный уровень безопасности устанавливает нижние границы длины блока и

длины ключа. Просто обработка 64-битного состояния с 80-битным ключом устанавливает фундаментальные нижние пределы объема требуемого пространства. Также аппаратная реализация - особенно компактная аппаратная реализация - способствует повторению.

DES был разработан с учетом наличия аппаратных средств. DES обладает очень конкурентными свойствами реализации. Работа над DES показывает реализацию вокруг 3000 GE [42], а сериализованная реализация может быть реализована примерно с 2300 GE [37]. Длина ключа DES ограничивает его полезность во многих приложениях и делает такие предложения, как DESXL (2168 GE), представляет значительный интерес [37].

Дифференциальный [3] и линейный [32] криптоанализ являются одними из самых мощных методов, доступных криптоаналитику. Чтобы измерить сопротивление настоящего дифференциального и линейного криптоанализа, мы обеспечиваем нижнюю границу числа так называемых активных S-боксов, участвующих в дифференциальной (или линейной) характеристике.

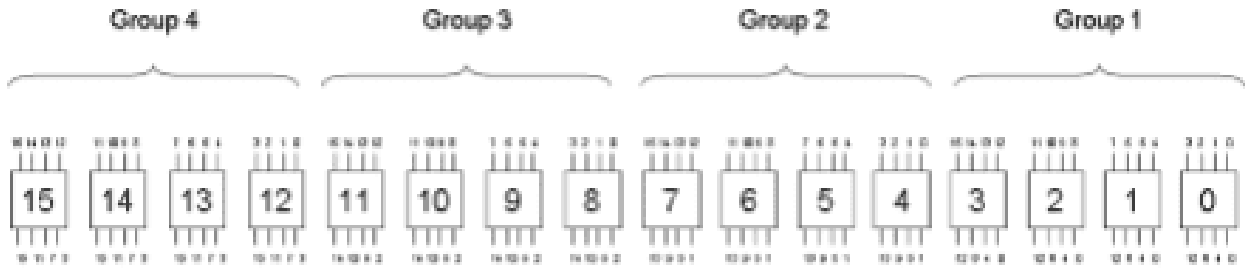


Рис. 3. Группировка S-боксов в настоящее время для целей криптоанализа.

Номера ввода указывают начало S-поля из предыдущего раунда, а номера выхода указывают S-box назначения в следующем раунде.

Дифференциальный криптоанализ. Случай дифференциального криптоанализа фиксируется следующей теоремой. Теорема 1. Любая пятигранная дифференциальная характеристика настоящего имеет минимум 10 активных S-боксов. Разделим 16 S-блоков на четыре группы (рис. 3), и, исследуя слой перестановки, можно установить следующее:

1. Входные биты в S-ящик поступают из 4-х различных S-блоков той же группы.

2. Входные биты в группу из четырех S-ящиков поступают из 16 различных S-блоков.

3. Четыре выходных бита из конкретного S-box вводят четыре различных S-блока, каждый из которых принадлежит к отдельной группе S-блоков в последующем раунде.

4. Выходные биты S-ящиков в отдельных группах переходят в различные S-боксы.

Из этих наблюдений следует доказательство теоремы 1 в Приложении III. Используя теорему 1, любая дифференциальная характеристика в течение 25 раундов настоящего времени должна иметь не менее 5 × 10 = 50 активных S-боксов. Максимальная дифференциальная вероятность нынешнего S-бокса равна 2-2, и поэтому вероятность одной 25-круговой дифференциальной характеристики ограничена 2-100. Передовые методы позволяют криптоаналитику удалять внешние раунды с шифра, чтобы использовать более короткую характеристику. Однако, даже если мы разрешаем злоумышленнику удалять шесть раундов с шифра, ситуация без прецедента, тогда данные, необходимые для использования оставшейся 25-битной дифференциальной характеристики, превышают доступную сумму. Таким образом, границы безопасности больше, чем мы требуем. Однако мы практически подтвердили, что оценка числа активных S-боксов в теореме 1 плотна.

Практическая конфирмация. Мы можем идентифицировать характеристики, которые включают в себя десять Sbox более пяти раундов. Следующая двухмерная итеративная характеристика включает в себя два S-ящика за раунд и удерживается с вероятностью 2-25 в течение пяти раундов.

$$\begin{aligned}\Delta &= 0000000000000011\\ &\rightarrow 0000000000030003\\ &\rightarrow 0000000000000011 = \Delta.\end{aligned}$$

Более сложная характеристика сохраняется с вероятностью 2-21 в течение пяти раундов.

$$\begin{aligned}\Delta &= 0000000000007070\\ &\rightarrow 000000000000000A\\ &\rightarrow 0001000000000000\\ &\rightarrow 0000000010001000\\ &\rightarrow 0000000000880088\\ &\rightarrow 0033000000330033.\end{aligned}$$

Хотя вероятность этой второй характеристики очень близка к значению 2-20, она не является итеративной и имеет мало практического значения. Вместо этого мы экспериментально подтвердили вероятность двумерного

итерационного дифференциала. В экспериментах с более чем 100 независимыми суб-ключами с использованием 223 выбранных пар открытого текста наблюдаемая вероятность была такой же, как и прогнозировалось. Это, по-видимому, свидетельствует о том, что для этой конкретной характеристики нет сопровождающего существенного дифференциала. Тем не менее, определение степени любого дифференциального эффекта - сложная и трудоемкая задача.

Случай линейного криптоанализа обрабатывается следующей теоремой, где анализируем наилучшее линейное приближение к четырем раундам настоящего.

Теорема 2. Пусть ϱ4R - максимальное смещение линейной аппроксимации четырех раундов настоящего. Тогда. $|\epsilon_{4R}| \leq \frac{1}{2^7}$. используем ее непосредственно, чтобы связать максимальное смещение 28-кругового линейного приближения с помощью

$$2^6 \times \epsilon_{4R}^7 = 2^6 \times (2^{-7})^7 = 2^{-43}.$$

Поэтому в предположении, что криптоаналитик должен только приблизиться к 28 из 31 раунда в настоящее время для установки ключевой атаки восстановления, линейного криптоанализа шифрования потребует порядка 284 известных текстовых / зашифрованных текстов.

Структура настоящего позволяет некоторые специальные формы атак. Среди специальных атак, которые мы рассматривали, палиндромных различий, поскольку симметричные различия сохраняются с вероятностью один над диффузионным слоем и некоторые продвинутые варианты дифференциальных атаки [28]. Хотя нападения казались многообещающими в течение нескольких раундов, они очень быстро потеряли практическую ценность и вряд ли будут полезны в криптоанализе настоящего. Усеченный дифференциальный криптоанализ [23, 24], вероятно, будет иметь ограниченную ценность, хотя следующие два раунда усеченное расширение выполняется с вероятностью 1.

$$\Delta = 0000000000000011$$
$$\rightarrow 0000000000030003$$

(повторить двухкруговую характеристику)

$\rightarrow$ ⋮

$\rightarrow$ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1

$\rightarrow$ 0 0 0 ? 0 0 0 ? 0 0 0 ? 0 0 0 3

$\rightarrow \delta_0\ \delta_1\ \delta_2\ \delta_3\ \delta_4\ \delta_5\ \delta_6\ \delta_7\ \delta_8\ \delta_9\ \delta_{10}\ \delta_{11}\ \delta_{12}\ \delta_{13}\ \delta_{14}\ \delta_{15}$ где все $\delta_i \in \{0,1\}$.

Даже при использовании для уменьшения длины дифференциальных характеристик требования к данным по-прежнему остаются чрезмерными.
Для стандартных процессоров известна реализация алгоритма AES, обеспечивающая скорость 7.6 тактов на байт на процессоре Intel Core 2 Q9550 или 6.9 тактов на байт на процессоре Intel Core i7 [10].

2.3. Программно-аппаратная реализация.

Принятие стандарта AES вызвало разработку дополнительных команд для процессоров семейства Intel (2008 г.) Сходное расширение PadLock engine существует в микропроцессорах от VIA Technologies.
Целью данного расширения является ускорение приложений, использующих шифрование по алгоритму AES что обеспечивает скорость шифрования порядка 0.75 тактов на байт [11].

Из-за больших размеров ключа (128, 192 и 256 бит) AES превосходит DES в безопасности.

Важным преимуществом AES является возможность реализации в разных платформах. AES подходит для небольших 8-разрядных микропроцессорных платформ и обычных 32-разрядных процессоров, и это подходит для выделенных аппаратной реализаций. Реализация оборудования может достигнуть пропускную способность в гигабитном диапазоне.

Эффективность реализации и свободная доступность алгоритма прокладывает путь для использования AES в таких приложениях, как беспроводная локальная сеть в соответствии со стандартом IEEE 802.11i и далее таких как ISO 18033-3 [2], IPSec и TLS.

AES используется во многих различных приложениях, его аппаратные реализации алгоритма в основном по оптимизации пропускной способности.
Цели реализации кремния заключается в том, чтобы создать модуль AES, для которого требуется лишь ограниченное количество аппаратных ресурсов с точки зрения площади кремния. Все проектные решения при разработке модуля направлены на снижение стоимости оборудования. Таким образом, результирующая цепь имеет наименьшую возможную площадь. Тем не менее, он полностью функционирует сам по себе. Для шифрования данных он не требует внешнего взаимодействия. Кроме того, модуль также поддерживает дешифрование - это функция, которая не поддерживается многими аппаратными модулями AES.

Вторая цель максимальная эффективность использования энергии. Для повышения энергоэффективности было применено несколько архитектурных улучшений. Методы обширного использования стробирования часов помогают снизить потребление энергии и устраняют резервные токи.

Модуль AES реализуется с использованием стандартных ячеек CMOS. Он не требует жестких макросов, таких как память (RAM, ROM), которые усложняют физическую реализацию и миграцию процессов. Используем модуль AES на 0,35 мм CMOS-процессе.

Измерения показали предсказанные схемой моделирования правильную функциональность схемы и подтвердили ее превосходные характеристики мощности.

AES представляет собой симметричный блочный шифр [1]. Он работает с 128-битными блоками данных. Алгоритм может шифровать и дешифровать блоки с помощью секретных ключей. Размер ключа может быть 128 бит, 192 бит или 256 бит. Фактический размер ключа зависит от требуемого уровня безопасности. Различные версии чаще всего обозначаются как AES-128, AES-192 или AES 256. Сегодня AES-128 является преобладающим и поддерживается большинством аппаратных реализаций.
Центральным принципом проектирования алгоритма AES является простота [10]. Простота облегчает реализацию на разных платформах под разными наборами ограничений. Простота реализуется двумя способами: принятие симметрии на разных уровнях и выбор основных операций.

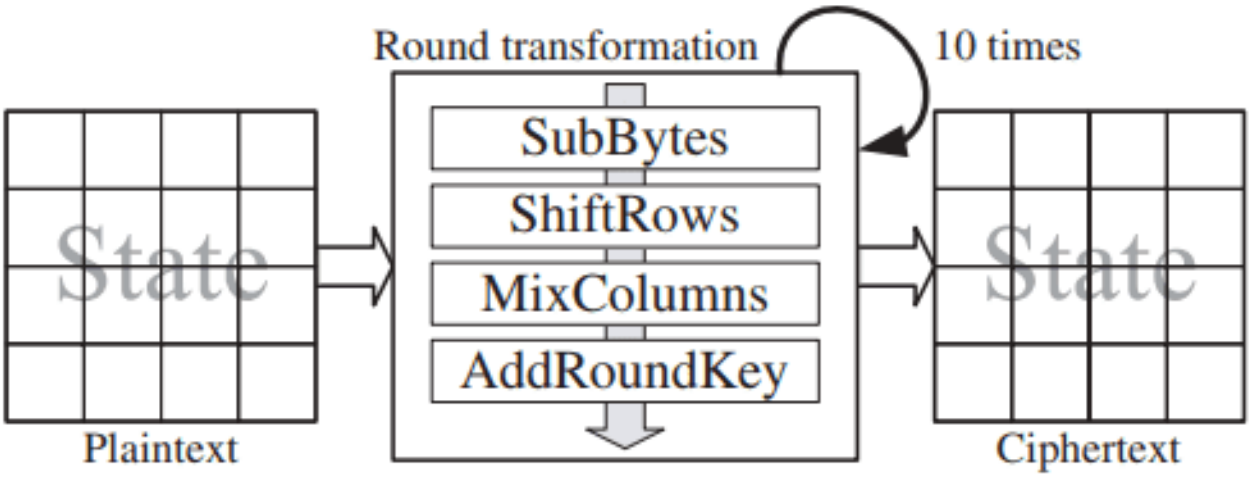
Первый уровень симметрии заключается в том, что алгоритм AES шифрует 128-битные блоки открытого текста, повторно применяя одно и то же округлое преобразование, описанное на рисунке 1. AES-128 применяет округленное преобразование 10 раз; AES-192 использует 12, а AES-256 использует 14 итераций.

Симметрию можно найти и в определении кругового преобразования AES. Симметрия в структуре позволяет многократно использовать аппаратные компоненты и, следовательно, создавать очень маленькие реализации.

Основные операции, используемые в алгоритме AES, можно легко описать в терминах операций над конечным полем GF (28). Это свойство позволяет нам рассуждать об алгоритме с использованием установленных математических методов, облегчая анализ безопасности, а также построение оптимальных реализаций. Во-вторых, конечное поле операции можно реализовать очень эффективно в аппаратных средствах по сравнению с целочисленной арифметикой, где наличие переноса требует специальных мер предосторожности.

Круговое преобразование изменяет состояние 128 бит. Начальное состояние - это входной текст, а конечное состояние - после округлых преобразований - это выходной шифротекст. Государство организовано как 4 матрицы байтов. Круговое преобразование скремблирует байты состояния

либо по отдельности, либо по строкам, либо по столбцам, применяя функции SubBytes, ShiftRows ,



На рисунке 1 AES выполняется циклическое преобразование

MixColumns и AddRoundKey последовательно. Начальная операция AddRoundKey предшествует первому раунду. Последний раунд немного отличается от остальных: операция MixColumns опущена.

Функции циклического преобразования - это линейные и нелинейные операции, которые могут быть обратимы для дешифрования с использованием их инверсий. Каждая функция влияет на все байты состояния. Функция SubBytes является единственной нелинейной функцией в AES. Он заменяет все байты состояния, используя таблицу look-up. Содержимое таблицы может быть вычислено инверсией с конечным полем, за которой следует аффинное преобразование в двоичном поле расширения GF (28). Результирующая справочная таблица часто называется S-Box. Тот же S-Box используется для всех 16 байтов состояния. Функция ShiftRows - простая операция. Он вращает строки состояния смещением. Смещение равно индексу строки: первая строка не сдвигается вообще, последняя строка сдвигается на 3 байта влево. Функция MixColumns обращается к столбцу State, и работает над каждым столбцом одинаково. Он интерпретирует столбец как многочлен над GF (28) со степенью <4. Байт состояния - это коэффициенты многочлена. Выходной столбец соответствует полиному, полученному умножением на постоянный многочлен и уменьшающему результат по модулю x4 + 1. Функция AddRoundKey добавляет круглый ключ к состоянию. Это 128-битная операция XOR. Новый круглый ключ выводится на каждую итерацию из предыдущего круглого ключа. Начальный круглый ключ равен первому секретному ключу. Вычисление круглых ключей основано на функции SubBytes и дополнительно использует некоторые простые операции на уровне байтов, такие как XOR.

Расшифровка вычисляет исходный текст зашифрованного текста. Во время дешифрования алгоритм AES отменяет шифрование, выполняя обратные округлые преобразования в обратном порядке. Круглое

преобразование расшифровки использует функции AddRoundKey, InvMixColumns, InvShiftRows и InvSubBytes-in этот заказ. AddRoundKey - его собственная обратная функция, потому что функция XOR является ее собственным обратным. Круглые ключи должны вычисляться в обратном порядке. InvMixColumns нуждается в другом постоянном полиноме, чем это делает MixColumns. InvShiftRows вращается вправо, а не влево. InvSubBytes меняет таблицу поиска S-Box на обратное аффинное преобразование, за которым следует одна и та же инверсия по GF (28), которая использовалась для шифрования.

Большинство операций AES ориентированы по байтам. Таким образом, они могут эффективно выполняться на 8-битных процессорах. В 32-разрядных процессорах AES также эффективен, поскольку некоторые 8-битные операции могут быть объединены для формирования 32-битных операций. В аппаратных реализациях подходит любой размер слова. Большинство аппаратных реализаций предпочитают 128-битные архитектуры. Это обеспечивает максимальную степень параллелизма для увеличения параллелизма вычислений AES. Более высокая степень параллелизма позволяет повысить пропускную способность. Некоторые реализации даже разворачивают десять итераций кругового преобразования. Используя дополнительное оборудование для каждого из десяти раундов, создается конвейерная версия аппаратного обеспечения AES. Это происходит по цене 10-кратных аппаратных ресурсов. Такой подход, очевидно, пренебрегает симметрией, присутствующей в проекте AES. Кроме того, конвейерная обработка не всегда полезна, так как существуют режимы работы, используемые для массового шифрования, которые не могут использовать параллелизм, заданный архитектурой конвейера. Большинство аппаратных реализаций AES имеют только один раунд, реализованный на аппаратном обеспечении, который используется повторно для вычисления всех десяти итераций. Для пропускной способности в мегабитном диапазоне нет необходимости реализовывать все круглые аппаратные средства с использованием 128-битной архитектуры. Меньшие архитектуры могут выполнять требования. Размер архитектуры (32 бит, 128 бит) определяет размер схемы. Для 32-битной архитектуры требуется 4 S-Box для вычисления функции SubBytes 32-битного слова, а для 128-битной архитектуры требуется 16 S-Box. S-Box - это самые просторные части аппаратной реализации AES. Количество S-боксов определяет общий размер аппаратного модуля AES. Таким образом, использование эффективного подхода для реализации S-Boxs имеет решающее значение для аппаратного обеспечения AES. Основными вариантами реализации S-Box являются поисковые таблицы с использованием ПЗУ. ROM должны хранить 8 256 ¼ 2048 бит. В качестве альтернативы, можно вычислить вывод S-Box путем вычисления инверсии конечного поля и последующего аффинного преобразования. Wolkerstorfer et al. указал, что комбинационная логика может делать это вычисление так же эффективно, как ПЗУ могут выполнять таблицы вида [11]. В частности,

использование комбинационной логики является превосходным, когда требуется дешифрование. В этом случае размер ПЗУ удваивается, чтобы хранить 4096 бит, тогда как вычислительный подход может повторно использовать схему инверсии для InvSubBytes.

Требование к хранению в реализации AES сильно влияет на общий размер схемы. Для хранения в реализации AES-128 требуется не менее 256 бит: 128 бит для хранения состояния и 128 бит для хранения фактического круглого ключа. На некоторых платформах более эффективно использовать дополнительную память. На FPGA Xilinx и Pramstaller и Wolkerstorfer [7] показано, что дублирование государства может снизить общие затраты на оборудование. В схемах со стандартными ячейками хранение с использованием триггеров является более дорогостоящим. Таким образом, стандартные схемы ячеек обычно хранят состояние только один раз. Вопросы памяти также представляют интерес для создания «круглого ключа». Программные реализации на 32-битных платформах заранее заблаговременно вычисляют все круглые ключи. Это экономит время, когда несколько блоков должны быть зашифрованы и облегчают дешифрование, когда круглые клавиши используются в обратном порядке. В области эффективных аппаратных реализаций ключевое расписание предпочтительно выполняется «на лету» для сохранения хранилища. На каждой итерации циклического преобразования соответствующий круглый ключ вычисляется из предыдущего. Для дешифрования должен быть вычислен так называемый ключ обратного шифрования до начала дешифрования. Этот ключ является последним круглым ключом шифрования.

Первые аппаратные реализации алгоритма AES проявились во время процесса отбора AES.

Реализации AES, оптимизированные для платформ FPGA, могут существенно отличаться от реализации стандартных ячеек, поскольку функции затрат FPGA сильно отличаются от функций стандартных ячеек. Поэтому выбор связанной работы в основном ограничивается реализацией CMOS, чтобы упростить сравнение. Другим критерием выбора для соответствующей работы является эффективность области.

Аппаратное обеспечение AES от Satoh et al. [8] - 32-битная архитектура. Он имеет разделенный блок шифрования / дешифрования и схему расширения ключа. Четыре S-бокса используются для работы SubBytes, а также для основного планирования. Есть два очень больших мультиплексора. Первый используется для выбора 32-битных слов из состояния или хранилища ключей. Второй мультиплексор необходим на выходе пути данных для выбора соответствующего результата. Из-за их внутренней структуры им нужен один множитель MixColumns и два мультипликатора InvMixColumns. Полностью функциональный модуль шифрования и дешифрования поддерживает 128-битные ключи. Он имеет аппаратную сложность 5400 ворот и достигает 311 Мбит / с.

Очень регулярный подход был представлен Мангард и др. [5]. Эта 32-битная архитектура выполняет шифрование и дешифрование для различных размеров ключей. Он использует 16 экземпляров ячейки данных, которая хранит 8 бит на ячейку. Ячейка содержит множитель MixColumns и выполняет все преобразования, кроме SubBytes. Параметрируемое количество S-Box, которое может быть 4 или 16, делает дизайн масштабируемым с точки зрения пропускной способности данных.
Реализация встроенной интегральной схемы (ASIC) в этой работе является улучшенной версией, которая также поддерживает дешифрование. Это потребовало значительного изменения компонентов, таких как MixColumns и S-Box. В то время как результаты [4] основаны на синтезе и моделировании на уровне транзисторов, в этой работе результаты измерений исходят от изготовленного чипа.

Оптимизированная реализация кремния криптографической алгоритм требует строгой методологии проектирования который придерживается утвержденных принципов проектирования. В нем есть для тщательной оценки различных вариантов дизайна. Дизайн, низкий размер матрицы и низкое энергопотребление являются основные цели, в то время как высокая пропускная способность данных незначительна.
Выбранная аппаратная архитектура дизайна чипа в основном определяет свойства реализации.

Хорошо продуманные проектные решения по архитектурному уровеню помощи в удовлетворении требований, таких как низкая мощность потребление, низкий размер матрицы и допустимая пропускная способность ставки. Почти все опубликованные аппаратные реализации AES имеют пропускную способность в гигабитах в качестве оптимизации цели. Размер матрицы напрямую влияет на стоимость производство высокоточных цифровых схем. Низкое энергопотребление важно для ограничения для бесконтактных устройств, таких как смарт-карты или RFID-метки.

Использование полноразмерных ячеек может вызвать некоторые проблемы.

Во-первых, время разработки значительно возрастает, что делает чип намного дороже. Дополнительно, быстрый переход на новую технологию CMOS это невозможно. Поэтому мы решили сделать все оптимизация в языке описания аппаратного обеспечения (HDL). Это облегчает быстрый дизайн поток, включая синтез, место и маршрут для разных CMOS, и это создало первое в своем роде право кремния.
Для AES алгоритм, было принято решение поддерживать 128-битный версия для шифрования и дешифрования. Наименьший полезный размер слова, а именно 8-битная архитектура, и алгоритм реализован так, что аппаратные ресурсы используются как можно больше.

Выделенная оперативная память более эффективна, если размер памяти больше, чем 256 бит, которые требуются в нашем случае. Дополнительно, выделенная схема ОЗУ будет потреблять гораздо больше потому что

оперативная память использует энергоемкую предварительную зарядку битовых строк для каждого доступа.

Потребляемая мощность стала основной оптимизационной.целью сегодняшнего проекта СБИС. Различия между потребляемой мощностью и потреблением энергии.

Для устройств с батарейным питанием потребление энергии на операцию - это цель оптимизация. Это означает, что продукт задержки питания должен быть сведено к минимуму. Для устройств с пассивным питанием как бесконтактные смарт-карты, средняя потребляемая мощность является серьезной проблемой. Продолжительность операций не имеет значения. Потребляемая мощность за такт общего энергопотребление операции может быть больше. Здесь часто необходимо сериализовать операции поскольку параллельный расчет превысит доступную мощность. Реализация AES сводит к минимуму среднюю потребляемую мощность.

