Технологические особенности разработки доверенных встраиваемых ОС для защищённых микроконтроллеров

OS DAY 2022

Олег Дьяков

директор по **инновационным** проектам

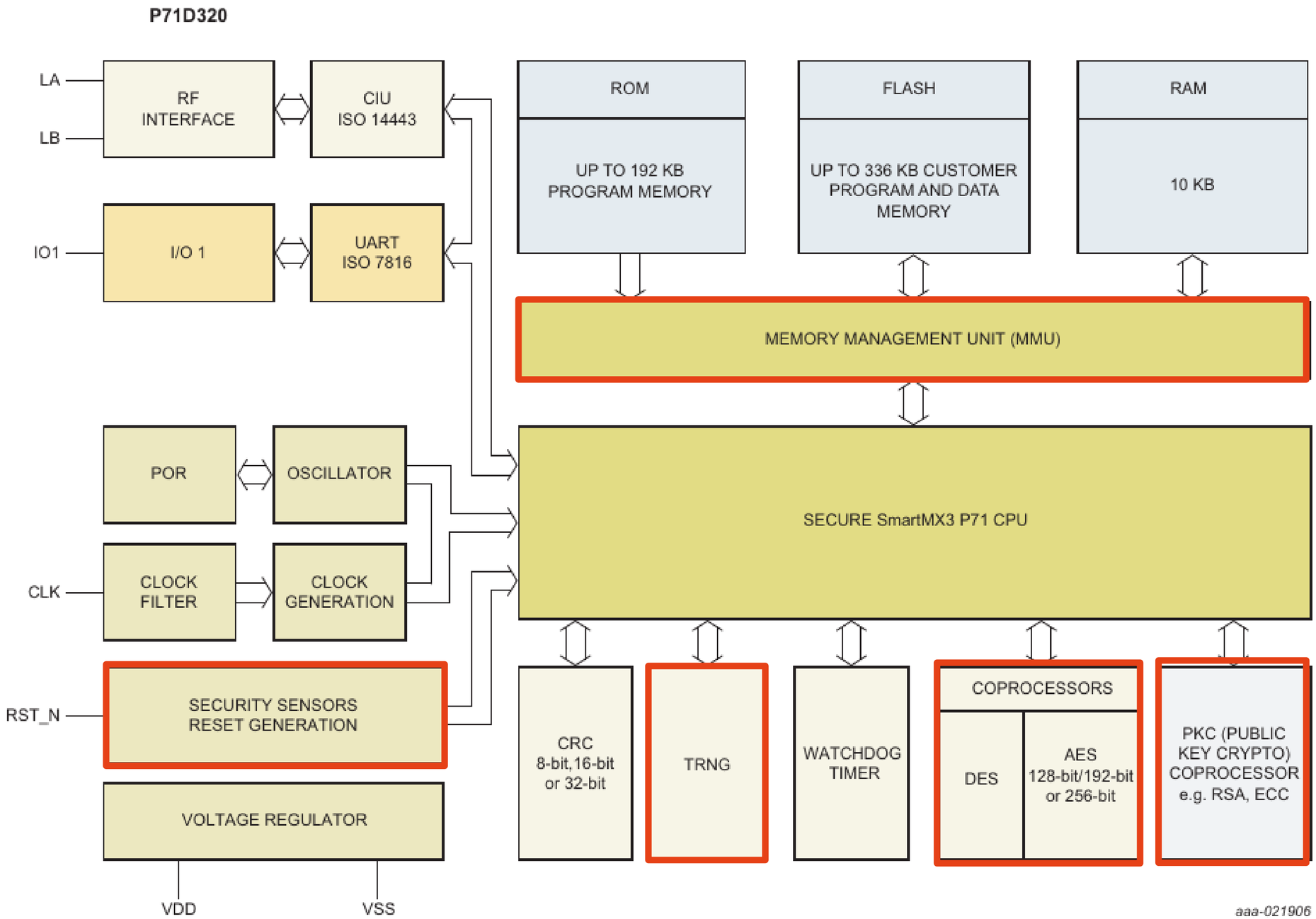
АО "Аладдин Р.Д."

Архитектура защищённых контроллеров

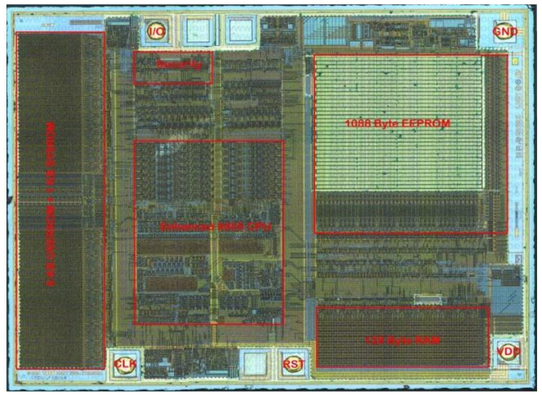
Защищённые микроконтроллеры

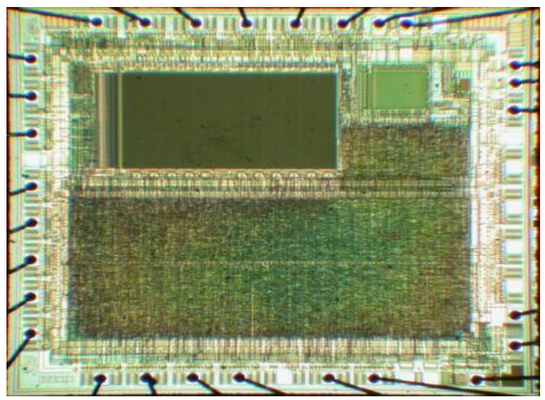
- Защищённые микроконтроллеры (ЗМК) достаточно широко используются для обеспечения безопасного хранения секретных данных и выполнения криптографических операций
- Применяются при изготовлении банковских и идентификационных смарт-карт, SIM карт, USB токенов, встроенных элементов безопасности для IoT (eSE), Trusted Platform Module (TPM) и т.д.
- При разработке программного обеспечения для защищённых микроконтроллеров нужно учитывать повышенные требования кибербезопасности, особенности архитектуры и жизненного цикла

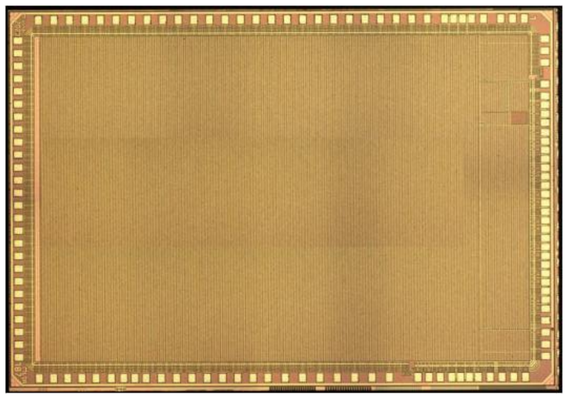
Архитектура защищённого микроконтроллера