Общая потребляемая мощность схемы CMOS сумма статического и динамического потребления энергии. Статическая потребляемая мощность, вызванная током утечки в основном зависит от размера чипа. Динамическое энергопотребление состоит из загрузки и выгрузки общей емкости (CL) чипа. Уравнение (1) представляет влияние на динамическую мощность потребления. Проектные меры по снижению энергопотребления является результатом минимизации факторов в этом уравнении.

$$P_{dyn} = C_L \cdot V_{DD}^2 \cdot f_{CLKeff} \cdot E_{sw} \qquad (1)$$

Емкость нагрузки на чипе CL увеличивается по мере увеличения ворота помещаются на матрицу. Это означает, что снижение размера штампа, а также уменьшения напряжения питания (VDD) до минимально напрямую снижает потребление энергии. Эти два коэффициента каким-то образом предопределены системой lowdie- размера и условий эксплуатации чипа. Предполагая фиксированное напряжение питания, лучший вариант для маломощного дизайна уменьшаются эффективные часы частота fCLKeff схемы. Это уменьшает мощность потребления.

Синхронизация часов - очень эффективная мера для эффективной тактовой частоты. В архитектуре все регистры данных и почти все регистры логики управления синхронизируются только тогда, когда происходит изменение потенциального сигнала.

Например, в ОЗУ может быть только один 8-разрядный регистр написанных одновременно. Это имеет то преимущество, что только один регистр потребляет энергию на активном фронте такта.

Кроме того, нет необходимости иметь мультиплексоры в вход регистра для сохранения его старого значения. Это уменьшает площадь кремния цепи. В нашем оборудовании AES реализации, эта мера применялась строго во всех подмодулях. Оказалось, что он снижает значительно мощность

потребление. Таким образом, части схемы практически отключены, когда они не в использовании.

Активность переключения Esw схемы может быть уменьшена используя метод, называемый логикой сна. Всякий раз, когда вывода комбинационной схемы не требуется изменений входных данных, тем не менее, вызовет переключение активности и, следовательно, потребляемой мощности внутри хотя расчетные данные не нужны. Чтобы предотвратить эту нежелательную активность переключения входы комбинационной схемы маскируются с использованием ворот. Сигнал сна, который отключает ворота и предотвращает все коммутационные действия комбинационной логики за воротами, потому что вход постоянно ноль.

Внедрим AES с шифрованием и дешифрованием с использованием фиксированного ключа размером 128 бит. Это уменьшает

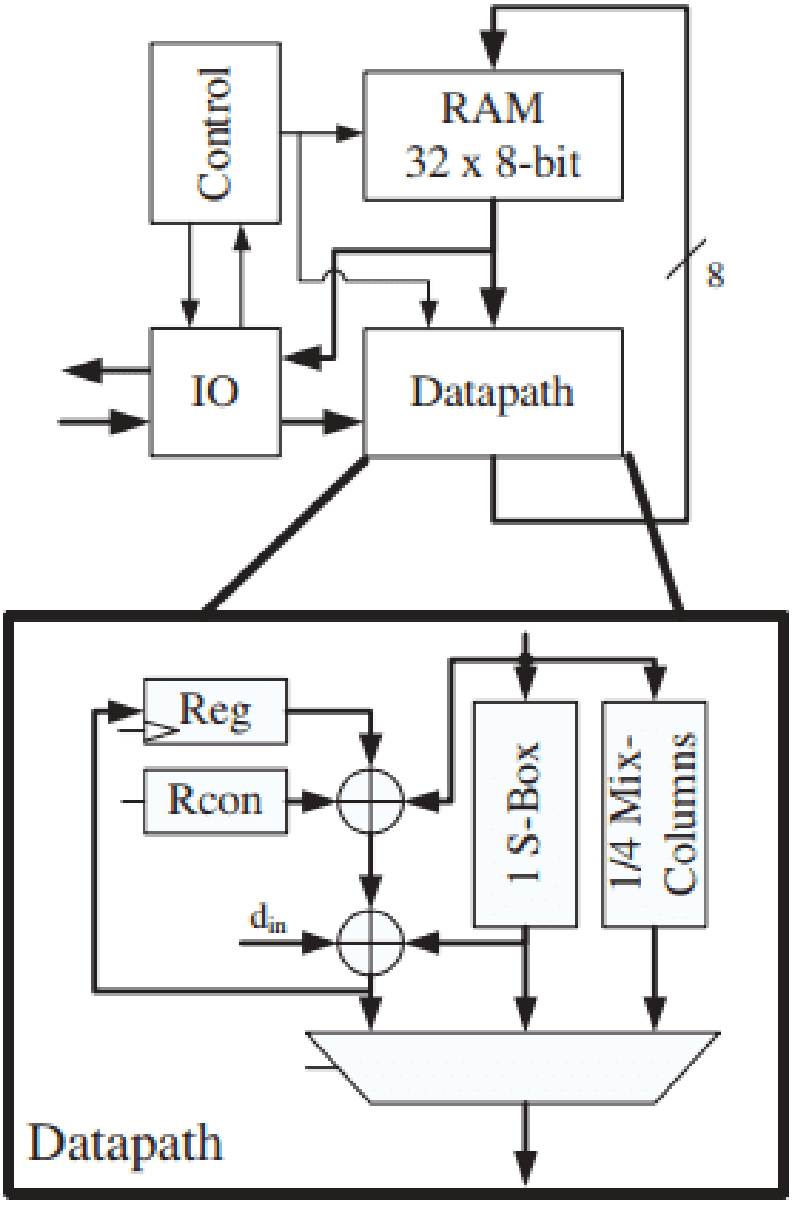


Рисунок 2 Архитектура 8-битного модуля AES

количество раундов до десяти и требуемая память для государства плюс круглый ключ не превышает 256 бит. Маломощные требования нашего чипа слишком ограничительны, чтобы позволить использовать 128-битные операции. Даже 32-разрядная реализация AES не соответствует нашим потребностям. Поэтому решили реализовать 8-битную архитектуру AES, где все операции потребляют значительно меньше энергии, чем 32-разрядные операции. Архитектуру AES можно увидеть на фиг.2.

Основными частями AES являются контроллер, RAM, датапат и модуль ввода-вывода. Модуль ввода-вывода имеет интерфейс микроконтроллера, который позволяет AES модулю, использоваться в качестве сопроцессора. Контроллер принимает команды от модуля ввода-вывода и обеспечивает управление сигналами для ОЗУ и последовательности данных для последовательности операции AES. Контроллер реализован как проводной конечный автомат. Это позволяет оптимизировать эффективности с точки зрения низкого энергопотребления и низкий размер матрицы. Он состоит в основном из 4-битного круглого счетчика и адресных регистров для адресации строк и столбцов ОЗУ. Эти счетчики реализуются как сдвиговые регистры используя одноструннное кодирование. Однократное кодирование гарантирует, что изменения состояния вызывают только два перехода сигнала.

Кроме того, одноразовое кодирование уменьшает нежелательную сбойную активность управляющих сигналов.

Конечные автоматы последовательно повторяют десять раундов состоящий из операций AddRoundKey, ShiftRows, SubBytes, MixColumns или их обратные операции.

Кроме того, все круглые ключи генерируются вовремя для каждого раунда AES. Этот круглый ключ «на лету» помогает сократить необходимое хранилище емкости блока RAM до 256 бит. Первый 128 бит сохраняют фактическое состояние, а второе 128 бит сохраняет текущий круглый ключ. Поскольку резервная память отсутствует для хранения промежуточных значений, контроллер должен гарантировать, что ни один байт состояния или ключевой байт не будет перезаписывается, если это необходимо снова во время расчета.

ОЗУ однопортовое для облегчения реализации силикона. Он реализуется как флип-флоп-память.

Широкое использование стробирования часов снижает потребление энергии.

Эта стандартная ячейка облегчает физическую реализацию используя выделенный блок макросов ОЗУ.

Путь к данным модуля AES содержит комбинационную логику для вычисления преобразований AES SubBytes, MixColumns и AddRoundKey и их обратных операций (рис.2). Преобразование ShiftRows / InvShiftRows реализуется путем соответствующей адресации ОЗУ. Он выполняется, когда результаты операции S-Box записываются обратно.

Остальные компоненты данных - это подмодуль Rcon, некоторые XOR-блоки и 8-битный регистр для хранения промежуточных результатов во время основного планирования. Rcon - это схема, которая обеспечивает константы, необходимые для ключевого графика. Ворота XOR необходимы для генерации круглых ключей и повторно используются для добавления состояния с круглым ключом во время преобразования AddRoundKey. Кроме того, ввод данных и ввод ключа обрабатываются путем передачи данных.

Нашей целью было выравнивание энергопотребления всех операций с данными, возникающих при выполнении алгоритма AES. Эквалайзер очень важен для бесконтактных устройств, потому что самая энергоемкая операция может привести к сбросу всей цепи. Этот сброс может быть вызван падением напряжения питания ниже определенного минимума. Подмодули данных, такие как S-Box или MixColumns, были спроектированы таким образом, что их потребление энергии почти одинаково.

Шифрование или дешифрование 16 байтов входных данных последовательно записываются в ОЗУ через 8-битный интерфейс микроконтроллера, за которым следуют 16 байтов ключей. Начальная операция AddRoundKey выполняется во время загрузки ключа. Для дешифрования ключ обратного шифрования должен быть загружен, потому что все круглые ключи вычисляются в обратном порядке. Выдача команды пуска на вход управления начинает шифрование или дешифрование. Десять раундов AES с функциями SubBytes, ShiftRows, MixColumns для шифрования и функциями InvSubBytes, InvShiftRows, InvMixColumns для дешифрования выполняются в соответствии со спецификацией алгоритма. При вычислении AddRoundKey, который равен для шифрования и дешифрования, последующий круглый ключ получен от его предшественника, использует функциональные возможности S-Box, Rcon и XOR данных. Шифрование может выполняться в течение 1032 тактовых циклов, включая операцию ввода-вывода. Для расшифровки требуется 1165 тактов из-за более сложного ключевого расписания.

Значимым преимуществом 8-битной архитектуры дизайна является сокращение количества S-блоков от четырех или более 32-разрядной реализации до одного экземпляра. Это уменьшает требуемые ресурсы кремния. Единый S-Box используется для операций SubBytes и InvSubBytes, а также для ключевого планирования. S-Box является самой большой частью AAT datapath. Существует несколько вариантов реализации AES S-Box. Самый очевидный вариант - 512 X 8-разрядный ПЗУ для реализации 8-битного поиска таблицы для шифрования и дешифрования.

Особенно подходящим вариантом является вычисление значений замещения с использованием комбинационной логики, как показано в [11]. Одна из особенностей этого S-Box заключается в том, что его можно конвейерно вставить в регистры. В S-Box используется один этап конвейера, который сокращает критический путь S-Box и снижает активность скрещивания. Кроме того, этот регистр трубопроводов используется как

промежуточное хранилище во время операции ShiftRows. Во время замены 1 байта следующий байт считывается из памяти. Замещенный байт записывается в текущий адрес чтения. Выбрав правильно прочитанные адреса, операции SubBytes и ShiftRows объединены. ShiftRows ухудшается до простой адресации.

Чтобы уменьшить активность сигнала в схеме S-Box, применяется расширенный вариант логики сна. Когда выход S-Box не нужен, входы отключены, чтобы поместить S-Box в режим ожидания. Это делается потому, что изменение сигнала на входе комбинационной схемы вызывает потребление энергии, хотя никаких полезных вычислений не выполняется. В режиме ожидания для ввода S-Box применяется значение 0x52 для шифрования или 0x63 для дешифрования. Это имеет то преимущество, что вывод S-Box равен нулю, потому что SBox (0x52) ¼ 0x0 и InvS-Box (0x63) ¼ 0x0. В дополнение к более низкой активности сигнала в режиме ожидания простой XOR достаточно для мультиплексирования вывода S-Box и других выходов компонентов данных (MixColumns и т. д.).

Другим инновационным решением является вычисление операций MixColumns и InvMixColumns. Построим подмодуль, который вычисляет кватер элементов MixColumns и InvMixColumns за один цикл. Вместо использования четырех множителей, как указано в [12], используем один модифицированный множитель.

Столбцы состояния рассматриваются как полиномы над GF (28), которые умножаются по модулю x4 þ 1 на фиксированный многочлен c (x) для шифрования. Для дешифрования используется обратный многочлен c 1 (x). Уравнение (2) показывает вывод обоих уравнений для уменьшения усилий для комбинированного вычисления шифрования и дешифрования. Вычитая полином c (x) шифрования из многочлена дешифрования c 1 (x), можно видеть, что после извлечения общих коэффициентов в дополнение к шифрованию используются только коэффициенты {08} и {0c} результат.

$$
\begin{aligned}
c(x) &= \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \\
c^{-1}(x) &= \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\} \\
c^{-1}(x) - c(x) &= \{08\}x^3 + \{0c\}x^2 + \{08\}x + \{0c\} \\
c^{-1}(x) - c(x) &= \{08\}(x^3 + x) + \{0c\}(x^2 + 1) \\
c^{-1}(x) &= c(x) + \{08\}(x^3 + x) + \{0c\}(x^2 + 1)
\end{aligned}
$$

Этот новый подход приводит к эффективной реализации мультипликатора, как показано на рисунке 3. Он вычисляет 1 байт за цикл после фазы предварительной загрузки трех циклов. Обработка одного столбца состояния занимает семь тактов. Полная операция MixColumns или InvMixColumns занимает 28 тактов для преобразования всего состояния. Критический путь схемы умножителя еще короче, чем один из S-Box.

Силиконовая реализация чипа использует стандартную библиотеку ячеек CMOS стандарта 0.35 мм от Philips Semiconductors. Для обеспечения правильного кремния в первый раз была применена строгая методология проектирования. Схема была описана и проверена в HDL на уровне передачи регистров. Синтез использовал оболочку PKS из Cadence и генерацию генератора тактовых импульсов с помощью CT-Gen. Непрерывное тестирование и проверка устраняют ошибки во время шагов проектирования. Размещение и маршрутизация выполнялись с использованием Silicon Ensemble от Cadence. Проверка бэкэнд обеспечивала технологичность. Эти тесты состояли из статического анализа времени, моделирования мощности, LVS и DRC. После изготовления чипа проверили правильную функциональность схемы на среде тестера HP82000. Набор тестовых векторов, поставляемый из NIST, использовался для проведения функциональных тестов, а также для параметрирования чипа с точки зрения энергопотребления. Область оперативности относительно напряжения питания и тактовой частоты была оценена с помощью проверок развертки.

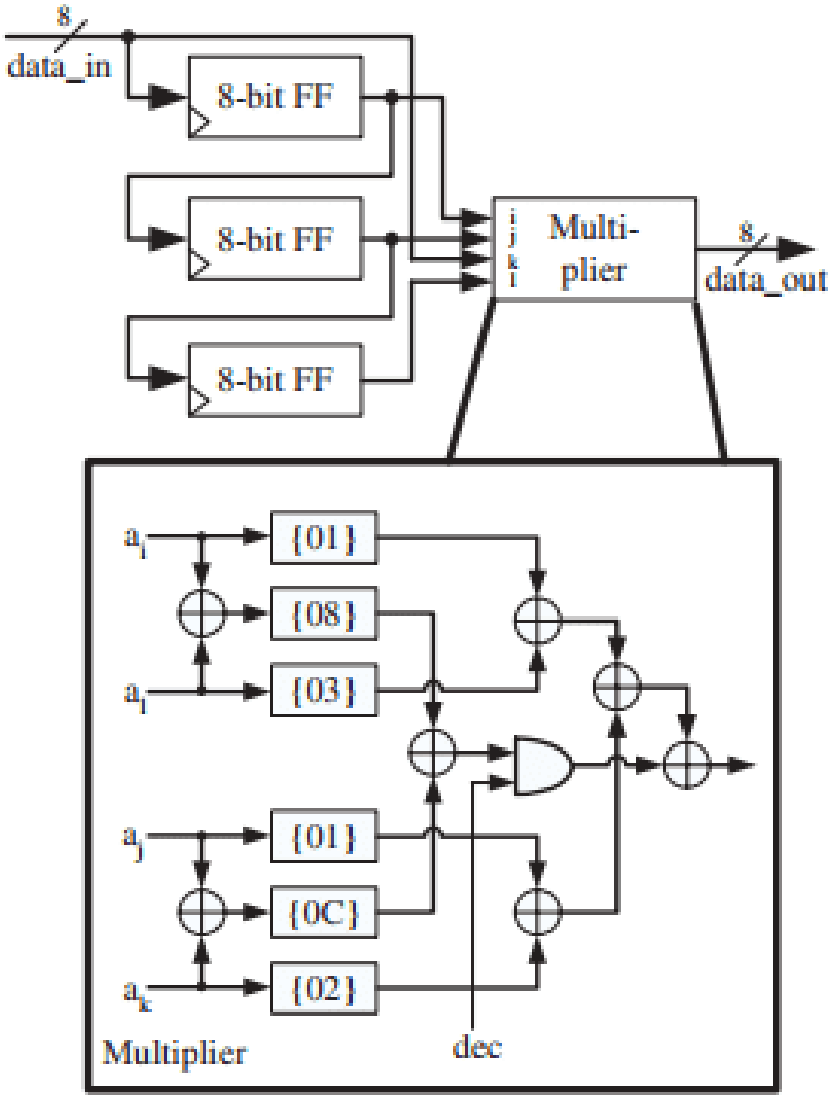


Рис.3. Подмодуль MixColumns

Размер кристалла чипа является наименьшей известной версией AES-128, поддерживающей шифрование и дешифрование. Ядро нуждается в области кремния 0,25 мм2 на CMOS 0,35 мм, что сравнимо примерно с 4400 эквивалентами затвора. Результаты синтеза указывают на сложность 3400 ворот. Разница в сложности компоновки связана с деревом часов, ячейками-заполнителями и другими накладными расходами. Этот размер будет служить долгое время как ссылка на реализацию AES-128. На фиг.4 показана фотография матрицы с использованием коэффициента увеличения 40. Самая большая часть схемы - это ОЗУ с флип-флопом. Это занимает 60% площади чипа. Контроллер, включая конечный автомат, круглый счетчик и регистры адресации, должен, как и S Box 12% каждый. MixColumns занимает 9% от общей площади кремния. Оставшаяся область занята подмодулем Rcon, воротами XOR и мультиплексором на вывод данных

Результаты измерений на чип-тесте показали правильную функциональность даже при очень низком напряжении питания. Чип правильно работает с напряжением питания> 0,65 В. При этом напряжении достигается тактовая частота ~ 2 МГц. При максимальном напряжении питания 3,3 В достигается максимальная тактовая частота 80 МГц. Для шифрования одного 128-битного блока требуется 1032 тактовых цикла, включая загрузку данных и чтение данных. Максимальная пропускная способность для шифрования составляет 9,9 Мбит / с. Производительность для расшифровки почти такая же, как для шифрования.

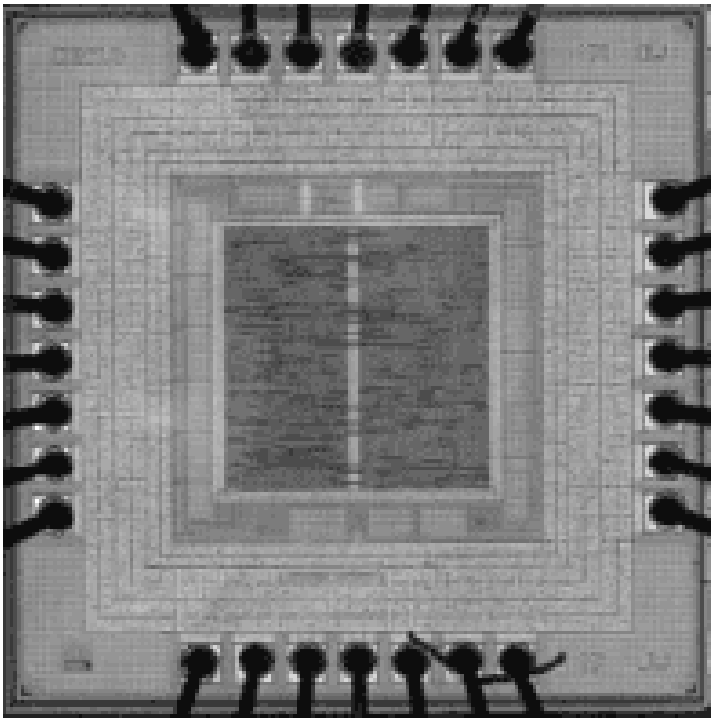


Рисунок 4 Фотография фото AES-128 IC

Общим методом измерения энергопотребления цифровых цепей является измерение их тока питания при условии постоянного напряжения питания. Поэтому падение напряжения тока питания измеряется на

небольшом омическом резисторе с помощью цифрового осциллографа. Этот метод неизменно подвержен ошибкам из-за текущих всплесков, которые являются общими для цифровых систем. Очень малые токи питания трудно отделить от шума. Поэтому применили альтернативный метод измерения энергопотребления чипа AES. Используем метод переноса заряда. Для маломощных схем особенно этот метод обычно применяется. Таким образом, падение напряжения емкости, которая подает микросхему, измеряется, когда чип выполняет свою работу. Измеренный средний расход тока чипа AES-128 составляет 3,0 мА при работе на целевой частоте 100 кГц и напряжении питания 1,5 В. Для сравнения, потребляемый ток потребления с использованием инструмента оценки мощности DIESEL от Philips Semiconductors составляет 3,3 мА за тот же период времени и одинаковое напряжение питания. Эти показатели мощности не включают в себя мощность, рассеиваемую модулями ввода-вывода, поскольку они имеют отдельное напряжение питания.

Распределение текущего потребления происходит следующим образом. Цепочка ОЗУ потребляет 52% мощности, затем контроллер, который использует 18%. Для передачи данных требуется оставшаяся мощность. S-Box потребляет 14%, схема MixColumns - 8%, а остальная часть схемы - 8%. Остальное потребляется выходным мультиплексором, регистром временного хранения, воротами XOR и модулем Rcon.

Сравним нашу работу с модулями AES в [6] и [8]. В таблице 1 показано, что для схемы Satoh требуется 5400 вентилей с технологией 0,11 мм. Наша схема требует 3400 ворот. Это на 40% меньше. Работа Мангард и др. имеет счетчик ворот 8500. Он включает функциональность CBC и интерфейс AMBA с технологией 0,6 мм. Не считая функциональности CBC, можно предположить, что количество заходов 7000. Наше решение требует только половины ресурсов

Table 1: Performance comparison with related work

AES-128 version	Technology (µm)	Area (GEs)	Throughput (Mbps)	Max. Frequency (MHz)	Power (µW)
Ours	0.35	3400	9.9	80	4.5
Satoh [8]	0.11	5400	311	130	—
Mangard [6]	0.6	7000	70	50	—

Достаточно разумное сравнение максимальной тактовой частоты и пропускной способности данных. Работа в [8] достигает 130 МГц-60%

быстрее, чем наша схема. Эта отличная тактовая частота объясняется более поздними технологиями процесса, чем архитектурными причинами. Критический путь [8] намного длиннее нашего, потому что он включает в себя полный S-Box, два больших мультиплексора, множитель MixColumns и XOR-gate. Для сравнения, наш критический путь составляет примерно одну треть длины, поскольку он состоит в основном из половины конвейерной S-Box и некоторой незначительной логики. Критический путь [6] аналогичен нашему.

Мы представили кремниевую реализацию AES, оптимизированную для низких требований к ресурсам. Он оптимизирован для низкого размера матрицы и предлагает отличные характеристики касательно потребляемой мощности. Ядро AES изготовленного чипа имеет площадь 0,25 мм2 по технологии CMOS 0,35 мм, которая сопоставима по размеру с песком. С точки зрения сложности схемы размер равен 3400 эквивалентов затвора. Это на 40% меньше, чем самая маленькая схема AES.

Помимо минимизации площади кремния, наш чип AES рассчитан на низкое энергопотребление. Ряд мер позволил снизить среднюю потребляемую мощность до <5 мВт при работе на частоте 100 кГц и 1,5 В. Широкое использование стробирования часов и новых подходов к снижению активности сигнала в подмодулях, таких как S-Box и MixColumns, позволило сделать эти цифры, Небольшая площадь цепи и ее предельное энергопотребление позволяют применять в устройствах с пассивным питанием, таких как RFID-метки или бесконтактные интеллектуальные карты. Хотя схема требует только минимальных ресурсов, пропускная способность почти 10 Мбит / достигается, когда она работает с максимальной тактовой частотой 80 МГц. Это ниже, чем другие реализации AES.

Считаем, что чип AES послужит эталоном для будущих аппаратных реализаций алгоритма AES, которые оптимизированы для условий с низким уровнем ресурсов по площади и мощности. Наша реализация не включает контрмеры против дифференциального анализа мощности (DPA), но его можно использовать в качестве ссылки для оценки стоимости противодействия DPA.

2.4. Аппаратная реализация симметричных шифров.

Многие работы по симметричным шифрам опубликованные в течение последних двух десятилетий. Потому что большинство основные применения симметричных шифров используют программное обеспечение не удивительно, что почти все алгоритмы - например, AES - были разработаны с хорошей производительностью программного обеспечения. Сдвиг парадигмы, приведет к увеличение спроса на легкие шифры, которые выполняют в аппаратных средствах.

Аппаратные реализации симметричных шифров единственно хорошо зарекомендовавший себя шифр, разработанный с сильной ориентацией на низкую стоимость оборудования - это данные стандарт шифрования (DES).

Сравнивая стандарты односторонней реализации AES и DES, мы находим, что последний потребляет всего около 6% от логики ресурсов AES и имеет более короткий критический путь. Однако исследователи описали маломощные, недорогая реализация AES, которая требует только 3400 (GE) и шифрует открытый текст внутри 1,032 такта.

DES. Вдохновленный однократной реализацией результаты AES и DES, мы реализовали сериализованный версия DES, которая обрабатывает 4-битные и 6-битные слова данных а не с 32 битами и 48 бит. Наша реализация требует 2,310 GE и шифрует открытый текст в течение 144 тактов. Это наименьший зарегистрированный DES реализации, жертвуя пропускной способностью для достижения минимальные требования к площади. Однако 56-битный ключ ограничивает предоставленную безопасность. Это реализация актуальна только для приложений нуждающихся в краткосрочной безопасности или когда значения защищены относительно низко. В некоторых недорогих приложений, такой уровень безопасности является адекватным.

DESL и DESXL. В нашей сериализованной реализации DES, ящики для замещения (S-коробки) занимают приблизительно 32% площади. Мы можем дополнительно уменьшить сложность ворот DES, заменив восемь оригинальных S-боксы с одним новым, исключая семь Sbox а также мультиплексор. Этот легкий DES вариант называется DESL и приводит к приблизительно На 20% меньше, чем у DES (1850 единиц по сравнению с 2,310 GEs). S-box был тщательно отобран и оптимизировано, что позволяет DESL противостоять общим атакам. Таких как линейный и дифференциальный криптоанализ и Атака Дэвиса-Мерфи. Таким образом, DESL обеспечивает безопасносный уровень, соответствующий многим приложениям. ключ отбеливание может применяться для усиления шифрования, с использованием шифрования DESXL, с уровнем безопасности приблизительно 118 бит. DESXL требует 2 170 GE и шифрует открытый текст в течение 144 тактов.

Помимо эффективного внедрения или изменения установленного шифра, альтернатива для легкой криптографий - это разработка нового аппаратно-оптимизированного шифрования с нуля. Мы следовали этому подходу при разработке Present, основанного на SPN (подстановочная перестановочная сеть) блочный шифр с 32 раундами, размером блока 64 бит и размером ключа из 80 или 128 бит. Основная философия дизайна была простота: никакая часть шифрования не была добавлена без уважительной причины, Рисунок 1 изображает очень простой дизайн. Круглый счетчик XOR в ключе планирования например, как сорвать целый класс атак с минимальными накладными расходами. Аппаратные средства дизайнеры предпочитают повторять, потому что это позволяет им повторно использовать части чипа и следовательно, уменьшают размер чипа. Если функции обновления ключа одинаковы в каждом round, это свойство может быть использовать в связи с атаками. Среди возможностей для достижения

отклоняющихся круглых функций выбрали тот, который наиболее эффективен для оборудования: добавление в ключ константы, зависящей от цикла планирования. Потому что необходим круглый счетчик во всяком случае, это не является дополнительной областью требования; для XOR требуется только 13 GE.

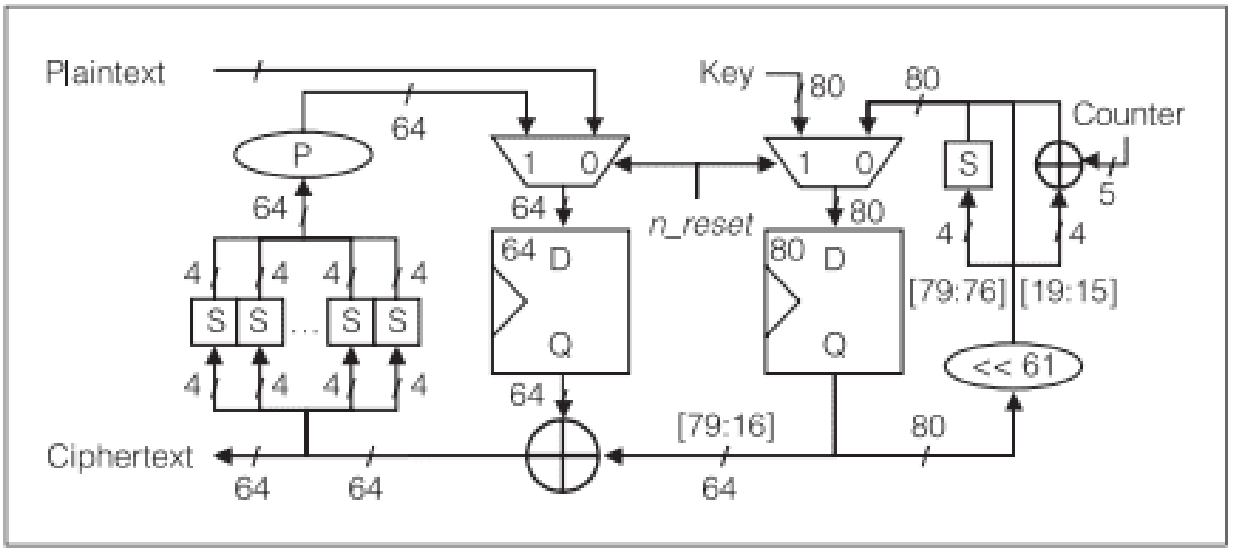


Рисунок 1. Путь данных для текущей реализации. (S: уровень замещения; P: перестановочный слой.) (Источник: Богданов и др. 2)

Настоящий, как и любой другой SPN, состоит из трех этапов: этап смешивания клавиш, уровень подстановки и перестановка слоя. Для микширования клавиш мы выбрали простой XOR потому что эта операция может быть эффективно реализована как в аппаратном, так и в программном обеспечении. Ключевое расписание состоит в основном из 61-битного вращения вместе с S-box и круглого счетчика. Слой замещения содержит 16 S-боксов с 4-битные входы и 4-битные выходы (4 3 4). Используем аналогичные S-боксы как в пути данных, так и в ключе, как в DESL, это может привести к значительной экономии площади и желательна при ее реализации. Выбор 4 3 4 скорее чем 8 3 8 S-боксы были также аппаратными; 8-разрядный S-боксы требуют примерно в 40 раз больше площади, чем 4-битные S-боксы (1000 ГГц против 25 ГП). Однако, 4-битный S-боксы должны быть выбраны очень тщательно, потому что они криптографически слабее, чем 8-битные S-боксы.

В таблице 1 сравниваются результаты внедрения различных легких шифров.

Многие варианты дизайна в AES отражают широкое обсуждение эффективной симметричной криптографии для программных реализаций.

В этом контексте эффективность означает не просто пропускная способность: ресурсы, необходимые для реализации шифрования должны быть небольшими. Фактически, во многих ситуациях, эффективность

ресурсов более критична, чем пропускная способность, особенно многие встроенные приложения шифруют только небольшие полезные нагрузки.
Особенно для устройств с батарейным питанием, низкая вычислительная сложность может иметь большие значение потому, что время обработки напрямую коррелирует с потребляемой мощностью. Современные микроконтроллеры могут входить в различные системы энергосбережения как только они закончат вычисление.
Следовательно, быстро выполняющийся алгоритм может уменьшить энергию потребления и продлить срок службы батареи устройство.

Table 1. Comparison of lightweight ciphers.

Cipher	Key bits	Block bits	Cycles per block	Throughput at 100 kHz (Kbps)	Logic process	Area (GEs)
Block ciphers						
Present	80	64	32	200.00	0.18 µm	1,570
AES	128	128	1,032	12.40	0.35 µm	3,400
Hight	128	64	34	188.20	0.25 µm	3,048
Clefia	128	128	36	355.56	0.09 µm	4,993
mCrypton	96	64	13	492.30	0.13 µm	2,681
DES	56	64	144	44.40	0.18 µm	2,309
DESXL	184	64	144	44.40	0.18 µm	2,168
Stream ciphers						
Trivium[5]	80	1	1	100.00	0.13 µm	2,599
Grain[5]	80	1	1	100.00	0.13 µm	1,294

* AES: расширенный стандарт шифрования; DES: стандарт шифрования данных; DESXL: легкий DES с ключевым отбеливанием.

Сравнивним ранее обсуждавшиеся шифры, которые в первую очередь оптимизированы для аппаратного обеспечения с несколькими программными шифрами. Для сравнения добавим два ориентированных на программное обеспечение потока шифры проекта eStream: Salsa20 и LEX.6. последний является модифицированной AES, в которой несколько байтов промежуточные состояния извлекаются и используются в качестве ключа поток. Salsa20, как и крошечный алгоритм шифрования (TEA) и алгоритм международного шифрования данных (IDEA), основанный на простых арифметических операциях.

Эти шифры были выбраны, потому что их потребление ROM и RAM подходит для небольших встроенных процессоров. Потоковые шифры обычно имеют длительную настройку фазы. Для LEX требуется только одно шифрование AES для настройки, и этап настройки Salsa20. Следовательно, эти шифры могут обеспечить эффективное шифрование небольших полезных нагрузок, часто встречающаяся во встроенных системах. Все обсуждаемые шифры были реализованы для 8-битного AVR микроконтроллера.

Чтобы сохранить исходный код небольшим, используем простой подход для всех наших программных реализаций.
Только таблицы замещения реализуются как (LUT), где это применимо, потому что это обеспечивает огромное ускорение для разумной памяти потребления. Множество быстрых программных реализаций шифров используют большие LUT для достижения более высокого уровня пропускной способности. Это приводит к неприемлемому увеличение размера кода для многих встроенных приложений. Для инверсии, необходимой в IDEA, используем медленный, но чрезвычайно малый алгоритм. Это объясняется небольшим кодом, а также огромным несоответствием между временем шифрования и дешифрования.

Результаты реализаций появляются в таблица 2. Как и ожидалось, программно-ориентированные шифры лучше работать на нашей платформе. У нас были проблемы уменьшения размер кода Hight, но он все еще показывает хорошую производительность шифрования. Хотя его код показывает низкую производительность, а также код IDEA, крайне мала. LEX - это модифицированный шифр AES, но его код меньше, чем у AES, потому что ему не хватает частти дешифрования.

Принимая во внимание компромисс между безопасностью, стоимостью, и производительностью, потоковые шифры кажутся хорошим выбором. LEX и Salsa20 преуспевают в обоих пропускной способности и размера, но они являются хорошим выбором, только если зашифрованная полезная нагрузка является достаточно большой. В противном случае, они производят вычислительные накладные расходы из-за их огромной длины блока и фазы настройки.

В таблице 1 приведены данные об аппаратной реализации наиболее известных блочных шифров.
Обозначения:

Nb – длина информационного блока (в битах);

Nk – длина ключа (в битах);

R – число раундов (циклов шифрования);

GE – условные логические элементы (Gate Equivalent);

Cl/bl – число тактов работы алгоритма, затрачиваемое на шифрование одного информационного блока – мера скорости работы алгоритма, пропускная способность;

SPN – SP сеть;

Feistel – схема Фейстеля;

Lai-Massey – схема Лая-Месси. Алгоритм общего пользования, «облегченный» вариант алгоритма общего пользования, специально разработанный легковесный криптографический алгоритм.

Алгоритм Публикация алгоритма Структура	Nb	Nk	R	GE Публикация реализации	Cl/bl	
AES-128* (2001)[10] SPN	128	128	10	3,100[12] 3,488[13] 2,400[9] 3400[13] 5400	160 1,032 226 1,032 54	
CAMELLIA (2001) [14] Feistel	128	128	18 24	11,350[14]		
		192				
		256				

10 FIPS PUB 197: Advanced Encryption Standard, U.S. Department of Commerce, November 2001.

Алгоритм Публикация алгоритма Структура	Nb	Nk	R	СЕ Публикация реализации	Сб/бл	
CLEFIA- 128* (2007) [15] GFN	128	128	18	2,678 2,890 2,996 4,962[15] 5,979[15] 3,488[16] 3,606[16]	36 18	
		192		6,636[15]	22	
		256		6,462[15]	26	
DES* (1975) ** Feistel	64	56	16	2,309[16] 3,600 [17]	164 28	
3-DES* (ANS X9.52, 1998) [18] Feistel **	64	168	48	2,309[16]		
DESX (2007) [14] Feistel	64	184	16+2	2,629[19]	164	
DESL (2007) [14] Feistel	64	56	16	1,848[16]	164	
DESXL (2007) [14] Feistel	64	184	16+2	2,168[19]	164	
GOST (1989) ГОСТ 28147-89 ** Feistel	64	256	32	65[20] 800[20] 1000[20]	264 264 32	
RIGHT (2006) [21] GFN	64	128	32	2,608 [LDPC09] 3,048[21]	34 34	
Hummingbird-2 (2012) [22]				3,226		
ICEBERG (2004) [23]	64	128	16	7,732[24] 5800	16	

11 FIPS 46, «Data encryption standard», U.S. Department of Commerce/National Bureau of Standards, National Technical Information
Service, 1977. (Revised as FIPS 46-1:1988, FIPS 46-2:1993, FIPS 46-3:1999).

Алгоритм Публикация алгоритма Структура	Nb	Nk	R	GE Публикация реализации	Gi bi	
IDEA [1990] [21] Lai-Massey	64	128	8.5	44,709		
KASUMI [1999] [a] Feistel network	64	64 128	8	2,990		
KATAN [2009]	32	80	254	802[26]	254	
[26]	48		254	927[26]		
	64		254	1054[26]	254	
KTANTAN [2009]	32	80		462[26]		
[26]	48			588[26]		
	64			688[26]	254	
KLEIN [2011]	64	64	12	1800	96	
[27]				1220[27]	267	
SPN		80	16	2629[27]	17	
				1478[27]	271	
		96	20	3768[27]	31	
				1528[27]	335	
LBLOCK [2011]	64	80	32	1320[28]		
[28]						
Feistel						
LED [2011]	64	64	32	966[29]	1,248	
[29]				2,696[29]	32	
SPN				688[29]	1,248	
				2,354[29]	32	
		80	48	1,040[29]	1,872	
				2,790[29]	48	
				688[29]	1,872	
				2,354[29]	48	
		128	48	1,264[29]	1,872	
				3,030[29]	48	
				700[29]	1,872	
				2,354[29]	48	

Алгоритм Публикация алгоритма Структура	Kb	Kk	B	GE Публикация реализации	IC SI	
mCityce (2007) [30] SPN	64	64	12	2,420[30]	13	
		96		2,681 [30]	13	
		128		2,949 [30] 4,128	13	
M65 (2009) [31] Feistel	64	64	32	1,396[31]		
		80		1,550[31]		
Piccolo (2011) [32] GFN	64	80	25	1,048[33] 1,490[33] 616[33] 1,051[33]	450 27 450 27	
		128	31	1,338[33] 1,776[33] 654[33] 1,069[33]	528 33 528 33	
				683 758 (1,269)		
PRESENT (2007) [34] SPN	64	80	31	1,075[35] 1570[34] 1000[36] 1169[36] 27,020[35]	547 32 560 560 1	
		128		1,391[35] 1684[35] [68]	559 32	
PRINCE (2012) [37]	64	128	12	3,491 3776[37]	12	
PRINT (2010) [38]	48	80		503[38] 402[38]	48	
	96	192		967		
TEA (1994) [39] Feistel	64	128	64	1664[40]		

Алгоритм Публикация алгоритма Структура	Nb	Nk	R	GE Публикация реализации	Clk	
XTEA (1998)	64	128	32	2,355[41] 3,490[26]		
TWINE (2012) [42] GFN	64	80	36	1503[42] 1799[42] 1116[42]	36 36 540	
		128		1866[42] 2285[42]	36 36	
SEA (2006) [43] Feistel	96	96	Var.	3,758[24] 1533[24]	93	
	108	108		4,000[43]	111	
	120	120		4,673[43]	1600	
	132	132		5,071[43]	121	
	144	144		5,761[43]		
SIMON (2012) [44]	32 48 64 96 128	64 72 96 128 144 192 256		763 839 1317 [44]		
SPECK (2012) [44]	32 48 64 96 128	64 72 96 128 144 192 256		884 984 1396 [44]		

Результаты реализации сильно рознятся, в зависимости от задачи, поставленной разработчиками:

- Наиболее компактными являются реализации алгоритмов ГОСТ 28147-89 – 615 GE, KATAN – 802-1054 GE (в зависимости от размера информационного блока), KTANTAN – 462-688 GE

Реализации остальных алгоритмов показали результаты существенно большие 1000 GE – общепринятой верхней границы размера микросхемы для симметричного криптоалгоритма.

- Размеры микросхемы существенным образом зависят от архитектуры реализации, которая, в свою очередь, диктуется финальными целями, как то оптимизация по площади, оптимизация по скорости и т.д. Если первая достигается за счет использования сериальной архитектуры (area for speed – serial/parallel), обрабатывающей информацию байтами, высокая скорость достигается за счет распараллеливания и конвейерной обработки данных, что неминуемо приводит к увеличению размера микросхемы.

Итеративная архитектура (iterative).

Увеличение производительности микросхемы за счет увеличения ее размеров (18 тактов на информационный блок при размере схемы 5979 GE против 36 тактов на информационный блок при размере схемы 4950 GE для алгоритма CLEFIA-128.

Типичным распределением ресурсов при аппаратной реализации является распределение ресурсов при аппаратной реализации алгоритма PRESENT (1570 GE): ключевая информация – 30%, S- и P-блоки – 29%, запись внутреннего состояния – 25%, операция XOR – 11%. При этом 55% ресурсов уходят на регистры памяти (ключ + внутреннее состояние) [34].

В настоящее время на мировом рынке наибольшая доля вычислительных устройств приходится на 8-битные контроллеры. Эти небольшие вычислительные устройства как правило имеют ограниченную несколькими десятками килобайт программную память, менее 1 Кбайт статической ОЗУ, небольшую тактовую частоту (порядка нескольких МГц), маленькие размеры регистров, ограниченный набор выполняемых операций.

Далее приводится сравнение отдельных блочных шифров по таким параметрам, как размер кода и объем требуемой оперативной памяти [45].
Приводимые данные были получены для 8-разрядных микроконтроллеров типа AVR, которые являются распространённым семейством 8-разрядных RISC-микроконтроллеров. Результаты реализаций указанных алгоритмов отображены в таблице 2.

В таблице 2 приведены данные о программной реализации на указанной выше платформе различных блочных шифров.

Таблица 2.

Обозначения:
Nb – длина информационного блока (в битах);
Nk – длина ключа (в битах);
R – число раундов (циклов шифрования);
SPN – SP сеть;
Feistel – схема Фейстеля;
Lai-Massey – схема Лая-Месси.

Алгоритм Структура	Nb	Nk	R	Вычислительные устройства	Размер кода в байтах ROM	SRAM в байтах RAM	Enc (cyc/byte)	Dec (cyc/byte)	PUBL.
AES-128 (2001) SPN	128	128	10		2,608	224	415	464	[45]
					1659	33			[46]
					1,912	432	125	181	[47]
					1,143		1,489	1,920	[48]
					2,747		991	1,206	[48]

Алгоритм Структура	Nb	Nx	R	Вычислительные устройства	Размер кода в байтах Byte	Объем в байтах Byte	Кол (стр./byte)	Окно (стр./byte)	Publ.
IDEA (1990) Lai-Massey	64	128	8.5		595 606	0 232	538	1,024	[45] [46]
SEA (2006) Feistel	96	96	var.		2,152 409 1904	0 24 249	805	805	[45] [46] [27]
TEA & (1994) Feistel	64	128	64		1,140 648 1954	0 24 15	764	764	[45] [46] [49]
XTEA					1160 1594	11			[50] [49]
KATAN32					5,816	1,681			[49]
KATAN48					7,076	1,969			[49]
KATAN64					8,348 3290 3606	1,953 625 200			[49] [27] [27]
KTANTAN 32					10,516	614			[49]
KTANTAN 48					11,764	702			[49]
KTANTAN 64					16,252	790			[49]
LEO 64,128					2,648	41			[49]
DES (1976) Feistel	64	56	16		4,314	0	1,079	1,079	[45]
DESL (2007) Feistel	64	56	16	ATMEL ATMega128	5,098		1,048	969	[45]
DESXL (2007) Feistel	64	184	16+2	ATMEL ATMega128	5,192 820	0 48	1,066	985	[45] [46]
HIGHT (2006) Feistel	64	128	32		5,672 402 2510 2050	0 32 117 132	371	371	[45] [46] [27] [27]
PRESENT (2007) SPN	64	80	31	ATmega16	908 1,000 941 854	0 18 18	1,849 5,809 80,704	1,405 8,197	[45] [46] [51] [25]
		128							

Алгоритм Cryptography	Nb	Nk	R	Вычислительная сложность	Размер ключа в байтах Byte	SRAM в байтах RAM	Enc (clk/byte)	Dec (clk/byte)	PUBL.
KASU					1,204	24			[40]
LEGOK					3,568	13			[49]
NOOR					364	32			[40]
KSAN					328	16			[40]
KLEIN					1,368	16			[40]
KLEIN64					5,464 2562	36 105			[49] [27]
KLEIN80					5,676 2672	36 107			[49] [27]
KLEIN96					5,860 2752	36 109			[49] [27]
mysm	16	256			1,502 [9548 1602	150 116	2,667	2,606	[40] [27] [27]
mCRPP					1,076	28			[40]
mCRPTON64					2,728	18			[49]
mCRPTON 96					2,824	20			[49]
mCRPTON 128					3,108	24			[49]
MIDORI					3,164	36			[49]
MIDORI					3,138	16			[49]
CLEFIA 128					4,760	180			[49]
CLEFIA 192					5,010	268			[49]
CLEFIA 256					4,824	268			[49]
TWINE				Omega (k)	1,368 2016	416 28	271	271	[42] [49]
PRINT	48	96			480 1,260		10,415 5,513	10,575 5,239	[48] [48]
	96	192			761 2,397		20,448 10,826	20,624 10,916	[48] [48]
PRIDWI 128					2,515			91	[49]
PRIDWI80					2,434			79	[49]
SKIPJACK					6628 2566 1542	19 133 190			[27] [27]

Другой распространенной платформой для реализации легковесных блочных шифров являются ПЛИС (FPGA). Результаты реализаций на указанной платформе ряда блочных алгоритмов отображены в таблице 3.