Инженерные методы защиты

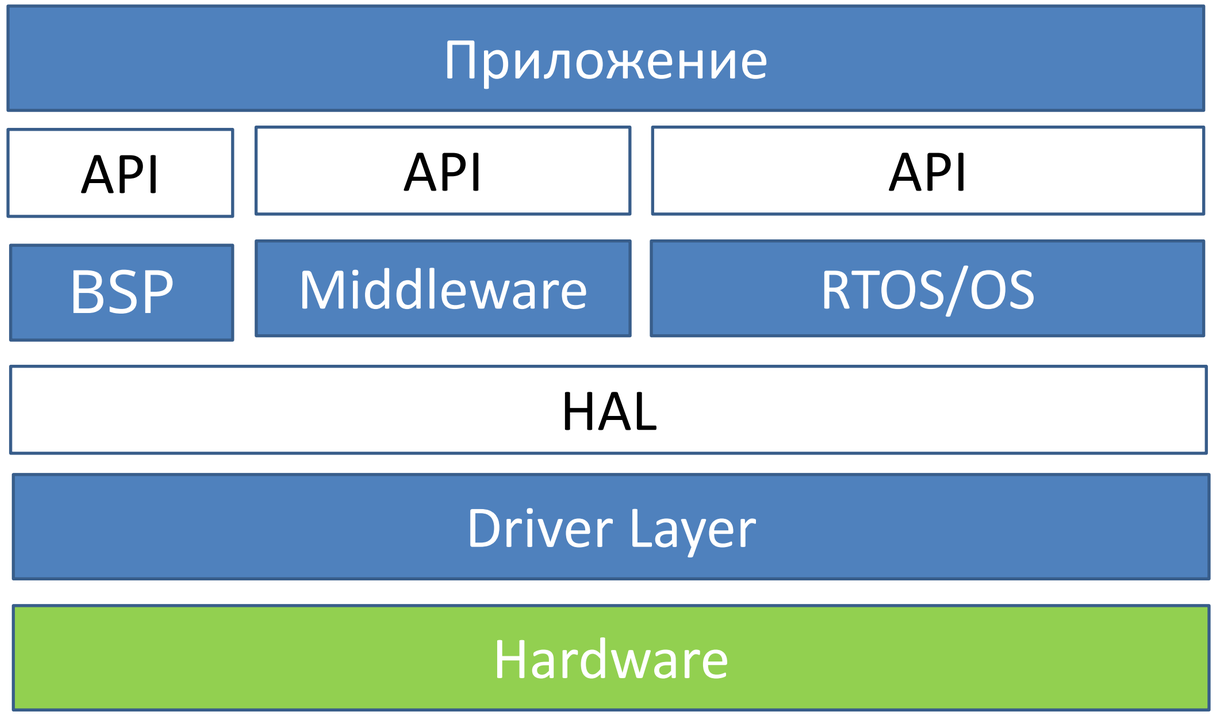






- Защитная экранная сетка
- Запутывание топологии (Glue Logic)
- Блок генерации шума в цепи питания
- Блок аппаратного скремблирования шины данных и шины адреса
- Блок оптического детектора
- Блок детектора изменения тактовой частоты
- Блок детектора изменения напряжения питания
- Отсутствие средств аппаратной отладки (при разработке используются эмуляторы аппаратной платформы)