Таблица 3.

Обозначения:
Nb – длина информационного блока (в битах);
Nk – длина ключа (в битах);
R – число раундов (циклов шифрования);
SPN – SP сеть;
Feistel – схема Фейстеля;
Lai-Massey – схема Лая-Месси.

Алгоритм Структура	Nb	Nk	R	FPGA	Число блоков	Тактов на блок	Макс. Fтакт. (MHz)	Произв. (Мбит/с)	
AES-128 (2001) SPN	128	128	10	XC2S30-5	222	46		139	
				Spartan-3 XC3S50-5	393	534		16.66	
				Spartan-II XC2S30-6	522		60	166	[52]
				Spartan-3 XC3S2000-5	17,425		196.1	25,107	[53]
				Spartan-2 XC2S15-6	264	3900	67	2.2	[53]
				Spartan-2 XC2V40-6	1,214		123	358	[54]
				Spartan-3	1,600		150	1700	[55]
				Spartan-II	124			2.2	
				Virtex-II	140		123	358	
				Virtex-II	1760		78	1000	
				XCV1000-4 [56]	2151			330	
				XCVE2000-8 [57]	440			1000	
				XC2S30-5 [52]	222			139	
				XC2S30-6 [52]	222			166	
				VIRTEX2(6000 [58]	542			1450	
				XCV-100-4 [59]	490			417	
				XCV-6000-8 [59]	490			743	
				XCV812E [60]	2744			256.5	
				Virtex-2 [61]	1760			1000	
				XC3S5000 -4 [62]	320			270	
CAMELLIA [11]	128			Spartan-3 XC3S50-5	318	675		18.41	
				XC4000XL [14]	874				

HIGHT (2006) Feistel	64	128	32	Spartan-3 XC3S50-5	91	160		65.48	
Hummingbird (2010)	16	256		Spartan-3 XC3S200-5	273		40.1	160.4	
ICEBERG (2004)	64	128		Virtex-2	631		–	1,016	[63]
IDEA (1990) Lai-Massey	64	128	8.5						
PRESENT (2007) SPN	64	80	31	Spartan-3 XC3S50-5	117	256		28.46	
				Spartan-3 XC3S400-5	176		258	516	
				Spartan-3 XC3S400-5	202		254	508	[35]
				Spartan-3E XC3S500	271		–	–	[64]
		128		Spartan-III XCS400-5	202		254	508	[35]
SEA (2006)	96	96	Var.						
Feistel	126			Virtex-II XC2V4000	424		145	156	[65]
TEA & XTEA (1994) Feistel	64	128	64	Spartan-3 XC3S50-5	254	112	62.6	36	
				Virtex-5 XC5VLX85-3	9,647		332.2	20,645	

Результаты реализации сильно рознятся, в зависимости от задачи, поставленной разработчикам.

Таблицы показывают, что подходящие блочные шифры, такие как Piccolo, TWINE, XTEA и AES имеют хорошую производительность, несмотря на компромисс между размером кода и количеством тактов на блок. Также мы видим, что аппаратно-ориентированные шифры (LED, PRESENT, KATAN и KTANTAN) показывают худшие результаты.

2.5. Прораммная реализация поточных шифров.

Сравнительный анализ поточных шифров, как «общего пользования», так и специально разработанных «легковесных» алгоритмов.

Обозначения:

Nk – длина ключа (в битах);

GE – условные логические элементы (Gate Equivalent);

Cl/ byte – число тактов работы алгоритма на байт зашифрованной информации – мера скорости работы алгоритма, пропускная способность.

Алгоритм	Nk	Тактов на инициализацию	GE	Cl/ byte
A2U2 (2011) [1]		56+5	284 [1] 226 [1]	
Enocoro-80 (2008) [2]			2,700	
Enocoro-128v2 (2010) [3]		4869.5	4,100	46.3
F-FCSR-H			4,800	
Grain-128		1137.5		31.2
Grain v.1 (2006) [4]	80	321	1,294[5] 2,599[5]	
MICKEY-128 v.2		56592.1	5,000	1231.4
Salsa20	128		3,842	
Snow 2.0		1086.0		5.0
Trivium (2005) [6]	80	1,333	749[7] 2,599[5] 3,488 5,504 1,294[5]	 8.0 0.125
WG-7 (2008) [8, 9]		10,084	1,097[10]	
AES-CTR		469.6		17.8

Данные о программной реализации некоторых поточных шифров для 8-разрядных микроконтроллеров типа AVR отображены в таблице 5. В то же время для процессоров общего назначения представленные алгоритмы демонстрируют весьма хорошие скорости работы. Так алгоритм Trivium на платформе x86 демонстрирует скорость порядка 4 тактов на байт, что почти в 5 раз выше скорости работы алгоритма AES, полученной для той же платформы.

Другой распространенной платформой для реализации легковесных блочных шифров являются ПЛИС (FPGA). Результаты реализаций ряда поточных алгоритмов на ПЛИС отображены в таблице 6. Для сравнения приводятся данные о реализации блочного шифра AES.

Таблица 5

Сравние результатов программной реализации на указанной выше платформе некоторых поточных шифров.

Обозначения:

Nb – длина информационного блока (в битах);

Nk – длина ключа (в битах).

Алгоритм	Nk	Nb	Шифро-вание (тактов на блок)	Расшиф-рование (тактов на блок)	Размер кода в байтах	SRAM в байтах
Salsa20	128	512?	18,400	NA	1,452	280
LEX	128	3207	5,963	NA	1,598	304

Таблица 6.

Алгоритм	Nb	Nk	FPGA	Число Slices	Max. Freq. (MHz)	Проп. способн. (Mb/s)	Эфф. Mb/s/ Slice
A5/1			Virtex-II	32		188.3	5.88
Grain-128	1	128	Virtex-II	48	181	181	3.77
			Spartan-II	48		105	2.19
			xc3s50-5	50		196	3.92
Grain v1	1	80	xc3s50-5	44		196	4.45
MICKEY-128 2.0	1	128	xc3s50-5	176		223	1.27
			Virtex-II	190	200	200	1.05
			Virtex-E	167	170	170	1.02
MICKEY v2	1	80	xc3s50-5	115		233	2.03
Trivium	1	80	Virtex-II	41	207	207	5.05
			Spartan-II	40		102	2.55
			xc3s50-5	50		240	4.80
Trivium (x64)	64	80	xc3s400-5	344		13,504	39.26
Phelix			Virtex-II	1213	63	1000	0.82
			Spartan-II	1077		750	0.70
RC4			Virtex-II	140		121	0.86
AES			Spartan-II	124		2.2	0.02
			Virtex-II	146	123	358	2.45
			Virtex-II	1780	78	1000	0.56

Сравнение реализаций блочных и поточных шифров показало, что вопреки распространенному убеждению, потоковые шифры не дают существенного преимущества в условиях ограниченных аппаратных или программных ресурсов приложения.

Зачастую поточные шифры весьма неудобны для легковесной реализации предназначенной для обработки очень небольших массивов информации, т.к. имеют, как правило, сравнительно большое время инициализации. Кроме того большинство поточных шифров требуют большого количества памяти для записи своего внутреннего состояния. Эти шифры не могут обеспечить эффективное шифрование небольших объемов данных, что наиболее характерно для встраиваемых систем.

Очевидно задача разработки легковесных шифров, ориентированных на реализацию на 4-, 8-, и 16-битных микроконтроллерах и ПЛИС.

Для криптографической хэш-функции имеется международный стандарт – The Secure Hash Standard (SHA-256) [2]. Наиболее скоростная реализация алгоритма работает со скоростью 7,420 Mb/s. Наиболее легковесная – имеет размер микросхемы 8,588 GE. Международных стандартов на легковесную хэш-функцию нет.

Ниже приводятся результаты аппаратной реализации для наиболее «легковесных» хэшфункций.

---

2 ISO 10118-3. IT. Security techniques. Hash functions. Part 3. Dedicated hash functions.

Таблица 7

Сравние результатов аппаратной реализации различных хэш-функций.

Обозначения:

Nb – длина информационного блока (в битах);

GE – условные логические элементы (gate equivalent);

Cl/bl – число тактов работы алгоритма на блок (байт?) обработанной информации – мера скорости работы алгоритма, пропускная способность.

Алгоритм	Ref	Об Публикация реализации	Суть	Примечания
[illegible]	[illegible]	[illegible]	[illegible]	
[illegible]		[illegible]	[illegible]	
[illegible]		[illegible]		
[illegible]		[illegible]		
[illegible]		[illegible]		
[illegible]		[illegible]		
[illegible]	[illegible]	[illegible]	[illegible]	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	
[illegible]	[illegible]	[illegible]	[illegible]	
[illegible]	[illegible]	[illegible]	[illegible]	
[illegible]	[illegible]	[illegible]	[illegible]	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	
[illegible]	[illegible]	[illegible]	[illegible]	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	
[illegible]	[illegible]	[illegible]	[illegible]	
[illegible]	[illegible]	[illegible]	[illegible]	
[illegible]	[illegible]	[illegible]	[illegible]	

Криптография с открытым ключом

В отличии от алгоритмов блочного шифрования, в настоящее время нет большого выбора асимметричных криптоалгоритмов, пригодных для малоресурсной реализации.

В стандарт ISO/IEC 29192-4 (Mechanisms using asymmetric techniques) включены следующие схемы легковесной асимметричной криптографии:

- cryptoGPS (по именам авторов – Girault, Poupard, Stern) – однонаправленный (oneway) механизм аутентификации, основанный на дискретном логарифме над эллиптической кривой. Алгоритм обеспечивает криптостойкость эквивалентную стойкости 80-битной симметричной криптосистемы. Известна его аппаратная реализация, требующая 724 тактов работы для схемы размером 2876 GE [5].
- ALIKE (Authenticated Lightweight Key Exchange) – однонаправленный механизм ключевого обмена, базирующийся на шифровании. Размер кода для процессора 8051 core – 1.6 Кбайт, время срабатывания на частоте 31 MHz – 80 мсек.
- Механизм выработки цифровой подписи IBS.

Известны характеристики его программной реализации:

Алгоритм	Размер кода [Byte]	RAM [Byte]	Время работы [мс]	Энергия [μJ]
IBS Выработка подписи	54,308	658	896	12,370
IBS Проверка подписи	55,374	922	5,610	77,400

5 ISO 29192-4. IT. Security techniques. Lightweight cryptography. Part 4: Mechanisms using asymmetric techniques

Особенно актуальной становится аппаратная реализация малогабаритного процессора для работы с асимметричными криптоалгоритмами на стандартных эллиптических кривых ECC. Разработаны аппаратные реализации малогабаритных процессоров для работы таких алгоритмов. Для уменьшения площади процессора, жертвуя гибкостью, выбирается работа с конкретной эллиптической кривой над стандартным двоичным полем, что обусловлено ограниченными возможностями встроенного устройства. Устройство работает со стандартными двоичными полями, которые обеспечивают как краткосрочную безопасность (113 бит), так и среднесрочную безопасность (193 бит).

ECC processor performance for scalar multiplication.		
Field	Total area (GE)	Source
$GF(2^{112})$	10,113	[B'06]
$GF(2^{131})$	11,970	[B'06]
$GF(2^{160})$	15,094	[B'06]
$GF(2^{186})$	17,723	[B'06]
$GF(2^{87})^2$	12,944	[B'06]
$GF(2^{113})$	14,735	[B'06]
$GF(p_{160})$	18,720	[G'05]
$GF(2^{163})$	23,000	[W'05]
$GF(p_{160})$	30,333	[OSS'04]

Основной причиной выбора бинарных полей, а не простых полей является отсутствие переноса младших разрядов в старшие при выполнении арифметических операций в поле, что хорошо подходит для аппаратной реализации. Второй причиной является простота структуры для операции возведения в квадрат, которая является основной операцией используемой в алгоритмах для процессора.

Для полей GF(2163) наиболее компактные реализации эллиптической криптографии используют порядка 15,400 GE (2009 г.). Наиболее быстрая реализация эллиптической криптографии над тем же полем дает 83,300 операций умножения в секунду (12 μs на операцию) и требует порядка 154,000 GE.

Наиболее быстрой микросхемой, является микросхема, работающая с 1024-битным модулем и выполняющая 6,350 модулярных возведений в степень в секунду (157.4 μs на одно возведение).

Использование сочетания программного и аппаратного обеспечения может существенно увеличить производительность криптографических алгоритмов с открытым ключом при минимальной площади микросхемы.

Эффективность реализации арифметики в конечных полях, в особенности операции умножения в конечном поле, определяют общую эффективность криптографических алгоритмов на эллиптических кривых. Поскольку асимметричные криптосистемы в среднем на три порядка

медленнее, чем симметричные криптосистемы, основной упор делается на скорость, а не на размер кода.

По сравнению с симметричной криптографией, асимметричная криптография с безопасной длиной ключа, реализованная на аппаратном уровне, по-прежнему требует значительно больших ресурсов, при этом реализуется вполне приемлемая скорость работы. Для криптографии с открытым ключом верхняя граница на размер микросхемы устанавливается в 15,000 GE.

В программной реализации, тщательная оптимизация алгоритмов позволяет микроконтроллерам выполнять операции ассиметричной криптографии менее чем за 1 секунду, что вполне достаточно для большинства приложений. При этом программно-аппаратная реализация, создаёт наилучший баланс между размером и скоростью для многих распространенных вычислительных приложений.

Как отметил президент IACR Барт Пренель в своем докладе «Stream Ciphers and Lightweight Cryptography» на международном семинаре в Пекине (июнь 2011 г.) нет одного оптимального решения, подходящего для использования в различных приложениях и встроенных системах – радиочастотных метках, бесконтактных смарт картах, сенсорах, сопроцессорах для 8-битных процессоров.

Одним из двух основных направлений развития низкоресурсной криптографии является эффективная реализация известных алгоритмов шифрования.

Очень важное значение имеет выбор архитектуры. Например, при аппаратной реализации алгоритма последовательные вычисления уменьшают размер схемы и потребляемую микросхемой мощность (но, увеличивают время работы). С другой стороны малое время обработки информации ведет к уменьшению потребляемой энергии. Это важно, особенно для устройств с батарейным питанием, потому что время обработки информации непосредственно взаимосвязано с энергопотреблением. В современные микроконтроллеры можно ввести различные режимы отключения питания и энергосбережения после окончаний вычислений. Таким образом, быстрое выполнение алгоритма может снизить потребление энергии и продлить время работы батарейного устройства. Поиск сбалансированного решения является непростой задачей. Необходимо, исследование зависимости скорости работы криптоалгоритма от размера микросхемы (area for speed – serial/parallel).
Получение «абсолютно лучших» реализаций вряд ли возможно хотя бы в силу отсутствия методов получения нижних оценок схемной сложности даже для простейших преобразований.

Другим направлением в развитии легкой криптографии помимо эффективной реализации или небольшой модификации известных алгоритмов шифрования, является разработка новых шифров, ориентированных на оптимальную реализацию на аппаратном уровне.

Задачей проектирования средств легковесной криптографии является нахождение компромисса между имеющимися ограничениями на используемые ресурсы и криптографической стойкостью разрабатываемого алгоритма. Даже выбор основных параметров алгоритма (размер информационного блока, размер ключа, размер внутреннего состояния у алгоритмов поточного шифрования, размер конечного поля в случае асимметричных криптоалгоритмов) требуется сделать в пределах обозначенных ограничений. Так, ограничения на используемые ресурсы делают привлекательным разработку криптоалгоритмов с малыми размерами информационного блока и ключа. Однако в этом случае криптографический алгоритм подвержен различным атакам. Поэтому, блочный шифр с малыми размерами информационного блока категорически не рекомендуется использовать в режиме ECB.

Предпочтение в использовании получают преобразования, требующие меньшего размера памяти вычислительного устройства (или меньшего числа логических элементов для их реализации).

Алгоритм выработки цикловых ключей должен порождать их in-place, т.е. цикловые ключи не должны требовать предвычислений.

В последнее время наметилась тенденция использовать в разрабатываемом легковесном алгоритме широко распространенные и хорошо исследованные элементарные преобразования (арифметические и логические операции и т.д.).

Используемые операции должны допускать различные способы реализации в зависимости от имеющихся в наличии ресурсов. Примером могут послужить так называемые ARX-шифры, построенные из преобразований трех видов: сложения по mod $2n$ (Addition), циклического сдвига (Rotation) и операции побитового сложения (Xor).

Во всех обзорах по низкоресурсной криптографии отмечается, что в настоящее время нет общей теории разработки LWC-алгоритмов. Целый ряд авторов предлагает выделить разработку сверхлегких криптографических алгоритмов (ultra-lightweight algorithms) в отдельное направление криптографии. При этом усиливается расхождение между программно- и аппаратно-ориентированными легковесными криптоалгоритмами. Фундаментальное различие в требованиях, предъявляемыми ресурсными ограничениями к программно- и аппаратно-ориентированным легковесным криптоалгоритмам было продемонстрировано в работе [15]. В частности, там было показано, что блочный шифр PRESENT чрезвычайно удобен для легковесной аппаратной реализации, но требует значительных ресурсов при программной реализации.

Поскольку основной задачей низкоресурсной криптографии является минимизация затрачиваемых ресурсов, важным направлением является многофункциональность – возможность с помощью одной микросхемы осуществлять шифрование, реализацию выработки имитовставки (MAC), генерацию псевдослучайной последовательности (PRNG) и т.д. При этом

разрабатываемые алгоритмы должны обеспечить эффективную обработку небольших объемов данных, что наиболее характерно для встраиваемых систем.

Важной проблемой для LWC алгоритмов являются атаки по побочным каналам (side-channel attacks). Эти вопросы пока мало исследованы, но учитывая условия, в которых будут работать алгоритмы, велика вероятность успеха таких атак. Так что перспективным направлением является разработка для низкоресурсных криптоалгоритмов контрмер против атак по побочным каналам.

Глава 3. Система поддержки принятия решений по обеспечению безопасности персональных данных.

3.1. Методика разработки системы поддержки принятия решений.

На современном этапе развития мировой экономики и социально-экономических отношений процесс деятельности организаций и предприятий как организационных систем со-стоит из большого количества деловых процессов, которые характеризуются различной степе-нью сложности управления, разнородностью источников и условий протекания, а также наличием значительного количества взаимосвязей между ними. Качество управления данными процессами напрямую влияет на качество деятельности и достижение поставленных стратегических целей всей организации. Исходя из этого, актуальной становится проблема принятия точных, обоснованных и своевременных решений в процессе организационного управления деловыми процессами и возникающих проблемных ситуациях.

В процессе исследования и анализа деловых процессов в качестве объекта исследования был рассмотрен процесс.

Области защиты персональных данных характеризуется рядом особенностей.

Во-первых, обширная сфера деятельности осуществляющие обработку персональных данных.

Во-вторых, нехватка квалифицированного персонала. Только специалист в области информационной безопасности может разработать систему защиты, адекватную имеющимся угрозам безопасности.

В-третьих, несовершенство имеющейся нормативно-методической базы.

Обеспечение безопасности персональных данных не разовое мероприятие, а непрерывный процесс, поэтому систему защиты необходимо постоянно поддерживать в актуальном состоянии. Среда, в которой осуществляется эксплуатация системы защиты персональных данных (СЗПДн), не является стационарной, поскольку со временем могут измениться:

- состав обрабатываемых персональных данных;
- организационная структура;
- бизнес-процессы;
- вычислительная сеть и др.

В процессе формирования и эксплуатации системы защиты персональных данных необходимо принимать решения, требующие от лица, принимающего решения (ЛПР), знаний в области информационной безопасности, а также достоверных сведений об информационной системе, бизнес-процессах и информационных потоках в организации.

Вследствие этого, одной из ключевых задач является поддержка принятия решений для обеспечения достаточного уровня защищенности персональных данных.

3.1. Методика разработки системы поддержки принятия решений.

Общая схема этапов разработки системы поддержки принятия решений (СППР) представлена на рис. 1.

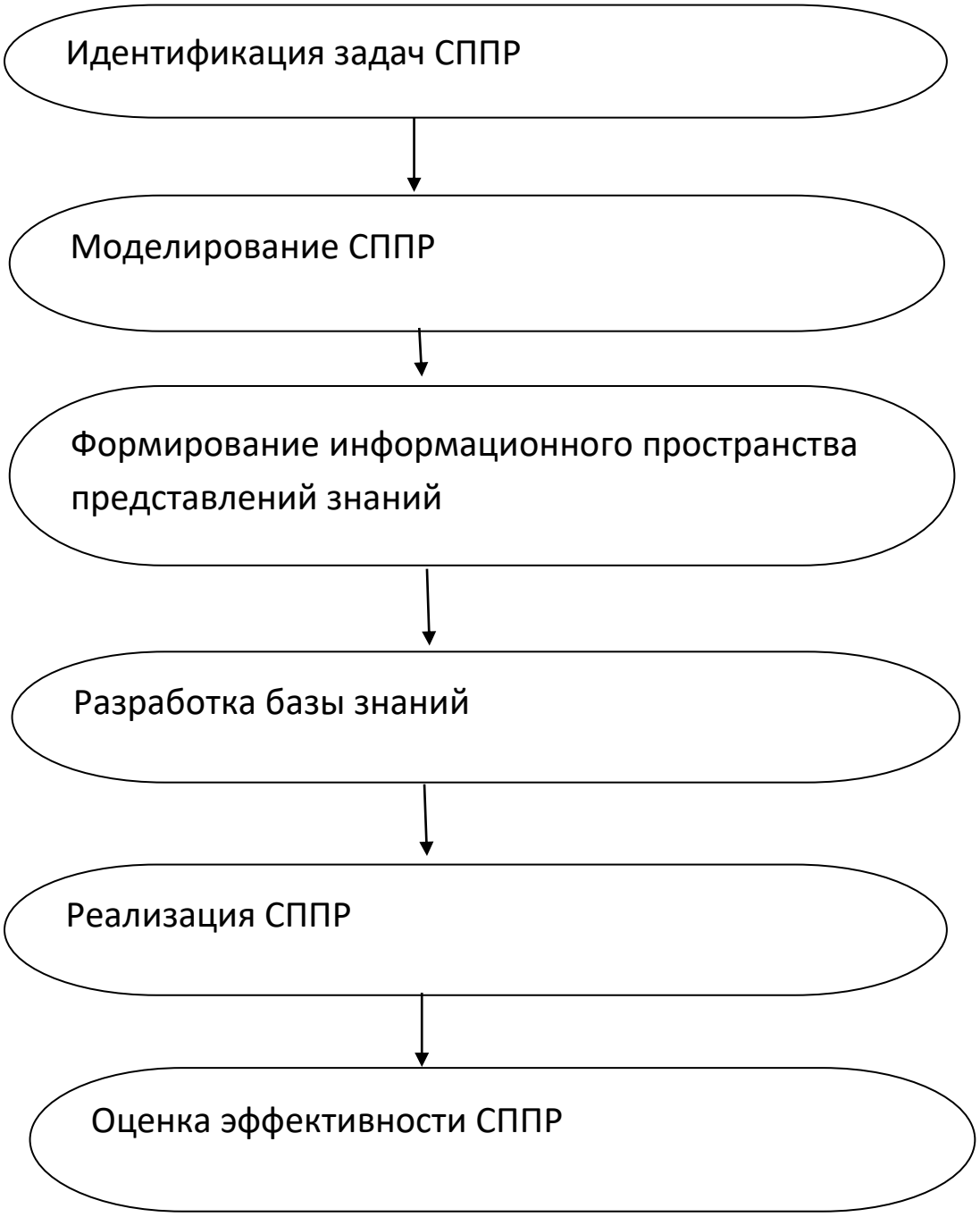


Рис. 1. Этапы разработки СППР

Этап *идентификации задач* СППР является предварительным, подготовительным этапом разработки системы. На данном этапе определяются основные задачи, которые должна решать СППР, ее функции и общая структура.

Общая структура СППР, отображающая наиболее общие принципы ее работы, представлена на рис. 2.

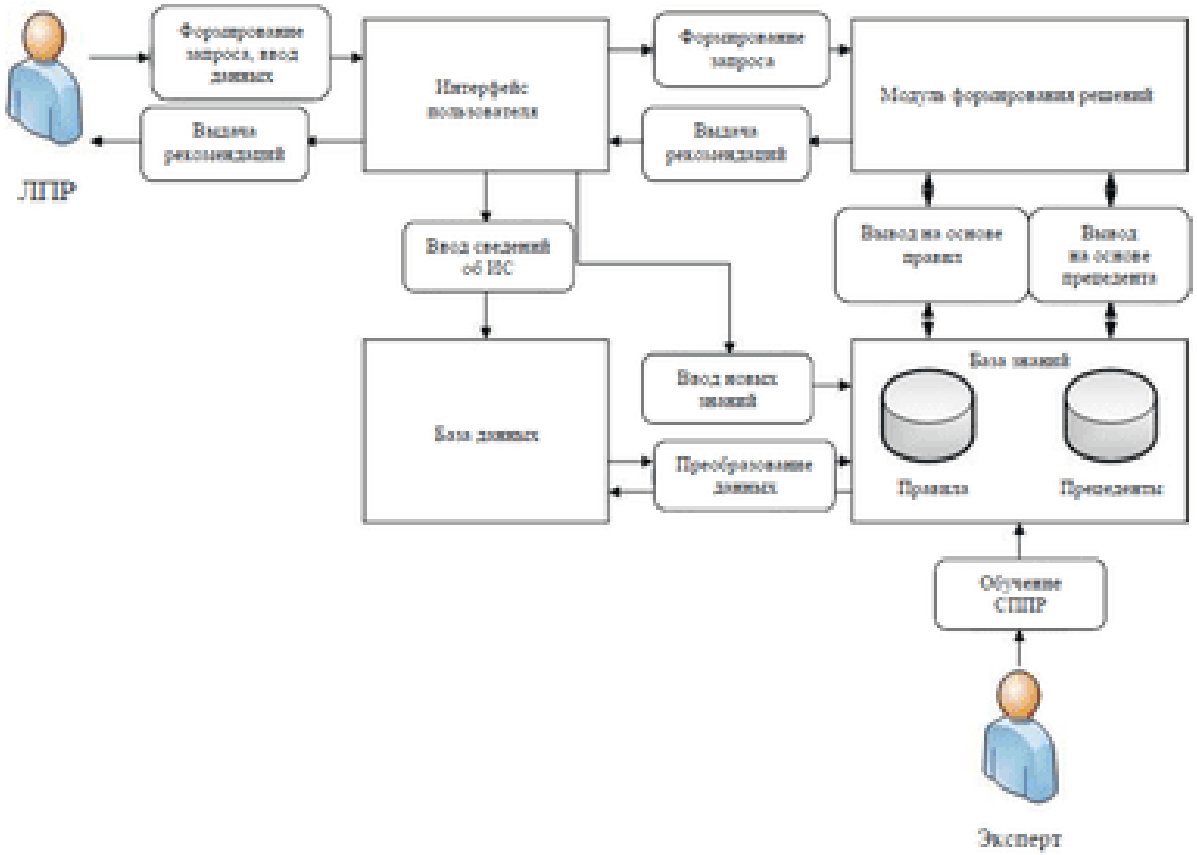


Рис. 2. Общая структура СППР

Также на данном этапе осуществляется сбор и накопление документации, которая содержит информацию о предметной области, используемую в процессе принятия решений [2].

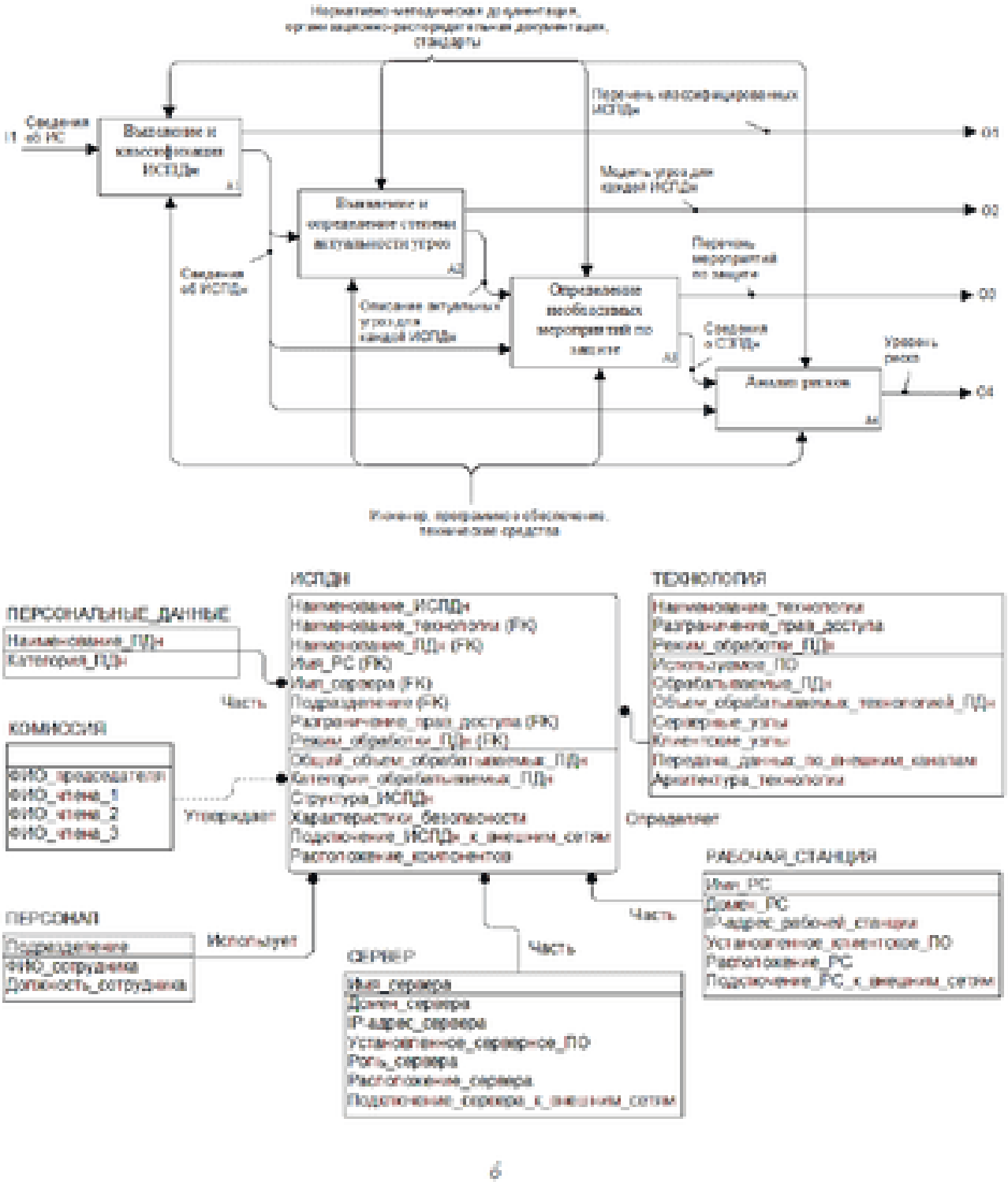
Для формирования рекомендаций предлагается совместное использование механизмов вывода на основе правил и на основе прецедентов: если решение не удается найти с помощью имеющихся правил, то инициируется поиск схожей ситуации в базе прецедентов. Если же поиск по базе прецедентов не дал результата, то выдается соответствующее сообщение пользователю с предложением сформировать новый прецедент. Таким образом происходит обучение базы знаний.

На этапе *моделирования предметной области* осуществляется построение комплекса системных моделей с использованием структурного подхода.

Системное моделирование является основой процесса создания интеллектуальных информационных систем [3]. Для построения системных

моделей использовалась технология структурного анализа и проектирования (SADT) и, в частности, технология IDEF. С ее помощью был построен комплекс функциональных, информационных и динамических моделей.

На рис. 3 представлена функциональная модель IDEF0 на одном из уровней иерархии (рис. 3, *а*), а также фрагмент информационной модели IDEF1X (рис. 3, *б*).



б

Рис.3. Примеры структурных моделей IDEF:
а – функциональная модель IDEF0; *б* – информационная модель IDEF1X

На этапе *формирования информационного пространства представления знаний* проводится онтологический и семантический анализ процессов создания системы защиты персональных данных.
Разрабатываемая СППР является интеллектуальной информационной системой, поэтому одним из основных этапов является этап *разработки базы знаний*. На данном этапе проводится определение прецедентов и правил поддержки принятия решений, а также формирование и подготовка необходимого набора данных и знаний [2].

3.2. Поддержка принятия решений на основе правил и прецедентов.

В разрабатываемой базе знаний используются две модели представления знаний: в форме правил и в форме прецедентов.

Правила – это способ представления основных знаний предметной области, которые объясняют возникновение тех или иных явлений, дают возможность прогнозировать развитие ситуации, позволяют связывать отдельные объекты реального мира [4]. Правила отображают модель рассуждения эксперта в ситуациях, по которым накоплено достаточно примеров принятия решений. В виде правил представляются знания, логическая система которых упорядочена.

С целью поддержки принятия решений в области обеспечения безопасности персональных данных выделяются следующие категории правил:

- правила принятия решений:
- правила выделения ИСПДн;
- правила классификации ИСПДн;
- правила определения угроз безопасности;
- правила определения актуальности угроз;
- правила выбора мероприятий по защите ПДн.
- правила построения базы знаний;
- правила адаптации прецедентов;
- правила оценки эффективности и анализа рисков.

Наиболее простые правила имеют вид:

$$R = <A1, A2, \ldots, An; B>, \quad (2)$$

где $A1 \ldots An$ – предпосылки срабатывания правила; B – заключение.

Поддержка принятия решений на основе прецедентов

В ситуациях, когда использование базы правил не позволяет получить решение, либо правила еще не сформулированы ввиду отсутствия достаточных примеров принятия решении, применяется база прецедентов [2]. При использовании вывода на основе прецедентов используются знания о предыдущих ситуациях или случаях (прецедентах).

Формальная модель системы вывода, основанного на прецедентах, представлена в виде упорядоченной тройки [2]:

CBR = <Case, Onto, Retr>, где Case – база прецедентов; Onto – онтология предметной области; Retr – алгоритм поиска прецедентов.

Для организации эффективного поиска в базе важно провести индексацию и классификацию прецедентов. В рассматриваемой СППР база прецедентов используется в задачах выбора мероприятий по защите персональных данных, а также выбора дальнейших действий при изменении информационной системы. По этой причине множество прецедентов Case классифицировано в соответствии с классами подсистем защиты персональных данных (подсистемы разграничения доступа, антивирусной защиты и др.) и классами проблемных ситуаций (изменение организационной структуры, изменение состава обрабатываемых ПДн и др.).

Последовательность основных процедур вывода на основе прецедентов приведена на рис. 5.

Рис. 5. Основные процедуры вывода, основанного на прецедентах

Первоначально формируется новый прецедент, описывающий сложившуюся проблемную ситуацию. Создание прецедента осуществляется на основе имеющихся в базе данных сведений об информационной системе в целом, а также об ИСПДн. В случае необходимости сведения уточняются у пользователя. Прецедент должен содержать описание проблемы, принятые для решения проблемы действия и результаты применения решения.

Соответственно, прецедент можно представить в виде совокупности следующих объектов [2]:

$$Casek = <Case\_namek, Ci, Xki, Dk, Ek>, \quad (3)$$

где *Case_namek* – идентификатор прецедента; *Ci* – класс, к которому относится прецедент; *Xk*
i – множество значений признаков, составляющих описание проблемной ситуации и образующих входной вектор; *Dk* – множество возможных решений, содержащихся в прецеденте;
Ek – множество оценок эффективности принятых решений.

Прецеденты представлены в базе знаний в виде отдельных фреймов. Новый прецедент содержит только слоты *Case_namek, Ci, Xki.*

После того, как новый прецедент создан, начинается поиск ближайших прецедентов в данном классе, основанный на аналогии. При этом происходит сравнение вектора признаков текущего прецедента и прецедента из базы.

Для сравнения прецедентов был выбран метод «ближайшего соседа». Недостатком данного метода является низкое быстродействие, однако эксплуатация СППР в режиме дефицита времени не планируется. Кроме того, поиск осуществляется не по всей базе, а только внутри определенного класса, что сокращает затраченное время. При использовании метода «ближайшего соседа» для каждого признака *Xki* из *X i* вычисляется мера сходства, в качестве которой используется взвешенная евклидова метрика:

$$sim(X_1, X_2) = \sqrt{\sum_{i=1}^{N} \omega_i (X_1^i - X_2^i)^2}, \quad (4)$$

где *Xki* и *Xji* – значение *i*-го признака для *k*-го и *j*-го прецедента соответственно; *N* – общее количество параметров для данного класса прецедентов; ω*i* – вес *i*-го признака.

В случае, если признак является качественным либо логическим, его значение нормализуется. Веса признаков определяются экспертами с использованием метода анализа иерархий.

Ближайшим является прецедент, метрика которого окажется минимальной.

После того, как ближайший прецедент найден, его решение адаптируется к проблемной ситуации при помощи соответствующих правил. Полученный в результате прецедент после оценки его эффективности заносится в базу знаний. В случае же, если ближайший прецедент найти не удается, то описание проблемной ситуации сохраняется как нерешенный инцидент.

Пользователю выдается соответствующее сообщение с предложением найти решение самостоятельно, после чего новый прецедент заносится в базу. Таким образом происходит обучение системы.

Реализация системы поддержки принятия решений подразумевает написание программного кода, а также физическое наполнение базы знаний. На этом же этапе осуществляется реализация прототипов системы.

На заключительном этапе производится *оценка эффективности* построенной СППР.

Модуль анализа рисков и оценки эффективности, наряду с модулями вывода на основе прецедентов и вывода на основе правил, является основным компонентом разрабатываемойсистемы поддержки принятия решений.

Использование данной СППР позволит организациям и предприятиям проводить классификацию ИСПДн, формировать модель угроз, проектировать систему защиты персональных данных, а также поддерживать ее в актуальном состоянии, не имея в своем штате специалистов в области информационной безопасности. Результатом применения СППР является повышение эффективности принимаемых решений и снижение затрат при создании и эксплуатации системы защиты персональных данных.

3.3. Разработка интеллектуальной системы поддержки принятия решений.

Структура интеллектуальной СППР может быть представлена в виде:

$$DSS = \{Onto, KB\{Rule, Case\}, M, S(M), Dec\} \qquad (2)$$

где Onto – онтология поддержки принятия решений;

$KB\{Rule, Case\}$ – база знаний, содержащая модуль правил Rule и модуль прецедентов Case;

$M = \{M_1, M_2, \ldots, M_n\}$ – множество объект- ных, онтологических, логических и математиче- ских моделей, реализующих функции моделирования процесса принятия решений;

S (M) – модуль, реализующий функцию выбора необходимой модели (моделей) для рассматриваемой задачи;

DEC – модуль формирования решений на основе базы знаний и математического моделирования.

Схема выбора решения с использованием ИСППР представлена на рис. 3.

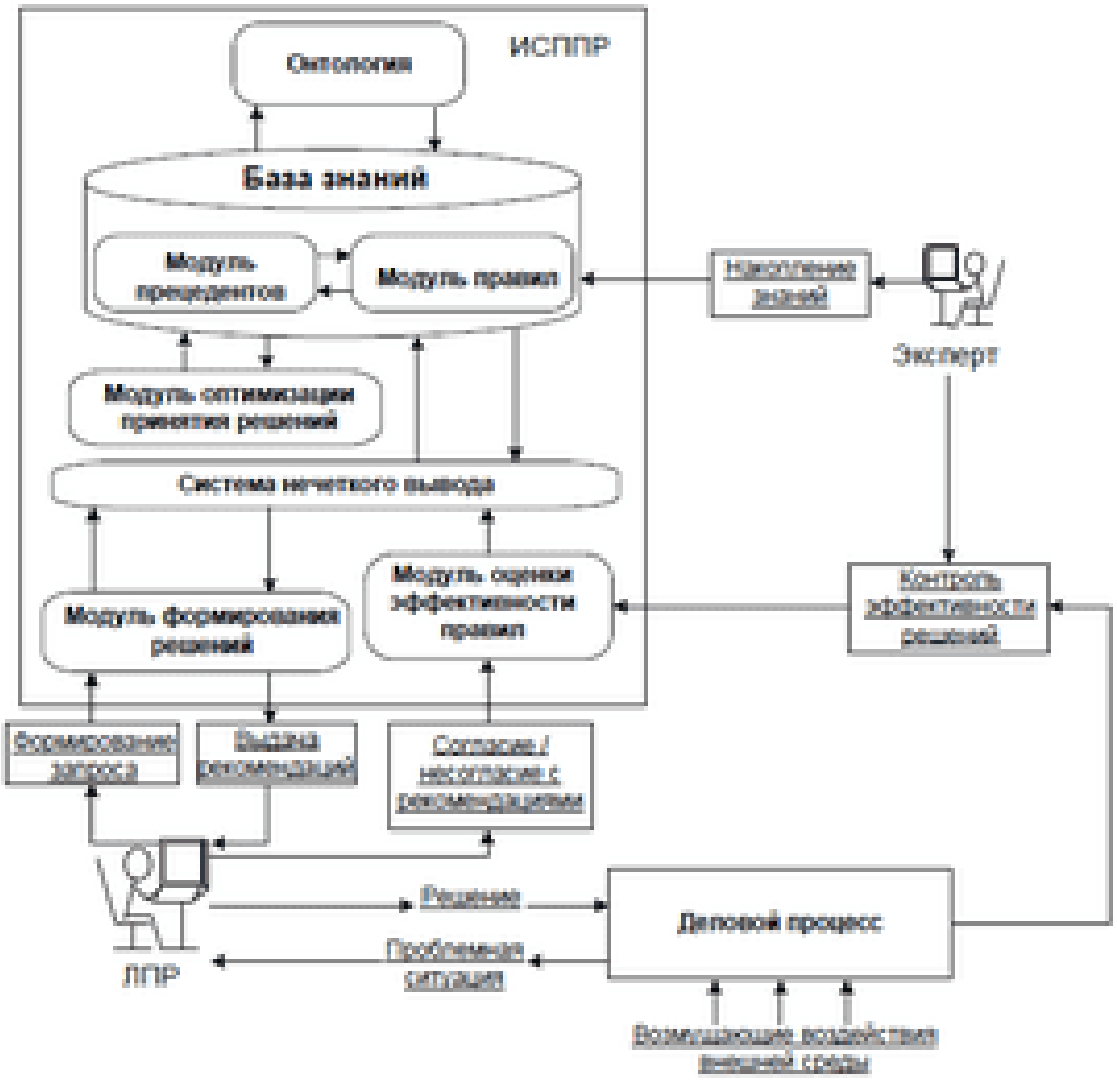


Рис. 3. Схема выбора решения с использованием ИСППР

При организационном управлении разработкой программных проектов возникают проблемные ситуации, описание которых составляется на основе наиболее информативных характеристик управляемых процессов. Описание используется при формировании запроса к ИСППР, который поступает на вход модуля формирования решений.

Поиск решений осуществляется с использованием построенной на основе онтологического анализа нечеткой базы знаний, содержащей правила принятия решений в проблемных ситуациях, а также прецеденты решения конкретных проблемных ситуациях, которые затем преобразуются в правила поддержки принятия решений.

Математическое обоснование формируемых решений производится с применением модуля оптимизации решений, взаимодействующего с модулем формирования решений и базой знаний через систему нечеткого вывода. Расчет оценок альтернатив в модуле оптимизации решений производится с использованием математических моделей, выбранных на основе онтологии поддержки принятия решений.

Таким образом, ИСППР выполняет форми-рование альтернатив решений (поиск возмож-ных вариантов решений) и оценку альтернатив (сопоставление вариантов), а также представле-ние рекомендаций пользователям.

3.4. Методология разработки интеллектуальной системы поддержки принятия решений на основе инженерии знаний.

Ключевым вопросом повышения качества управления сложными системами является повышение эффективности принятия решений в проблемных ситуациях. Трудности, возникающие в процессе принятия решений, заключаются в наличии факторов неопределенности, таких как неопределенность и недостаточность знаний о значении характеристик объектов в проблемных ситуациях, о целях и о ресурсах управления, в том числе временных, и другие виды неопределенностей при определении состояния объекта; наличие проблемы выбора наиболее информативных признаков проблемных ситуаций из большого количества параметров состояния объекта, которые можно получить и проанализировать за ограниченное время; необходимость анализа значительного количества признаков проблемных ситуаций качественной природы, слабо поддающихся аналитической обработке; дефицит времени на принятие решений.

Принятие правильных и своевременных решений должно быть поддержано предоставлением советующей информации, содержащей возможные альтернативы решений, выработанные в результате инструментального анализа возникшей проблемной ситуации, а также на основе знаний в области управления объекта, накопленными экспертами. Поэтому весьма актуально использование поддержки принятия решений в проблемных ситуациях с интеллектуальной информационно-аналитической системой, разработанной на принципах инженерии знаний в рассматриваемой предметной области. Инженерия знаний здесь определяется как совокупность методов и средств извлечения, накопления, обработки, представления и синтеза знаний.

Предлагаю подход к применению инженерии знаний, основанный на моделировании знаний экспертов и их методов решения проблем в предметной области.

Системы поддержки принятия решений (англ.Decision Support System) представляют собой специфический класс автоматизированных информационных систем, которые поддерживают деятельность по принятию решений. Эти системы помогают специалистам выбрать и /или сформировать нужную альтернативу среди множества вариантов при принятии ответственных решений, своевременно предоставляя необходимую информацию.

Таким образом, более точным наименованием системы, принадлежащей данному классу, является «Информационная система поддержки принятия решений» (ИСППР). Подобные системы относятся к

классу интегрированных интеллектуальных систем, сочетающих строгие математические методы и модели поиска решения с эвристическими, логиколингвистическими моделями и методами, базирующимися на знаниях специалистов – экспертов, моделях человеческих рассуждений и накопленном опыте. ИСППР – диалоговая система, которая обеспечивает пользователю доступ к моделям принятия решения и данным для того, чтобы поддержать слабоструктурированные и неструктурированные задачи принятия решения. В составе ИСППР средства искусственного интеллекта составляют экспертную подсистему, основанную на базе знаний и механизме логического вывода. Система обеспечивает поддержку лицу, принимающему решения, помогая в организации информации на основе базы знаний, формировании решений на основе логического вывода и обеспечивая доступ к результатам моделирования.

ИСППР должна соответствовать перечисленным ниже требованиям:

Информационная система поддержки принятия решений, анализируя данные о конкретной проблемной ситуации, знания и модели, должна давать актуальную и достоверную информацию, чтобы обосновать выбор альтернативы принятия решений и, кроме этого, сопровождающие пояснения пользователю.

В составе ИСППР должна быть база знаний о процессах управления в проблемной ситуации и база данных описания проблемных ситуаций.

ИСППР должна содержать средства дедуктивного логического вывода (для формирования рекомендаций в реальном времени) и индуктивного вывода (на стадии интеллектуального анализа данных), а также средства, имитирующие рассуждения по аналогии, для поиска прецедентов проблемных ситуаций.

В составе ИСППР должны быть средства общения с удаленными пользователями, необходимые для предоставления рекомендаций и объяснения рекомендуемого решения.

Важной функцией ИСППР должна стать способность автоматического обнаружения противоречий между знаниями, уже имеющимися в базе знаний, и новыми, поступающими от экспертов или от программ автоматизированного извлечения знаний из данных.

Время формирования рекомендации ИСППР должно находиться в пределах резервного времени для данного типа сложного динамического объекта и для соответствующего класса проблемных ситуаций. Структура ИСППР, построенной на принципах использования интеллектуальных технологий обработки информации, показана на рис. 1.