Структура программного обеспечения защищённого микроконтроллера

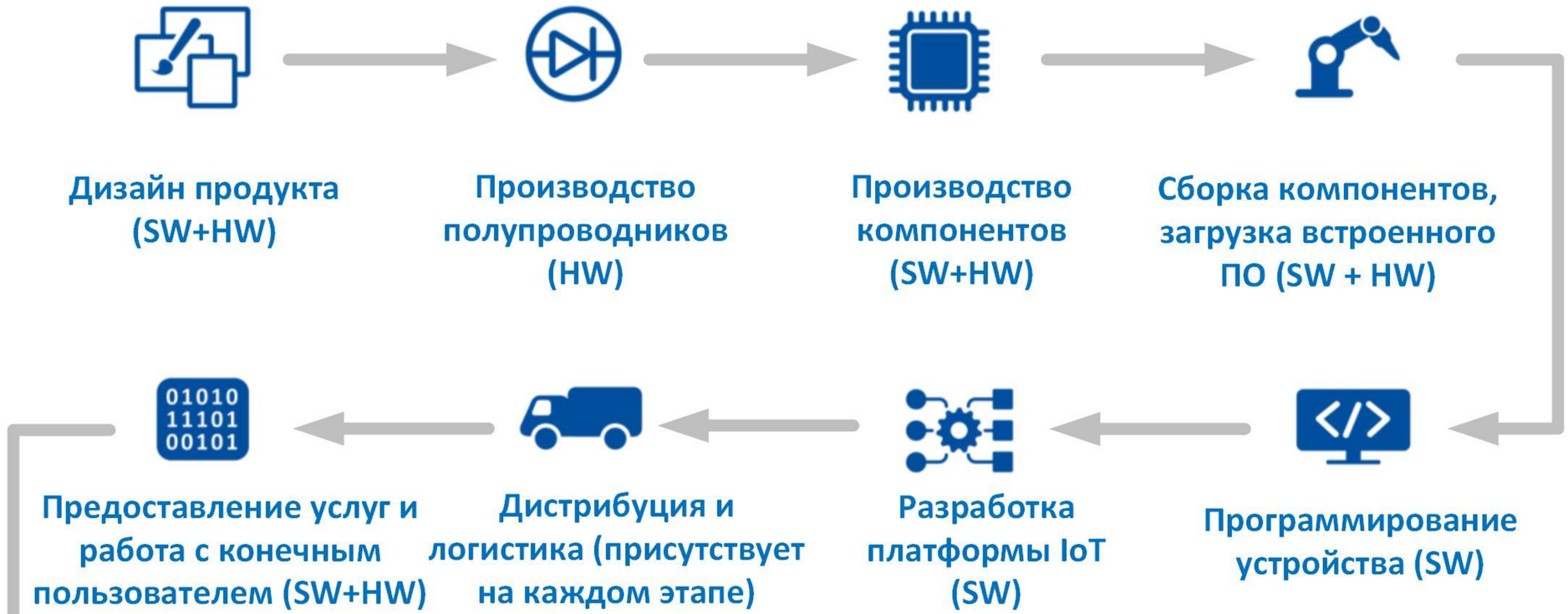


- API (Application program interface)
- BSP (board support package)
- HAL (Hardware Abstraction Layer)
- RTOS (Runtime Operation System)
  - это просто еще один компонент в середине стека программного обеспечения

- Достичь высокого уровня кибербезопасности невозможно только за счёт программных или аппаратных методов
- При разработке встроенного программного обеспечения для защищённых микроконтроллеров с целью достижения оптимального результата необходимо скоординировать работу аппаратного и программного обеспечения

Разработка доверенного встраиваемого программного обеспечения на основе принципов **Secure-By-Design**

Безопасная цепочка поставок IoT/M2M устройств (ENISA)



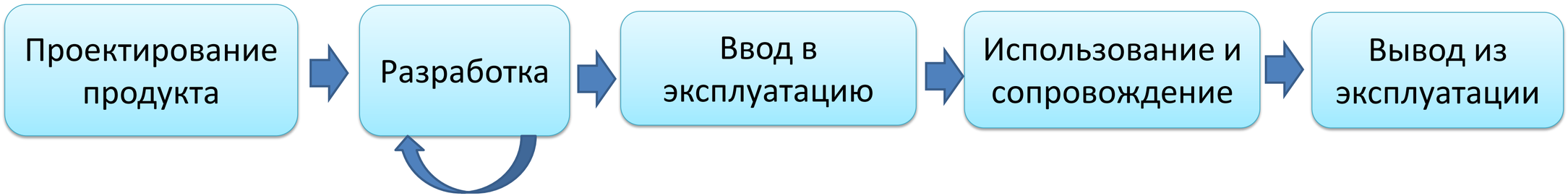
- Безопасность следует рассматривать как один из базовых аспектов жизненного цикла системы, а не как очередную её функцию