Рис.1. Структура информационной системы поддержки принятия решений

Для разработки системы поддержки принятия предложена методология объектно - когнитивного анализа [6].

МЕТОДОЛОГИЯ ОБЪЕКТНО-КОГНИТИВНОГО АНАЛИЗА ДЛЯ РАЗРАБОТКИ СИСТЕМ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ

В пространстве знаний информационной системы поддержки принятия решений выделены следующие знания:

- семантические метазнания, представленные в предметной онтологии;
- формальные знания, представленные в виде правил продукций (база правил), на основе которых осуществляется логический вывод решения в проблемной ситуации;
- знания о прошлых проблемных ситуациях и принятых управляющих решениях (прецеденты);
- фактографические данные, получаемые в результате мониторинга состояния сложной динамической системы.

Основой интеграции указанных различных моделей представления знаний является единое информационное пространство когнитивных элементов проблемных ситуаций, сформированное в результате объектно-когнитивного анализа и моделирования процесса управления сложными системами в проблемных ситуациях.

Целью моделирования является системное описание знаний, используемых в процессе управления сложными динамическими объектами в про блемных ситуациях.

В процессе исследования выявлены такие проблемы моделирования знаний, как проблема представления знаний в качестве семантических отношений между объектами предметной области, проблема моделирования знаний о динамике поведения объектов, проблема моделирования операций и методов обработки знаний для формирования рекомендаций по принятию решений. Для решения перечисленных проблем предложена методология разработки ИСППР, основой которой является объектно-когнитивный анализ предметной области, интегрирующий методы объектно-ориентированного анализа, онтологического анализа и семантической сети представления знаний [1].

Объектно-ориентированный анализ – способ анализа, изучающий требования к системе с точки зрения будущих классов и объектов, основываясь на словаре предметной области [2].

Онтологический анализ – это уровень анализа знаний, в основе которого лежит описание предметной области в терминах сущностей, отношений между ними, и действий над сущностями [3].

Семантический анализ – это анализ предметной области, направленный на описание и идентификацию базовых элементов предметной области, установление взаимосвязей (отношений) между ними и определение характеристик отношений [4].

Разработаны следующие принципы объектнокогнитивного анализа:

- принцип иерархической декомпозиции знаний, основанный на применении различных форм абстрактного представления знаний и полиморфизме описания отношений между объектами семантической сети и полиморфизме механизмов логического вывода;
- принцип наследования свойств, основанный на отношении «класс – подкласс» в описании понятий в различных формах абстракции представления знаний;
- принцип интеграции онтологического анализа и семантического моделирования предметной области на основе иерархии понятий;
- принцип введения понятий в онтологии предметной области на базе объектно-ориентированного моделирования процесса управления в проблемных ситуациях;
- принцип осмысления понятия сходства в семантических сетях на основе кластерного анализа терминов предметной области и разработки онтологии.

Объектно-когнитивный анализ предметной области включает следующие основные этапы. Вначале, в соответствии с методологией объектно-ориентированного анализа, выделяется множество значимых сущностей из этой области (множество классов и объектов). Затем идентифицируются значимые отношения, которые существуют между классами и объектами предметной области. На следующем этапе определяется, какие операции взаимодействия объектов представляются важными, и моделируется

поведение объектов. По результатам моделирования на основе онтологического анализа разрабатывается предметно-ориентированная онтология. В заключение значимые отношения оформляются синтаксически, то есть при помощи аксиом.

При моделировании ИСППР предложено использовать специальные методы и средства объектно-ориентированного моделирования предметной области, разработанные для проектирования информационных систем, чтобы воссоздать концептуальную модель экспертов в формализованной модели представления знаний. Разработаны принципы представления знаний с использованием языка объектно-ориентированного моделирования Unified Modeling Language (UML). Объектный подход содержит набор моделей, связанных с понятием класса/объекта, объединяющего данные (состояние) и поведение. На рис. 2 показана концептуальная модель ИСППР, устанавливающая отношения между вышеперечисленными компонентами представления знаний.

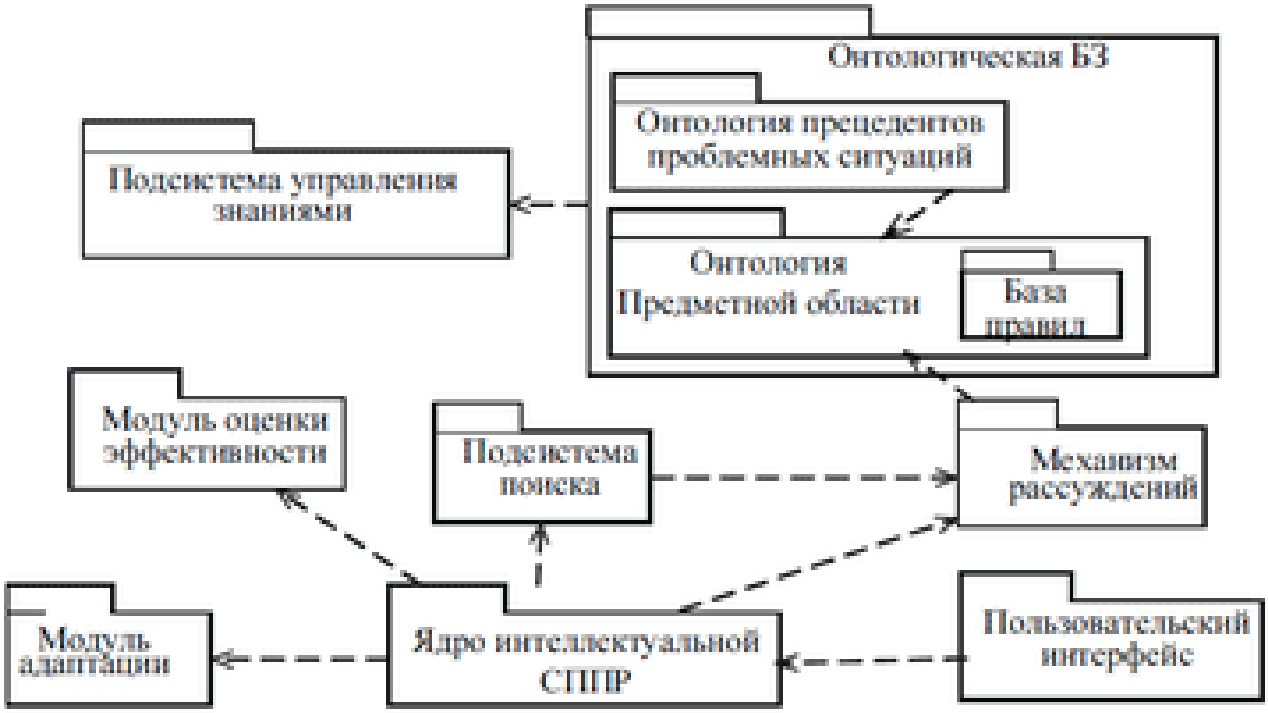


Рис.2. Модель архитектуры интеллектуальной системы поддержки принятия решений

Комплекс моделей является основой разработки базы знаний информационной системы поддержки принятия решений. Предложены методы и средства преобразования моделей в различные объектно-ориентированные нотации программирования и хранения знаний.

Таким образом, результатами объектнокогнитивного анализа являются формальные описания отношений между абстрагированными понятиями и сущностями, являющимися базовыми объектами предметной области (когнитивными элементами).

Модель семантического анализа проблемной ситуации можно представить в виде семантической сети (рис. 3), позволяющей определить цели моделирования ИСППР и сформулировать требования к результатам моделирования. Семантическая сеть обобщенная модель предметной области, имеющая вид графа *CC(A, R)*, в котором вершины ai∈A соответствуют объектам предметной области, а дуги *rj∈R* – отношениям между ними, *A≠Ó, R≠Ó*. В семантической сети соединено как метазнание, так и семантическое предметное знание.

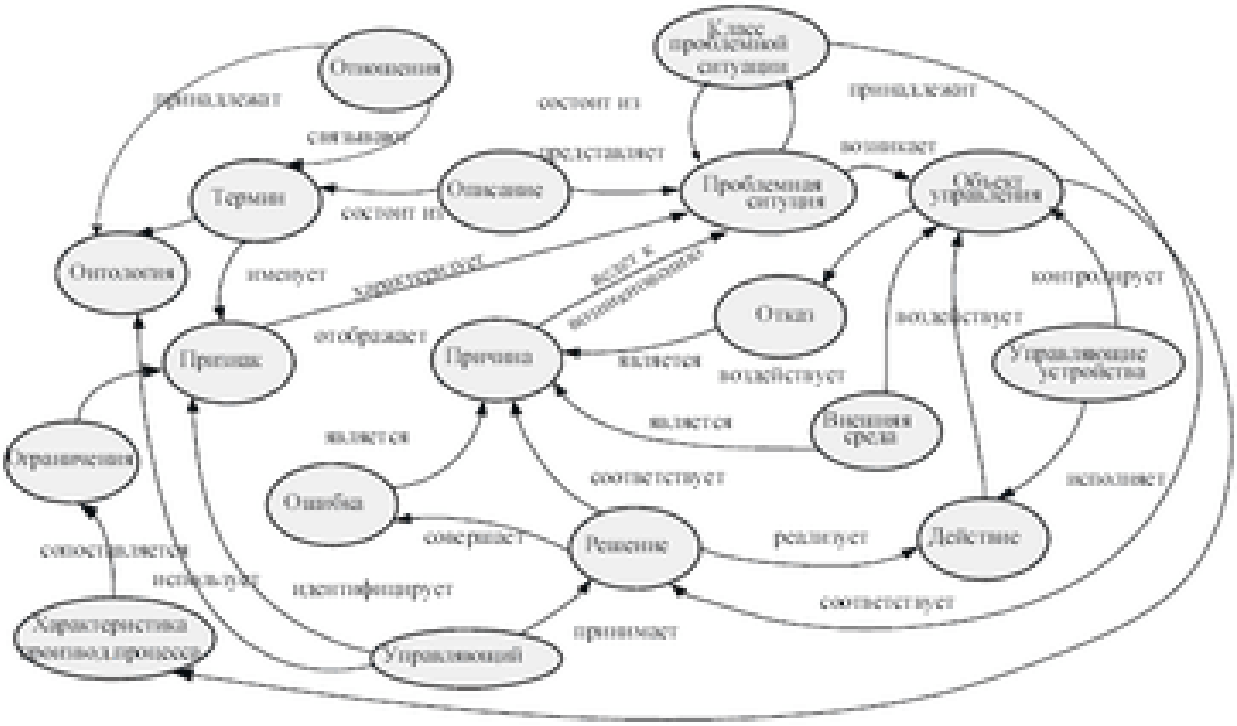


Рис.3. Семантическая сеть процесса управления в проблемной ситуации

Конкретизация обобщенного представления проблемной ситуации в рассматриваемой предметной области, особенно в части логических отношений, позволяет использовать логическую дедукцию для построения систем принятия решений. Обработка знаний при таком подходе базируется на использовании интерпретаторов семантических сетей, основанных на инференциальных (дедуктивных) возможностях семантических сетей и реализующих механизмы логического вывода, механизмы поиска, анализа, определения и конструирования объектов, составляющих семантическую сеть.

В семантической сети можно выделить отношения, которые особенно важны для представления и поиска знаний в системе принятия решений: логические отношения, каузальные отношения, отношения синонимии, зависимости и реализации.

Эти отношения характеризуют совместимость отдельных событий или фактов в проблемной ситуации, одновременность или последовательность явлений и т. д.

Они позволяют строить причинно-следственные связи, процедуры проверки целостности и непротиворечивости знаний. Для моделирования этих отношений требуются специальные модели представления интенсиональных знаний, подобные моделям, используемым в базах знаний.

Интеграция показанных выше моделей представления знания осуществляется на основе онтологического анализа.

Онтологический анализ – аналитическая работа с целью определения и объединения релевантных информационно-логических и функциональных

аспектов исследуемой системы в соответствующей содержательной онтологии [3]. Онтологический анализ направлен на исследование и интерпретацию системных связей в сложных предметных областях с применением методов и средств компьютерного моделирования.

Современное определение термина «онтология» в теории искусственного интеллекта неодназно;для практического использования наиболее подходящим является определение онтологии как знаний,формально представленных на базе концептуализации. Концептуализация предполагает описание множества объектов и понятий ,знаний о них и связей между ними . Под формальной моделью M^0 онтологической системы [3] понимается триада вида :

$$M^0=<O^{meta},\{O^{app}\},Inf^F>$$

Где O^{meta} -онтология верхнего уровня (метаонтология) ;

$\{O^{app}\}$- множество предметных онтологий и онтологий задач предметной области ; Inf^F-модель машины вывода ,ассоциированной с онтологической системой M^0. Сущностями метаонтологии $Onto^{meta}$ являются такие понятия ,как «объект» , «атрибут» , «значение» , «отношение» и т.п .Наполненная предметная онтология $Onto_i^{app}$ может рассматриваться как как компонент базы знаний при работе с конкретной предметной областью и являться, в свою очередь, шаблоном для построения динамического компонента базы знаний ,изменяющегося при переходе от исследования одной конкретной задачи к другой.

3.5. Концепция информационной поддержки принятия реений при управлении образовательным процессом.

3.2.концепция информационной поддержки принятия решений при управлении образовательным процессом

Одной из наиболее актуальных задач в области образования в современных условиях является повышение его качества .Эта проблема напрямую связана с содержанием образования и технологией реализации образовательных программ .Знания -главная ценность в деятельности университета .Университетские знания представляют собой методические разработки преподавателей ,результаты участия научных сотрудников университета в

научно -исследовательских работах .Эффективное использование накопленных за многие годы знаний несомненно повышает качество образовательных процессов .

Существующие подходы к решению проблемы повышения качества образования состоят в повышении качества преподавания, совершенствовании управления образовательными процессами .Решение проблемы повышения качества образования ,во-первых, подразумевает совершенствование учебно-методического обеспечения и составляющих образовательного процесса . необходимо использование современных интеллектуальных технологий, так как большинство критериев качественные,а не количественные .Во-вторых ,совершенствование управления качеством образования возможно путем внедрения системы виртуального образования: разработка элктронных учебных курсов ,электронных тестов ,тренироваочных заданий .

На сегодня большинство методического материала университета представляет собой архив методических литературных изданий .Эти знания не используются в полном объеме .Необходимо активизировать имеющиеся знания с целью порождения новых знаний. Для решения поставленной задачи предлагается разработка и внедрение интеллектуальной системы поддержки принятия ренений на основе базы знаний в систему управления образовательным процессов вуза на основе методологии лбъектно-когнитивного анализа ,интегрирующего методы и результаты объектно-ориентированного анализа ,онтологического анализа и семантической сети представления знаний .

Структурирование знаний о качестве образовательного процеса предлагается производить на основе онтологического анализа ,в основе которого лежит описание предметной области в терминах сущностей ,отношений между ними ,и действий над сущностями .Таким образом ,в процессе структуризации онтологя образовательного процесса представляется в виде иерархической системы .

Онтология задает единое информационное пространтство ,в котором интегрируются различные модели представления знания об образовательном процессе, знания о конкретной области подготовки специалистов ,представленное в форме онтологии $Onto^{edu}$ и $Onto_i^{app}$,правила управления образовательным процессом и прецеденты конкретных проблемных ситуаций , требующих принятия решений. Онтология образовательного процесса включает онтологию верхнего уровня (метаонтологию) $Onto^{meta}$,онтологию высшего образования $Onto^{edu}$ и предметные онтологии подготовки специалистов по различным специальностям $Onto_i^{app}$:

$Onto = <Onto^{meta}, Onto^{edu}, \{Onto_i^{app}\}, Inf^F>$,

Где Inf^F-модель машины вывода ,ассоциированной с онтологической системой *Onto*. Сущностями метаонтологии являются такие понятия ,как «объект» , «атрибут» , «значение» , «отношение» и т.п. Онтология $Onto^{edu}$ оперирует понятиями ,характеризующими процесс подготовки специалистов

высшего образования (например, «университет» , «преподаватель» , «обучаемый» , «учебный курс» и т.п.) и организуется по образцу метаонтологии $Onto^{meta}$. Предметная онтология $Onto_i^{app}$ содержит понятия ,характеризующие семантику особенностей подготовки специалистов по конкретным специальностям («образовательный стандарт», «дисциплина специализации» , «информационная технология» и др.) ,структурированные в соответствии с иерархией ,установленной для конкретной специальности. Структура рассматриваемых предметных областей предполагает наличие в онтологии отношений наследования ,статической агрегации , а также нескольких типов ассоциативных парадигматических отношений : отношений агрегации ,причинно-следственных отношений ,отношений сходства ,отношений семантического подобия .Наполненная предметная онтология $Onto_i^{app}$ может рассматриватьсякак компонент базы знаний при работе с конкретной предметной областью и являться ,в свою очередь, шаблоном для построения динамического компонента базы знаний ,изменяющегося при переходе от исследования одной конкретной задачи к другой .

Современная обучающая среда должна быть адаптивной, что обусловливает соответствующие требования к формированию ,наполнению ,использованию ,адаптации и обучению базы знаний ИСППР. Для ИСППР университета потребовалось несколько предметных онтологий, в соответствии с перечнем специальностей подготовки специалистов .

Онтология, в частности, необходима для того, чтобы пользователь мог работать с базами данных методических материалов, относящихся к широкому кругу специальностей ,разрабатывать методическое обеспечение для новых специальностей на основе повторного использования методик обучения фрагментам знаний ,освоенных преподавателями университета .

Онтологический подход к разработке базы знаний ИСППР нашел свое применение в разработке первой версии онтологии учебного процесса по специальности «Управление качеством» на основе стандартного моделирования UML и разработаны диаграммы классов ,моделирующих соответствующие фрагменты знаний .

Для разработки онтологии использовались средства построения онтологии Stanford’s Protégé 3.1.1 с OWL и SWRL-дополнениями для кодирования онтологии и базы знаний .

Формализация правил осуществлена на основе дескриптивной логики , а прецеденты являются экземплярами классов онтологии и представляют собой совокупность объектов .Вывод на прецедентах производится на принципах аналогии и соответствует CBR-принципам.

Далее приведен пример использования образовательной базы знаний для информационной поддержки образовательног процесса . В таблице показаны способы использования онтологии образовательного процесса для подготовки специалиста по конкретной специальности в виде тематической декомпозиции электронного учебного курса и разработки тестов . $Onto^{edu}$ -

разработка электронного учебного курса и учебно-тренировочных заданий к нему. $Onto_i^{app}$- тематическая декомпозиция ,содержания учебных фрагментов. $Onto^{ease}$-поиск решений в проблемной ситуации (разработка новых учебных курсов)на основе прецедентов методических разработок университета .

База знаний , разработанная на основе объектно-когнитивного подхода , может использоваться для создания показателей качества , для разработки учебных курсов.

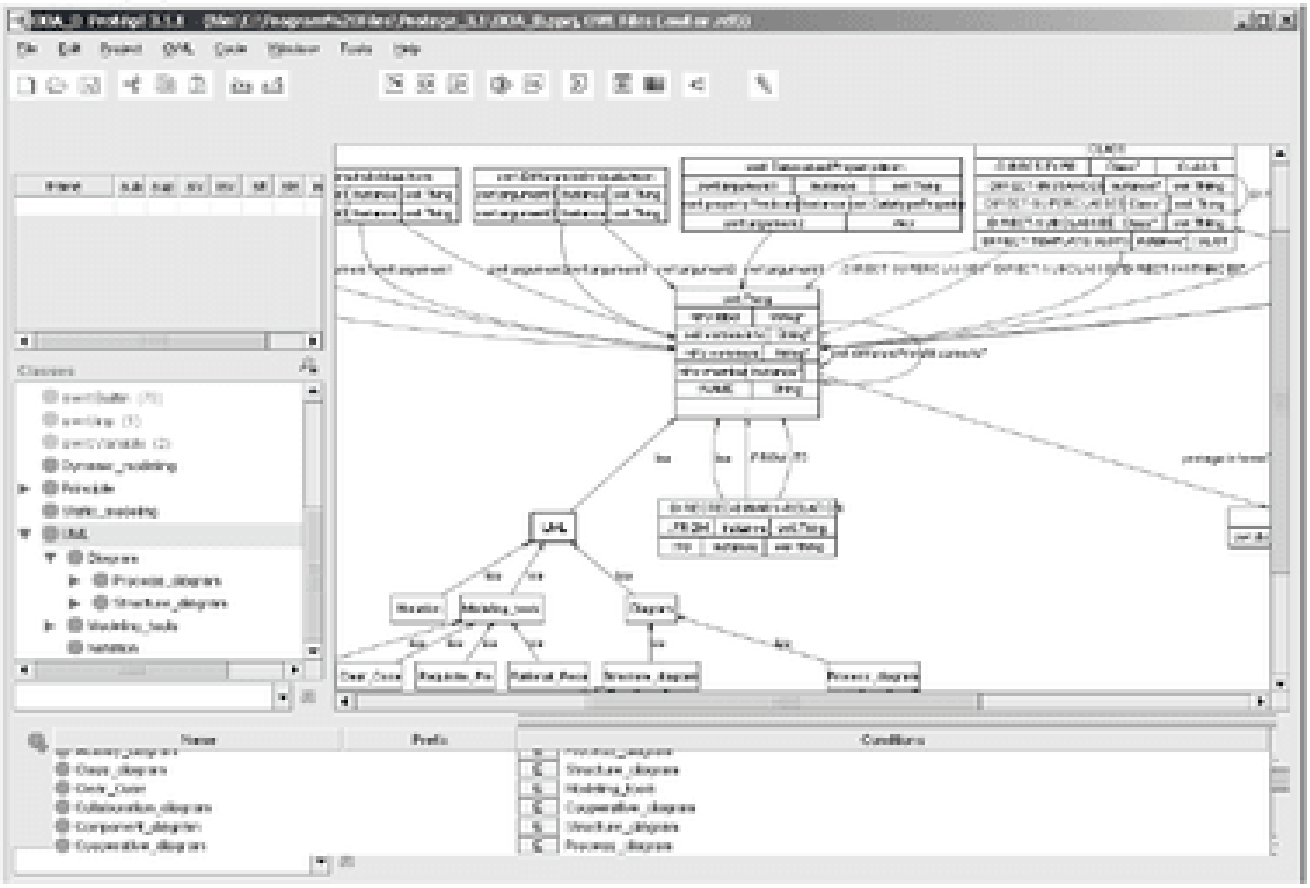


Рис. 6. Разработка онтологии с использованием средства Stanford's Protégé 3.1

Использование в учебном процессе электронных учебных курсов разработанных на основе базы знаний ,способствует повышению качества обучения за счет развития дидактики и методики создания новых форм обучения ;снижению затрат на организацию и проведение образовательного процесса ,перераспределению нагрузки преподавателей с рутинной на творческую деятельность;индивидуальную работу с обучаемыми;повышению опреативности обеспечения учебного процесса учебно -методическими средствами при изменении структуры и содержания обучения.

Глава 4. Системы для решения прецедентых задач.

4.1. Механизм оптимального выбора прецедента при наблюдении ситуационного вектора с количественными координатами

Процесс оперативного вывода по прецеденту является сложным и однозначным. Сложность оперативного вывода по прецеденту и прогнозирования его последствий усугубляется тем, что этот процесс практически всегда осуществляется в условиях действия факторов неопределенности и риска. Основной задачей управления интегральными схемами, например, чипами или микрочипами, является обеспечение помехоустойчивости, решение которой в условиях перекрываемости сигналов весьма затруднено [1].

Для интеллектуальной поддержки принятия решений на основе прецедентов проблемных ситуаций используется правила распознания класса прецедентов, к которому относится текущая поблемная ситуация [2].

В работе Прохорова М.Д. и Федунова Б.Е. [3] был введен алгоритм выбора прецедента при наблюдении ситуационного вектора с количественными координатами. Основным содержанием от результатов работы [3] является описание матрицы знаний и вывод прецедента, основанный на метод логического вывода максимин. Наши результаты отличаются от [3] тем, что выбор прецедента основывается на методе логического вывода минимакс.

Допустим, имеем матрицу знаний по прецедентам, которая была представлена в работе [3].

Представим алгоритм определения функции принадлежности μ_{dj} (x_1, ... x_i, .. x_n) прецедента d_j, интерпретируемого как нечеткое множество на универсальном множестве. $Ud = Ux_1 \times \dots Ux_i \times \dots \times Ux_n$, где Ux_i – универсальное множество, на котором заданы термины лингвистической переменной x_i, а Ud – декартово произведерние универсальных множеств Ux_i.

Рассмотрим принадлежности нечеткого множества, которое образно следующим отношением $\mu_{a1}{}^{j1}(x_1)$ ∨... ∨$\mu_{a1}{}^{j1}(x_i)$ ∨ … $\mu_{a1}{}^{j1}(x_n)$, где через «∨» обозначена операция «max» или в терминах математической логики предикат конъюнкции.

Анализируя весь блок логических высказываний, относящейся к прецеденту d_j (блок соответствующих строк матрицы знаний), замечаем, что они представляют собой объединение соответствующих нечетных множеств, образовавшихся при рассмотрении строк выделенного блока. Функция принадлежности этого объединения, отождествляемая с функцией принадлежности прецедента d_j, в соответствии [7] , будет:

$\mu_{dj}(x_1, \dots, x_n) = (\mu_{a1}{}^{j1}(x_1)$ ∨... ∨ $\mu_{an}{}^{j1}(x_n)) \wedge \dots \wedge (\mu_{a1}{}^{jkj}(x_1)$ ∨... ∨ $\mu_{an}{}^{jkj}(x_n))$

где через «∧» обозначена операция «min» , или в терминах математической логики предикат дезъюнкции.

Формально представленный алгоритм определения функции принадлежности прецедента d_j можно записать в виде:

фиксируем произвольную точку $(x_1^*, \dots, x_n^*) \in U_{x1} \times \dots \times U_{xn}$,

для каждого блока матрицы знаний, соответствующего d_j, определяем μ_{dj} $(x_1, \ldots, x_n)$ в этой точке согласно схеме табл. 1.

Таблица 1.
Выбор принадлежности прецедента

n/n	Координаты ситуационного вектора			Max	min	d
	x_1	...	x_n			
:	:	:	:	:	:	:
$\mathbf{J_1}$: $\mathbf{J_{Kj}}$	$(a_1^{j1})^*$: $(a_1^{\mathbf{Jkj}})^*$	… : …	$(a_n^{j1})^*$: $(a_n^{\mathbf{Jkj}})^*$	$\max (a_i^{j1})^*$: $\max (a_i^{\mathbf{Jkj}})^*$	$\min_j \max_i (a_i^j)^*$	μ_{dj}
:	:	:	:	:	:	:

Операция $\min_i a_i^{js}$ производится над числами, стоящими в строках "*i*", $1 \le i \le n$ и в столбец "min" заносится минимальное число в соответствующей строке.Операция $\min_j \max_i (a_i^j)^*$выделяет среди полученных строчных минимумов $1 \le j_s \le K_j$ наибольший. Это число и является значением функции принадлежности $\mu_{dj}(x_1, \ldots x_n)$ в этой фиксированной точке $(x_1^*, \ldots x_n^*)$. Проведя такие вычисления для каждой точки универсального множества, получим интересующие нас функции принадлежности.

На основе текущих измерений точка $(x_1^*, \ldots x_n^*)$ формируется с количественными значениями его координат. Только в этой фиксированной точке $(x_1^*, \ldots x_n^*)$ в момент поступления замера и нужно определить значение функции $\mu_{dj}(x_1, \ldots x_n)$.

При наблюдении ситуационного вектора с количественными координатами (все координаты вектора измерены по числовым шкалам) для выбора наиболее подходящего прецедента вовсе нет необходимости полностью определять функции принадлежности $\mu_{dj}(x_1, \ldots x_n)$ на всем множестве точек универсального множества. Достаточно рассчитать их значение только для фиксированных числовых значений координат вектора, который мы получили в результате наблюдения. Для этого придется однократно воспользоваться алгоритмом беря в качестве $(x_1^*, \ldots x_n^*)$ координаты наблюдаемого ситуационного вектора.

В результате для каждого прецедента d_j мы получим число $d_j(x_1^*, \ldots x_n^*)$, являющееся степенью принадлежности d_j точке $(x_1^*, \ldots x_n^*)$.

Исходя из такой интерпретации, наиболее предпочтительным прецедентом для разрешения наблюдаемой ПрС/С будет прецедент d_j^*, для которого

$$d_j^*(x_1^*, \ldots x_n^*) = \min_{1 \le s \le p} d_j(x_1^*, \ldots x_n^*)$$

В отличие от [5] в оперативно реализуемом механизме вывода по прецеденте нет необходимости «принудительно» вводить метрику в алгоритм предпочтения прецедентов.

Таким образом, в идею алгоритма входит метод минимакса – правило принятия решений, используемое в теории игр, теории принятия решений, исследовании операций, статистике и философии [6,с.149;7, с 144; 8, с.45]

Систему работы RFID можно представить как процесс, посредством которого собираются данные в определенный момент времени. Причем не редко встречаются ситуации, когда в определенный момент времени одновременно поступают данные от n меток [1], случайным образом перекрывающихся друг друга.

Согласно работ [9,10] , поведенческая модель радиоприемного тракта (RX chain) состоит из Verilog – модулей, которые реализуют процессы детектирования поднесущей, фильтрации, усиления и детектирования входного высокочастотного сигнала, представленного в языке Verilog 16 – разрядным двоичным сигналом. Иными словами, сигнал по форме является вектором **x** объема 16, связанных с влиянием перекрывающих сигналов. То есть на искажение влияют **n** перекрывающихся сигналов. Допустим, что истинный сигнал представим в виде вектора I_0, на который наложили искажение **u,** состоящее из **n** сигналов (векторов), принимающие значения из множества $I_1, I_2,\ldots, I_d$.

Данные от **n** меток, поступающие в определенный момент времени одновременно, описываются ситуационным вектором с координатами $\{x_1, x_2, \ldots, x_n\}$, представленными лингвистическими переменными:

- x_1 – энергетика сигнала от метки № 1;
- x_2– энергетика сигнала от метки № 2; и так далее;
- x_n– энергетика сигнала от метки № **n**

Пусть для этого класса наблюдались два прецедента d_1, d_2, каждый из которых применялся в двух (разных) случаях. Лингвистические переменные принимают следующие унифицированные значения (термы): x_i = { статический низкий уровень, статический высокий уровень}, i = 1, …, n.

Пусть каждый терм лингвистических переменных представлен унифицированным универсальным множеством (десятибалльная шкала) и унифицированными кусочно – линейными (треугольными) функциями принадлежности. Допустим, например, матрица знаний для этой системы представлена в табл. 2, при n = 3.

Таблица 2

Пример выбора принадлежности прецедента

№ п/п	Координаты ситуационого вектора			Max	Min
	x_1	x_2	x_3		
1.1	0,6	0,8	0,4	0,8	0,8
1,2	0,6	0,2	0,8	0,8	
1.3	0,6	0,8	0,6	0,8	
1.4	0,9	0,8	0,8	0,9	

2.1 2.1	0,4 0,6	0,4 0,2	0,5 0,6	0,5 0,6	0,5

Вывод: наиболее предпочтительным прецедентам для разрешения ПрС/С (x_1^*= 0,4, x_2^*= 0,4, x_3^*= 0,6) является прецедент d_2.

Таким образом , сформулирован алгоритм оперативного вывода по прецеденту с применением в управлении интегральными схемами.

4.1.1. Вероятностная модель процесса выбора прецедента

В работе Прохорова М. Д. и Федунова Б.Е. [3] был введен алгоритм выбора прецедента при наблюдении ситуационного вектора с количественными координатами.

Пусть состояние ПрС/С описывается ситуационным вектором с координатами ($x_1, \ldots, x_n$) и каждая координата x_i - лингвистическая переменная с множеством термов прецедентам (блок прецедента). Каждая строка матрицы представляет собой конкретный ситуационный вектор, при котором в прошлом успешно реализовался соответствующий прецедент.

Таблица 1

№ п/п	Координаты ситуационного вектора				Прецедент
	x_1	x_2	…	x_n	
1.1 : $1.k_1$	a_1^{11} : $a_1^{1k_1}$	a_2^{11} : $a_2^{1k_1}$	… : …	a_n^{11} : $a_n^{1k_1}$	d_1
:	:	:	:	:	:
$m.1$: $m.k_m$	a_1^{m1} : $a_1^{mk_1}$	a_2^{m1} : $a_2^{mk_1}$	… : …	a_n^{m1} : $a_n^{mk_1}$	d_m

Перенумеруем строки блока прецедента d_j двумя индексами: первый индекс – номер прецедента (здесь он является номером блока), второй индекс – порядковый номер ситуационного вектора в этом блоке.

Полученную упорядоченную таким образом систему логических высказываний называют нечёткой матрицей знаний или просто – матрицей знаний.

На основе текущих измерений точка $(x_1^*, \ldots, x_i^*, \ldots, x_n^*)$ формируется с количественными значениями его координат. Только в этой фиксированной точке $(x_1^*, \ldots, x_i^*, \ldots, x_n^*)$ в момент поступления замера и нужно определить значение функции принадлежности $\mu_{dj}(x_1, .. x_i, .. x_n)$.

Допустим, что имеем прецедент d_j, который может быть получен из следующих возможных ситуационных векторов $\mathbf{a}_{j.1} = (a_1^{j1}, a_2^{j1}, \ldots, a_n^{j1})$, $\mathbf{a}_{j.2} = (a_1^{j2}, a_2^{j2}, \ldots, a_n^{j2})$, …, $\mathbf{a}_{j.k_1} = (a_1^{jk_1}, a_2^{jk_1}, \ldots, a_n^{jk_1})$.

Очевидно, что ситуационный вектор принимающий значения из множества $\{\mathbf{a}_{j.1}, \mathbf{a}_{j.2}, \ldots, \mathbf{a}_{j.k_j}\}$ является реализацией случайного вектора $\mathbf{A} = (A_1, A_2, \ldots, A_n)$, и каждый элемент a_i^{jx} (i=1,…,n) ситуационного вектора $\mathbf{a}_{j.x}$ принимает значения из множества Ω объема N_i. Допустим, что вероятность того, что i-й элемент (i=1,…,n) принимает значение a_i^{jx} есть $p(a_i^{jx})$, причем $\sum_{\alpha=1}^{N_i} p(\omega_\alpha) = 1, \quad (\omega_\alpha \in \Omega).$ Введем следующее обозначение

$$r_l(a_i^{jx}) = \begin{cases} 1, & \text{если } a_i^{jx} = \omega_l, \\ 0, & \text{иначе.} \end{cases}$$

другими словами, имеем вектор $\mathbf{r}_i(a_i^{jx}) = (r_1(a_i^{jx}), r_2(a_i^{jx}), \ldots, r_{N_i}(a_i^{jx})).$

Теорема 1. Вероятность того, что прецедент D_j примет значение d_j определяется по формуле

$$\mathrm{P}(D_j = d_j) = \sum_{x=1}^{k_j} \prod_{i=1}^{n} p(a_i^{jx}).$$

С использованием формул (см., например, [12-13]) комбинаторики и формулы (2) имеем справедливость представленной теоремы.

Тогда представленное распределение можно представить как

$$\mathrm{P}(D_j = d_j) = \sum_{x=1}^{k_j} \prod_{i=1}^{n} N_i! \prod_{l=1}^{N_i} \frac{p^{r_l(a_i^{jx})}}{r_l(a_i^{jx})!}.$$

На практике, как правило, вероятности $p(\omega_\alpha)$ (α=1,…,N_i, i=1,…,n) не известны. Следовательно, формулы (3) и (3’) не находят фактического применения.

Допустим, что имеются реализации s прецедентов $d_1, d_2, \ldots, d_s$. Иначе говоря, ряд фактических данных $\mathbf{d}=\{d_1, d_2, \ldots, d_s\}$ можно трактовать как реализацию выборки объема s, элементы которой подчиняются представленному распределению. Пусть для каждого j=1,…, μ, где $\mu = \prod_{i=1}^{s} k_i$, существует вектор$\mathbf{z}_j=(z_{1j},..., z_{dj})$, определяемый как $\mathbf{z}_j = \sum_{i=1}^{s} \mathbf{r}_i^{jx}$, причем индексы в правой и левой части связаны между собой взаимно однозначным соответствием, которое не единственно.

Теорема 2. Элементы множества $W(d, \mathbf{z})=\{W(d, \mathbf{z}_1), \ldots, W(d, \mathbf{z}_\mu)\}$ являются несмещенными оценками для вероятности $\mathrm{P}(D = d)$, которые при j=1, …, μопределяются как

$$W(d,z_j)=\sum_{x=1}^{k}\prod_{i=1}^{n}\prod_{l=1}^{N_i}\frac{\begin{pmatrix} z_{\omega_{x_j}} \\ r_l(a_i^{jx}) \end{pmatrix}}{\begin{pmatrix} kN_i \\ N_i \end{pmatrix}}.$$

Приведенная теорема доказывается аналогично доказательству приведенной в работе [11].

Итак, имеем множество несмещенных оценок вероятности проявлений искажений. Наиболее подходящая несмещенная оценка $W(d, \mathbf{z}_g)$ для вероятности оправдываемости метеорологического прогноза d $P(D=d)$ распределения определяется из всего множества полученных несмещенных оценок $W(d, \mathbf{z})=\{W(d,\mathbf{z}_1), \ldots, W(d, \mathbf{z}_\mu)\}$**,** согласно определениям.

Определение 1. Решение $\mathbf{z}_g$, основанное на наблюдении, является наиболее подходящим из множества $\mathbf{z}=\{\mathbf{z}_1, \ldots, \mathbf{z}_\square\}$, если

$$\prod_{i=1}^{s} W(\mathbf{x}_i, \mathbf{z}_g) = \max_{j=1,K,\mu} \prod_{i=1}^{s} W(\mathbf{x}_i, \mathbf{z}_j),$$

где при i=1, … , s элементы множества $W(\mathbf{x}_i, \mathbf{z})=\{ W(\mathbf{x}_i, \mathbf{z}_1), \ldots, W(\mathbf{x}_i, \mathbf{z}_\square)\}$ являются несмещенными оценками для вероятности $P(D=d)$ распределения , определенными ранее.

Определение 2. Несмещенная оценка $W(\mathrm{d}, \mathrm{z}_g)$ для вероятности $P(D=d)$ распределения является наиболее подходящей из всего множества несмещенных$W(d, \mathbf{z})=\{W(d, \mathbf{z}_1), \ldots, W(d, \mathbf{z}_\mu)\}$, определяемых , если $\mathbf{z}_\mathrm{g}$ – наиболее подходящее решение, основанное на наблюдении.

Теорема 3. Наиболее подходящая несмещенная оценка $W(\mathbf{u}, \mathbf{z}_g)$ для вероятности $P(\mathbf{U}=\mathbf{u})$ модели является состоятельной, асимптотически нормальной и асимптотически эффективной.

4.2. Механизм вывод прецедента по введенному ситуационному вектору

Целью исследования является разработка и обоснование алгоритмов решения задач с программной реализации алгоритмической оболочки скорейшего вывода прецедента. Использование методов теоретического анализа литературы и математического моделирования авторы определяют механизмы вывода, основанные на прецедентах и изучения матричных прецедентов знаний. Кроме того, исследование показывает прецедент алгоритм выбора под руководством ситуационный векторных количественных координат. Практическая ценность в том, что представленные материалы могут быть полезны для решения текущих проблем с функционированием сессии общих целей антропоцентрического объекта.

Современные технические антропоцентрические объекты, действующие в сложных предметных областях (самолеты, в частности боевые самолеты, транспортные средства, операторы сложных стационарных

объектов), не мыслятся без интеллектуальной поддержки их экипажей. (Roudavski и Маккормак, 2016 г.). Важно отметить, что такие интеллектуальные системы должны работать согласованно с активизированной концептуальной моделью экипажа, помогая ему решать текущие задачи с учетом генеральной задачи сеанса функционирования антропоцентрического объекта. В ряде таких бортовых интеллектуальных систем не обойтись без оперативного вывода по прецеденту.

Антропоцентрическим объектом (Антр/объектом) называется его оболочка и реализованная в ней совокупность его макросоставляющих (борт Антр/объекта) в составе:

- бортовых измерительных устройств (иногда и бортовых комплексов измерительных устройств), получающих информацию о внешнем и внутри бортовом мире,
- *системообразующего ядра* Антр/объекта, в котором (ядре) главенствующая роль принадлежит команде операторов (экипажу),
- бортовых исполнительных устройств (бортовых комплексов исполнительных устройств), воздействующих на внешний и внутри бортовой мир. (Федуном, 2005; Roudavski & Маккормак, 2016; Желтов и Федунов, 2016 г.).

В настоящее время осознано, что совершенствование «аппаратной составляющей» разрабатываемых антропоцентрических объектов (Антр/объект) *недостаточно* для желаемого резкого повышения эффективности их использования. (Федунов, 2011). Достичь этого возможно путем направления усилий конструкторов и ученых на совершенствование интеллектуальной составляющей «системообразующего ядра» бортового комплекса Антр/объекта – совокупности алгоритмов бортовых цифровых вычислительных машин (БЦВМ-алгоритмы) и алгоритмов деятельности экипажа, которая сейчас называется «бортовым интеллектом» и которая из набора разрозненных систем бортового оборудования создает функционально целостный комплекс, нацеленный на выполнение заданных задач сеансов функционирования (Федунов, 2010). [14].

На современном Антр/объекте алгоритмическим обеспечением БЦВМ решаются задачи обработки информации, управления, применения средств воздействия на внешнюю обстановку. Задачи оперативного назначения текущей цели сеанса функционирования (задачи целеполагания) и выбора рационального способа (тактики) достижения этой цели (тактические задачи) решаются только экипажем.

Результаты проведенных научно-исследовательских работ, совершенствование бортовых вычислительных машин, измерительных и исполнительных устройств Антр/объектов дают возможность разработать и реализовать на борту разрабатываемых Антр/объектов алгоритмы и системы нового типа, которые будут способны обеспечивать решение упомянутых тактических задач (Федосов, 2005; Федунов, 1996). [15.16].

Различают два класса бортовых интеллектуальных систем:

- интеллектуальная система «Ситуационная осведомленность экипажа», предъявляющей экипажу информацию о текущей внешней и внутри бортовой обстановки (информационная модель внешней и внутри бортовой обстановки) достаточную для адекватного назначения им текущего этапа (цели) полета (решение задачи целеполагания) (Грибкова & Федуном, 2010) [17];

- бортовые оперативно советующие экспертные системы типовых ситуаций (БОСЭС ТС) функционирования Антр/объекта, оперативно с заданным темпом корректировки предъявляющих экипажу способ достижения намеченной цели этапа, с глубиной его проработки достаточной для его реализации после согласия экипажа (Федунов, 1998). [18].

В названных интеллектуальных системах используются различные механизмы вывода(Федунов, 2002) [19], оперативно решающие возникающие у Антр/объекта проблемы. Один из них – оперативный вывод, основанный на прецедентах.

Уместно отметить, что п.п. Groumpos (2013) дает обзор современных интеллектуальных систем поддержки принятия решений. Она учитывает только системы, поддерживающие задачи 2GCL и 3GCL. Автор не нашел систем, предназначенных для борьбы с 1GCLs.

4.2. 1. Механизмы вывода, основанные на прецедентах.

В исследовании использовалась метод математического моделирования. Для того, чтобы представить примеры решения проблем с программной реализации алгоритмической оболочки скорейшего вывода прецедента авторы проанализировали условия лингвистической переменной. Анализ существующих исследований позволил определить пути решения проблем с программной реализации алгоритмической оболочки скорейшего вывода прецедента.

Эти механизмы вывода применяются в проблемных субситуациях (ПрС/С), сложность которых не позволяет провести их конструктивную формализацию, но по которым имеется опыт (прецеденты) их успешного разрешения.

Одна из трудностей этого подхода состоит в правильном подборе координат ($x_{1,...,}$ x_i, .., x_n) ситуационного выбора SV(ПрС/С-решение), описывающего эту ПрС/С, как по их количеству, так и по форме представления каждой координаты. Полнота описания ситуационного вектора и связь конкретного вектора с конкретным прецедентом устанавливается при длительной работе с экспертами - действительными носителями этого знания.

Как правило, координаты ситуационного вектора суть лингвистические переменные.

Лингвистическая переменная – координата ситуационного вектора.

В [20] Лотфи Заде определил лингвистическую переменную как переменную, принимающую свои значения из заданного множества терминов или высказываний естественного языка. Последние были названы термами.

Приведём пример лингвистической переменной:
“высота” = {сверхнизкая, малая, низкая, средняя, большая}.
Здесь “высота” – название лингвистической переменной (ЛП). В скобках представлено множество её термов.

Для работы с лингвистическими переменными каждый терм следует соотнести с соответствующим нечётким множеством (Кофман, 1982; Ротштейн, 1999) [21,22], которое в свою очередь представляется через универсальное множество (универсум) и функцию принадлежности элементов этого универсального множества рассматриваемому нечёткому множеству.

Функция принадлежности принимает значения из интервала [0,1]. Она количественно оценивает степень принадлежности элемента нечёткому множеству.

Пример. Зададим терм “сверхнизкая” (С/Н) лингвистической переменной “высота”.
Универсальное множество U = [h = 0,…,100].
Функция принадлежности нечёткого множества «сверх низкая» представлена на рис. 1

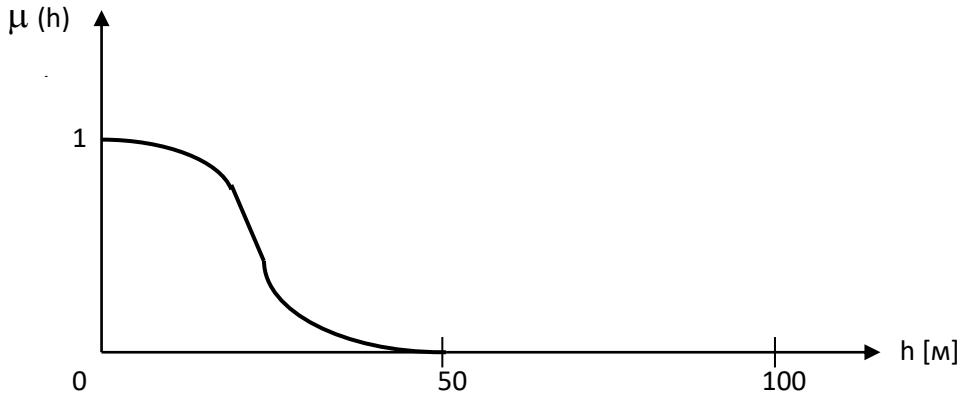


Рис.1. Функция принадлежности нечеткого множества "супер низкий"

Заметим, что как универсальные множества, так и функции принадлежности на нем назначаются по результатам (совместного с экспертами) исследования соответствующей предметной области.

При большом числе термов их функции принадлежности обычно задают в унифицированном виде. Чаще всего - это кусочно-линейная функция.

4.2. 2. Матрицы знаний по прецедентам.

Одними из характерных особенностей поставленных перед совершенствованием работы RFID являются составление и изучение

алгоритма выбора прецедента при наблюдении ситуационного вектора с количественными координатами.

Ранее в работе Прохорова М. Д. и Федунова Б.Е. [3] был введен алгоритм выбора прецедента при наблюдении ситуационного вектора с количественными координатами.

Аналогично результатам работы [3] представим матрицу знаний по прецедентам.

Пусть состояние ПрС/С описывается ситуационным вектором с координатами ($x_{1,...,}$ x_i, .., x_n) и каждая координата x_i - лингвистическая переменная с множеством термов $A_i = \left\{a_i^1,...,a_i^j,...,a_i^{K_i}\right\}$. Для некоторых конкретных реализаций ситуационного вектора, когда каждая лингвистическая переменная принимает одно из своих возможных значений (конкретный терм), есть прецедент успешного разрешения этой ПрС\С.

Пусть накоплено множество d_j, j = 1,…,p прецедентов и каждый из них связан с множеством конкретных ситуационных векторов, при которых он (прецедент) был избран.

Составим матрицу такого соответствия (табл.1). Сгруппируем строки матриц по прецедентам (блок прецедента). Каждая строка матрицы представляет собой конкретный ситуационный вектор, при котором в прошлом успешно реализовался соответствующий прецедент.

Таблица 1. Матрица композиция

• № п/п	**Координаты ситуационного вектора**					**Прецедент**
	x_1	…	x_i	…	x_n	
1.1 : 1к$_1$	a_1^{11} : $a_1^{1K_1}$	..... :	a_i^{11} : $a_i^{1K_1}$	..... :	a_n^{11} : $a_n^{1K_1}$	d_1
:	:	:	:	:	:	:
j1 : jк$_j$	a_1^{j1} : $a_1^{jK_j}$	..... :	a_i^{j1} : $a_i^{jK_j}$	..... :	a_n^{j1} : $a_n^{jK_j}$	d_j
:	:	:	:	:	:	:
p1 : pк$_p$	a_1^{p1} : $a_1^{pK_p}$	..... :	a_i^{p1} : $a_{1i}^{pK_p}$	..... :	a_n^{p1} : $a_{1n}^{pK_p}$	d_p

Перенумеруем строки блока прецедента d_j двумя индексами: первый индекс – номер прецедента (здесь он является номером блока), второй индекс – порядковый номер ситуационного вектора в этом блоке.

Введённая матрица определяет систему логических высказываний вида «<u>если</u> …, <u>то</u> …, <u>иначе</u> …». Например, строка матрицы j_1 шифрует высказывание:

Если $x_1 = a_1^{j1}$ и $x_2 = a_2^{j1}$ и … и $x_i = a_i^{j1}$ и … и $x_n = a_n^{j1}$, то d_j, (1)
иначе аналогичное выражение для следующей строки и т. д.

Полученную упорядоченную таким образом систему логических высказываний называют нечёткой матрицей знаний или просто – матрицей знаний.

4.3. Алгоритм вычисления функции принадлежности прецедента d_j.

Систему работы RFID можно представить как процесс, посредством которого собираются данные в определенный момент времени. Причем не редко встречаются ситуации, когда в определенный момент времени одновременно поступают данные от *n* меток, случайным образом перекрывающих друг друга.

Прежде всего, представим алгоритм [22] определения функции принадлежности $\mu_{dj}(x_1, .. x_i, .. x_n)$ прецедента d_j, интерпретируемого как нечёткое множество на универсальном множестве. $Ud = Ux_1 \times \ldots \times Ux_i \times \ldots \times Ux_n$, где Ux_i - универсальное множество, на котором заданы термы лингвистической переменной x_i, а Ud – декартово произведение универсальных множеств Ux_i.

Каждое логическое высказывание типа (1) или, что тоже, каждая строка матрицы знаний представляет собой нечётное отношение соответствующих нечётких множеств. Так для (1.) это будет $a_1^{j1} \times a_2^{j2} \times \ldots \times a_n^{jn}$

Функция принадлежности нечёткого множества, образованного этим нечетким отношением в соответствие с А. Кофман (1982) и А. П. Ротштейн (1999) [21,22] будет

$$\mu_{a_1^{j1}}(x_1) \wedge \ldots \wedge \mu_{a_i^{j1}}(x_i) \wedge \ldots \wedge \mu_{a_n^{j1}}(x_n),$$

где через "$\wedge$" обозначена операция "min".

Анализируя весь блок логических высказываний, относящийся к прецеденту d_j (блок соответствующих строк матрицы знаний), замечаем, что они представляют собой объединение соответствующих нечётких множеств, образовавшихся при рассмотрении строк выделенного блока. Функция принадлежности этого объединения, отождествляемая с функцией принадлежности прецедента d_j, в соответствии с А. Кофман (1982) и А. П. Ротштейн (1999) [21,22] будет

$$\mu_{d_j}(x_1,...,x_i,...,x_n) = (\mu_{a_1^{j1}}(x_1) \wedge \ldots \wedge \mu_{a_i^{j1}}(x_i) \wedge \ldots \wedge \mu_{a_n^{j1}}(x_n)) \vee$$

$$\ldots \vee (\mu_{a_1^{jK_j}}(x_1) \wedge \ldots \wedge \mu_{a_i^{jK_j}}(x_i) \wedge \ldots \wedge \mu_{a_n^{jK_j}}(x_n))$$

где через «$\vee$» обозначена операция "max".

Формально представленный алгоритм определения функции принадлежности прецедента d_j можно записать в виде:

а) фиксируем произвольную точку $(x_1^*,...,x_i^*,...,x_n^*) \in U_{x_1} \times \ldots \times U_{x_i} \times \ldots \times U_{x_n}$,

б) для каждого блока матрицы знаний, соответствующего d_j, определяем μ_{dj} (x_1, .. x_i, .. x_n) в этой точке согласно схеме таблицы 2.

Таблица 2. Матрица Фиксированная точка блок

.

№ п/п	• ***Координаты ситуационного вектора***					min	Max	d
	x_1		x_i		x_n			
:	:	:	:	:	:	:	:	:
j_1 : j_s : j_{K_j}	$(a_1^{j1})^*$: $(a_1^{js})^*$: $(a_1^{jK_j})^*$	..... : ..... :	$(a_i^{j1})^*$: $(a_i^{js})^*$: $(a_i^{jK_j})^*$	..... : ..... :	$(a_n^{j1})^*$: $(a_n^{js})^*$: $(a_n^{jK_j})^*$	$\min_i (a_i^{j1})^*$ ... $\min_i (a_i^{is})^*$ ... $\min_i (a_n^{jK_j})^*$	$\max_{js} \min_i (a_i^{js})^*$	μ_{d_j}
:	:	…… …	:	…… …	:	……	……..	:

Заметим, что для фиксированной точки (x_1^*, ..., x_i^*, ..., x_n^*) блок матрицы, представленный в табл. 2., является числовым, так как вместо каждого терма a_i^{js} из этого блока поставлено значение его функции принадлежности $(a_i^{js})^*$, вычисленное в соответствующей x_i^*. Операция $\min_i a_i^{js}$ производится над числами, стоящими в строках “i”, $1 \le i \le n$ и в столбец “min” заносится минимальное число в соответствующей строке. Операция $\max_{jS} \min_i a_i^{js}$ выделяет среди полученных строчных минимумов $1 \le j_s \le K_j$ наибольший. Это число и является значением функции принадлежности μ_{dj} (x_1, .. x_i, .. x_n) в этой фиксированной точке $(x_1^*,...,x_i^*,...,x_n^*)$. Проведя такие вычисления для каждой точки универсального множества, получим интересующие нас функции принадлежности.

4.3.1. Алгоритм выбора прецедента при наблюдении ситуационного вектора с количественными координатами

При наблюдении ситуационного вектора (Ротштейн, 1999) [19] с количественными координатами (все координаты вектора измерены по числовым шкалам) для выбора наиболее подходящего прецедента вовсе нет необходимости полностью определять функции принадлежности μ_{dj}(x_1, .. x_i, .. x_n) на всем множестве точек универсального множества. Достаточно рассчитать их значение только для фиксированных числовых значений координат вектора, который мы получили в результате наблюдения. Для этого придется однократно воспользоваться алгоритмом п. 1.3, беря в качестве $(x_1^*,...,x_i^*,...,x_n^*)$ координаты наблюдённого ситуационного вектора.

В результате для каждого прецедента d_j мы получим число $d_j(x_1^*,...,x_i^*,...,x_n^*)$, являющееся степенью принадлежности d_j точке $(x_1^*,...,x_i^*,...,x_n^*)$.

Исходя из такой интерпретации, наиболее предпочтительным прецедентом для разрешения наблюдаемой ПрС/С будет прецедент d_j^*, для которого

$$d_j^*(x_1^*,...,x_i^*,...,x_n^*) = \max_{1\le j\le p} d_j(x_1^*,...,x_i^*,...,x_n^*).$$

Систему работы RFID можно представить как процесс, посредством которого собираются данные в определенный момент времени. Причем не редко встречаются ситуации, когда в определенный момент времени одновременно поступают данные от *n* меток (см.[2]), случайным образом перекрывающих друг друга.

В отличие от [23] в оперативно реализованном механизме вывода по прецеденте нет необходимости «принудительно» вводить метрику в алгоритм предпочтения прецедентов.

4.3.2. Иллюстративный пример.

Пусть некоторая ПрС/С описывается ситуационным вектором SV(ПрС/С-решение) с координатами{x_1, x_2, x_3, x_4}, представленными лингвистическими переменными:

x_1 – удельная энергия летательного аппарата;

x_2 - увеличение длины траектории при движении летательного аппарата только в горизонтальной плоскости;

x_3 - увеличение длины траектории при движении летательного аппарата только в вертикальной плоскости;

x_4 - степень достижения конечного результата.

Пусть для этого класса ПрС/С наблюдались два успешных прецедента d_1, d_2, каждый из которых применялся в двух (разных) случаях.

Лингвистические переменные принимают следующие унифицированные значения (термы):

x_i = {малая (мл.), средняя (ср.), высокая (выс.)}, i = 1, …, 4. (см. рис.3 и табл.4).

Пусть каждый терм лингвистических переменных представлен унифицированным универсальным множеством (десятибалльная шкала) и унифицированными кусочно-линейными (треугольными) функциями принадлежности (рис.2).

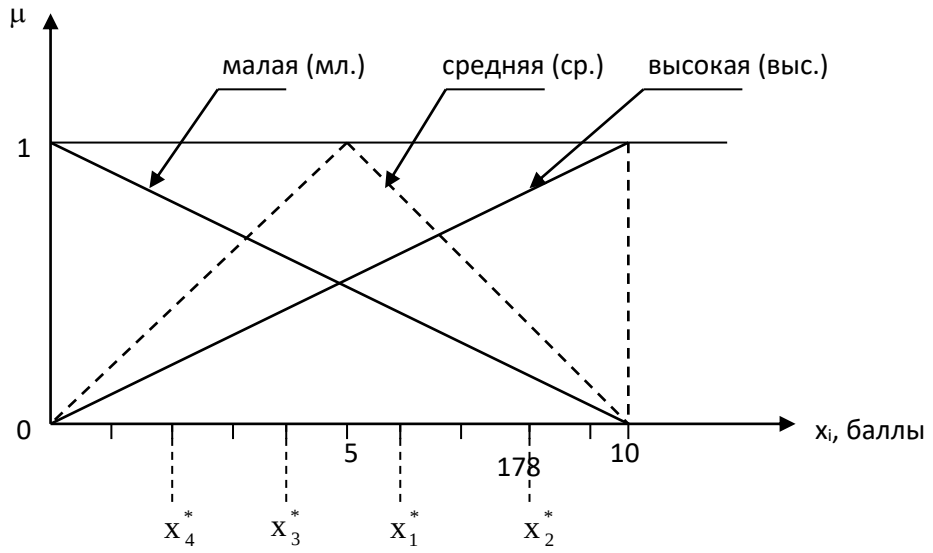


Рисунок 2. Графическое представление термина лингвистических переменных

Матрица знаний для этой ПрС/С представлена в табл.3.

Таблица 3. PSR / S матрица.

№ п/п	• Координаты ситуационного вектора				Прецеденты
	x_1	x_2	x_3	x_4	
1.1 1.2	выс. выс.	выс. ср.	выс. ср.	выс. выс.	d_1
2.1 2.2	выс. ср.	ср. выс.	выс. ср.	мл. ср.	d_2

Пусть наблюдается ПрС/С, описываемая координатами $x_1^* = 6,\quad x_2^* = 8,$ $x_3^* = 4,\quad x_4^* = 2$.

Тогда, согласно алгоритму выбора предпочтительного прецедента получим решение в табл.4.

Таблица 4.

№ п/п	Координаты ситуационного вектора				Min	max
	x_1	x_2	x_3	x_4		
1.1 1.2	0.6 0.6	0.8 0.4	0.4 0.6	0.2 0.2	0.2 0.2	0.2
2.1 2.1	0.6 0.8	0.4 0.8	0.4 0.6	0.8 0.4	0.4 0.4	0.4

Вывод: наиболее предпочтительным прецедентом для разрешения ПрС/С
$(x_1^* = 6,\ x_2^* = 8,\ x_3^* = 4,\ x_4^* = 2)$ является прецедент d_2.

4.4.Алгоритмическая оболочка оперативного вывода по прецеденту.

Выполнена программная реализация алгоритмической оболочки оперативного вывода по прецеденту. Оболочка реализована на языке С++ в среде Microsoft Visual Studio 2008 с использованием библиотеки QT под операционную систему Windows.

Полученный программный продукт позволяет смоделировать произвольную проблемную ситуацию (проблемную субситуацию), задавая ее ситуационным вектором SV(ПрС/С – решение) с полным описанием его координат (для каждой координаты вводится набор термов и относящаяся к

каждому терму трапецевидная функция принадлежности и универсальное множество, на котором она определена) и матрицей знаний.

Любая конкретная реализация проблемной ситуации представляется этим же ситуационным вектором SV(ПрС/С – решение) с текущими количественными координатами.

Работу реализованной алгоритмической оболочки рассмотрим на примерах.

4.4.1. Примеры решения задач с использованием программной реализаци алгоритмической оболочки оперативного вывода по прецеденту.

Пример 1. Пусть рассматриваемая нами проблемная ситуация описывается следующим ситуационным вектором:

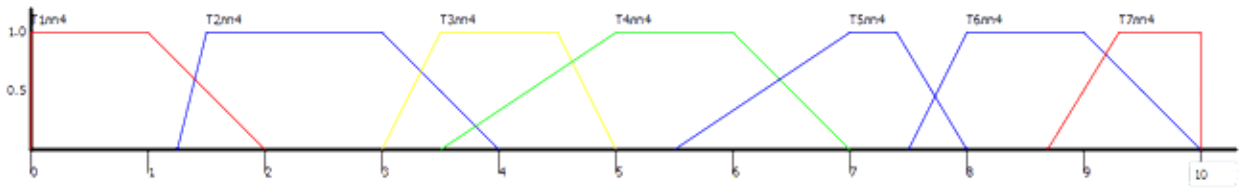
(ПрС/С – решение) = {ЛП1, ЛП2, ЛП3, ЛП4, ЛП5},

где ЛП1..ЛП5 – координаты ситуационного вектора (лингвистические переменные). Область значений лингвистических переменных определяются термами:

$$ЛП1 = \{Т1_{лп1}, Т2_{лп1}, Т3_{лп1}, Т4_{лп1}, Т5_{лп1}\}$$
$$ЛП2 = \{Т1_{лп2}, Т2_{лп2}\}$$
$$ЛП3 = \{Т1_{лп3}, Т2_{лп3}, Т3_{лп3}\}$$
$$ЛП4 = \{Т1_{лп4}, Т2_{лп4}, Т3_{лп4}, Т4_{лп4}, Т5_{лп4}, Т6_{лп4}, Т7_{лп4}\}$$
$$ЛП5 = \{Т1_{лп5}, Т2_{лп5}, Т3_{лп5}\}$$

Для каждой лингвистической переменной формируется ансамбль трапецевидных функции принадлежности, заданных на соответствующем универсальном множестве.

Для ЛП4 они будут выглядеть следующим образом (рис3.):



Для остальных лингвистических переменных – аналогично.

Формируется вектор возможных исходов

Пусть в матрицу знаний введены четыре прецедента (исхода):

И = {И1, И2, И3, И4}

Пусть с помощью экспертов сформирована матрица знаний (табл.5).

Таблица 5. Матрица знаний

Getting results

	LV1	LV2	LV3	LV4	LV5	Outcomes	Value
1	T3Lv1	T1Lv2	T1Lv3	T5Lv4	T1Lv5	O1	0.125
2	T2Lv1	T2Lv2	T3Lv3	T4Lv4	T2Lv5	O1	0.333333
3	T1Lv2	T4Lv3	T5Lv4	T2Lv5	T2Lv5	O2	0

4	T4Lv1	T1Lv2	T3Lv3	T4Lv4	T1Lv5	O2	0
5	T2Lv1	T2Lv2	T1Lv3	T5Lv4	T2Lv5	O2	0.125
6	T3Lv1	T1Lv2	T1Lv3	T6Lv4	T1Lv5	O3	0
7	T3Lv1	T2Lv2	T3Lv3	T5Lv4	T3Lv5	O3	0
8	T2Lv1	T1Lv2	T4Lv3	T1Lv4	T1Lv5	O3	0

Пусть на вход оперативного вывода по прецеденту подается текущий SV(ПрС/С – решение), значения координат которого указаны в табл.6.

Таблица 6. Координаты S матрицы

Current SV		
LV1	3	3
LV2	5	5
LV3	6.5	6.5
LV4	6	6
LV5	4	4

Для этого значения SV(ПрС/С – решение) матрица знаний преобразуется в количественную матрицу знаний (табл.7).

Таблица **7.** Матрица Количественные знания

Getting results							
	LV1	LV2	LV3	LV4	LV5	Outcomes	Value
1	0.6	1	0.125	0.333333	0.5	O1	0.125
2	1	0.5	0.333333	1	1	O1	0.333333
3	0.6	1	0	0.333333	1	O2	0
4	0	1	0.333333	1	0.5	O2	0
5	1	0.5	0.125	0.333333	1	O2	0.125
6	0.6	1	0.125	0	0.5	O3	0
7	0.6	0.5	0.333333	0.333333	0	O3	0
8	1	1	0	0	0.5	O3	0

В результате применения к матрице знаний алгоритмов логического вывода получаем рекомендации о предпочтительном прецеденте (исходе). Для данного количественного ситуационного вектора это прецедент И1.

В результате расчетов определяем предпочтительный прецедент И1 для разрешения конкретной реализации ПрС/С, определяемой SV(ПрС/С – решение), который показан в таблице 6.

Представлен оперативный вывод по прецеденту и программно реализованная алгоритмическая оболочка оперативного вывода. Программная реализация алгоритмической оболочки ориентирована как на автономное ее использование, так и на встраивание ее в базу знаний бортовых оперативно советующих экспертных систем [24].

Список литературы:

1. Филлипенко И.В. Математическая модель систем радиочастотной идентификации с кодовым разделением каналов // Восточно-Европейский журнал передовых технологий. 2011. 5/3 (53). С. 34-36.
2. Черняховская Л.Р., Федорова Н.И., Низамутдинова Р.И. Интеллектуальная поддержка принятия решений в оперативном управлении деловыми процессами предприятия // Вестник УГАТУ. Серия управление, вычислительная техника, информатика. Уфа: УГАТУ. 2011. т.15, № 2 (42). С. 172 – 176.
3. Прохоров М.Д., Федунов Б.Е. Вывод по прецеденту в базах знаний бортовых интеллектуальных систем, размещаемых на борту антропоцентрических объектов. // Искусственный интеллект и принятие решений. 2010/03. с. 62 -73.
4. Дшхунян В.Л. Электронная идентификация. Бесконтактные электронные идентификаторы и смарт-карты / В.Л. Дшхунян, В.Ф.Шаньгин. М.: ООО «Издательство АСТ»: Издательство «НТ Пресс», 2004. С. 695.
5. Варшавский П.Р., Еремеев А.П. Моделирование рассуждений на основе прецедентов в интеллектуальных системах поддержки принятия решений. // Искусственный интеллект и принятие решений. № 2. 2009. с.45-57.
6. Блэкуэлла Д., Гиршик М.А. Теория игр и статистических решений / пер. с англ. И.В.Соловьева. под ред. Б.А.Севастьянова. М.: изд. иностр. Лит. 1958. с. 374.
7. Демьянов В.Ф., Малоземов В.Н. Введение в минимакс. М.: Наука, 1972. с. 368.
8. Godonoaga A., Balan P. A probabilistic method for solving minimax problems with general constraints // Bul/ Acad/ Stiinte Repub/ Mold/ Mat., 2010, номер 1, с. 33-46.
9. Галев А.В., Косолапов А.С. Исследование влияния структурных помех на помехоустойчивость систем с широкополосными шумоподобными сигналами при когерентном приеме. // Электронное научно- техническое издание Наука и образование. 2012, № 4, апрель с. 1-15.
10. Вертегел В.В. Моделирование системы радиочастотной идентификации . //Вісник СевНТУ: зб. Наук. Пр. Вип. 131/2012. Серія Інформатика, електроника, зв`язок. Севостополь, 2012. с. 95-101.
11. Искакова А.С. Определение наиболее подходящей несмещенной оценки вероятности оправдываемости прогноза в метеорологии. // Сибирский журнал индустриальной математики. 2002 г.Том V, 1(9). С.79-84.
12. Риордан Д. Введение в комбинаторный анализ. М. 1963. – 287 с.
13. Савельев Л.Я. Комбинаторика и вероятность. М.: Наука. 1975.– 424с.
14. Федунов Б.Е.. Интеллектуальная поддержка экипажа на борту антропоцентрического объекта**.** **–** М., Мехатроника, автоматизация, управление. №2, 2010. стр.62-70

15. Системы управления вооружением истребителей: основы интеллекта многофункционального самолета. Под редакции академика РАН Е.А. Федосов. Российская Академия Ракетных и Артиллерийских Наук. М. Машиностроение. 399 стр.2005.
16. Федунов Б.Е. Проблемы разработки бортовых оперативно-советующих экспертных систем. // Изв. РАН. ТиСУ. 1996. № 5.
17. Грибков В.Ф., Федунов Б.Е. Бортовая информационная интеллектуальная система «Ситуационная осведомленность экипажа боевых самолетов». Сборник докладов VIII Всероссийской юбилейной конференции «Проблемы совершенствования робототехнических и интеллектуальных систем летательных аппаратов. Москва, МАИ-ПРИНТ,2010. стр.171-177.
18. Федунов Б.Е. Конструктивная семантика для разработки алгоритмов бортового интеллекта антропоцентрических объектов. // Изв. РАН. ТиСУ. 1998. № 5.
19. Федунов Б.Е. Механизмы вывода в базе знаний бортовых оперативно советующих экспертных систем. // Изв. РАН. ТиСУ. 2002. №4
20. Заде Л. Понятие лингвистической переменной и её применение к принятию приближённых решений. М.: Радио и связь, 1976.
21. Кофман А. Введение в теорию нечетких множеств. М.: Радио и связь, 1982.
22. Ротштейн А.П. Интеллектуальные технологии идентификации. Винница: Универсум, 1999.
23. Варшавский П.Р., Еремеев А.П. Моделирование рассуждений на основе прецедентов в интеллектуальных системах поддержки принятия решений. Журнал ЭИскусственный интеллект и принятие решений». №2. 2009. стр. 45-57.
24. Федунов Б.Е., Шестопалов Е.В.. Оболочка бортовой оперативно советующей экспертной системы для типовой ситуации полета «Ввод группы в воздушный бой». – М., Изв. РАН, ТиСУ, 2010, №3, стр.86 - 103

М.М.ИЛИПОВ ., Г.Б.ИСАЕВА

РАЗРАБОТКА ЭФФЕКТИВНЫХ АЛГОРИТМОВ ЗАЩИТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ МИКРОПРОЦЕССОРНЫХ КАРТ

монография

Басылуға 15.09.2020 ж. қол қойылды
Таралымы 500 дана. Форматы 60x84 $^{1}/_{16}$.
Офсеттік қағаз. Көлемі 11,5 баспа табақ. Тапсырыс № 43