Secure-by-Design

- Secure-by-Design — это подход к разработке программного и аппаратного обеспечения, направленный на минимизацию уязвимостей систем и уменьшение поверхности для атак путём проектирования и создания безопасности на каждом этапе жизненного цикла информационной системы
- Подразумевает включение спецификаций безопасности в дизайн, непрерывную оценку безопасности на каждом этапе и соблюдение передовых практик
- Существуют некоторые готовые методологии разработки Secure-By-Design (например Microsoft Security Development Lifecycle, Cisco SDL)
- Аналогом SDL в России является ГОСТ Р 56939-2016

Secure-by-Design методология разработки

Agile development lifecycle

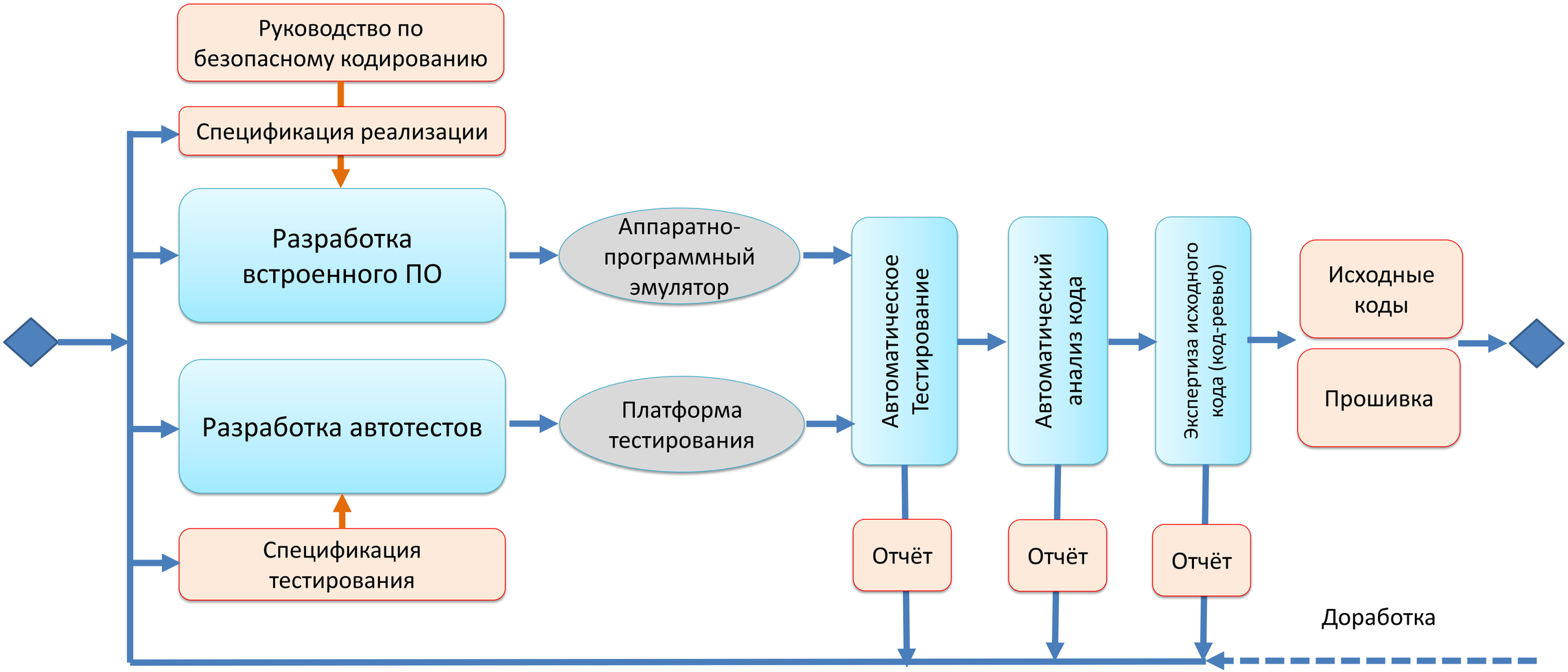


Secure-by-design development lifecycle

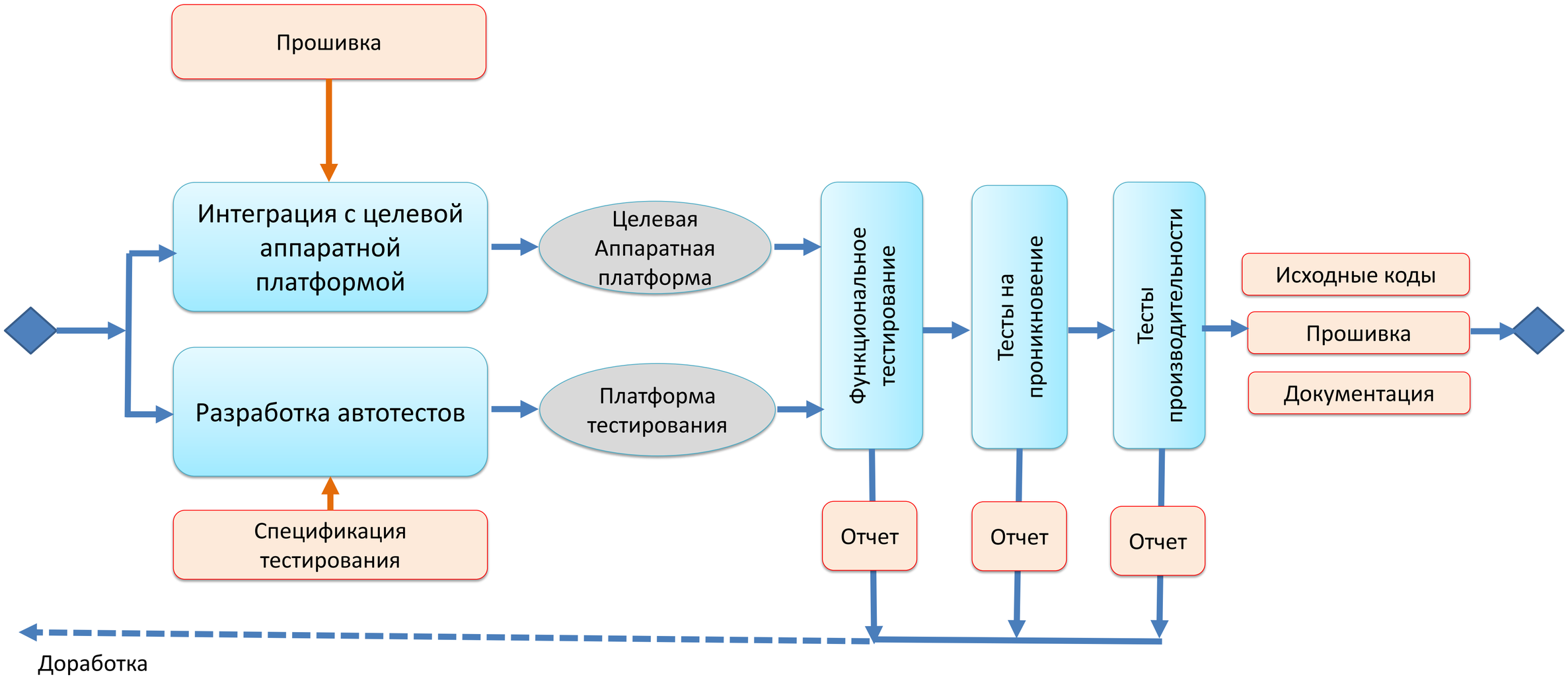
Требования безопасности	Безопасное кодирование	Сертификация	Аудит	Безопасная утилизация
Моделирование угроз	Тесты на проникновение	Безопасный провижининг	Непрерывный мониторинг	
Оценка рисков	Анализ исходного кода		Безопасное обновление	
Руководство по безопасности	Экспертиза исходного кода (код-ревью)			

Проектирование программного обеспечения ЗМК

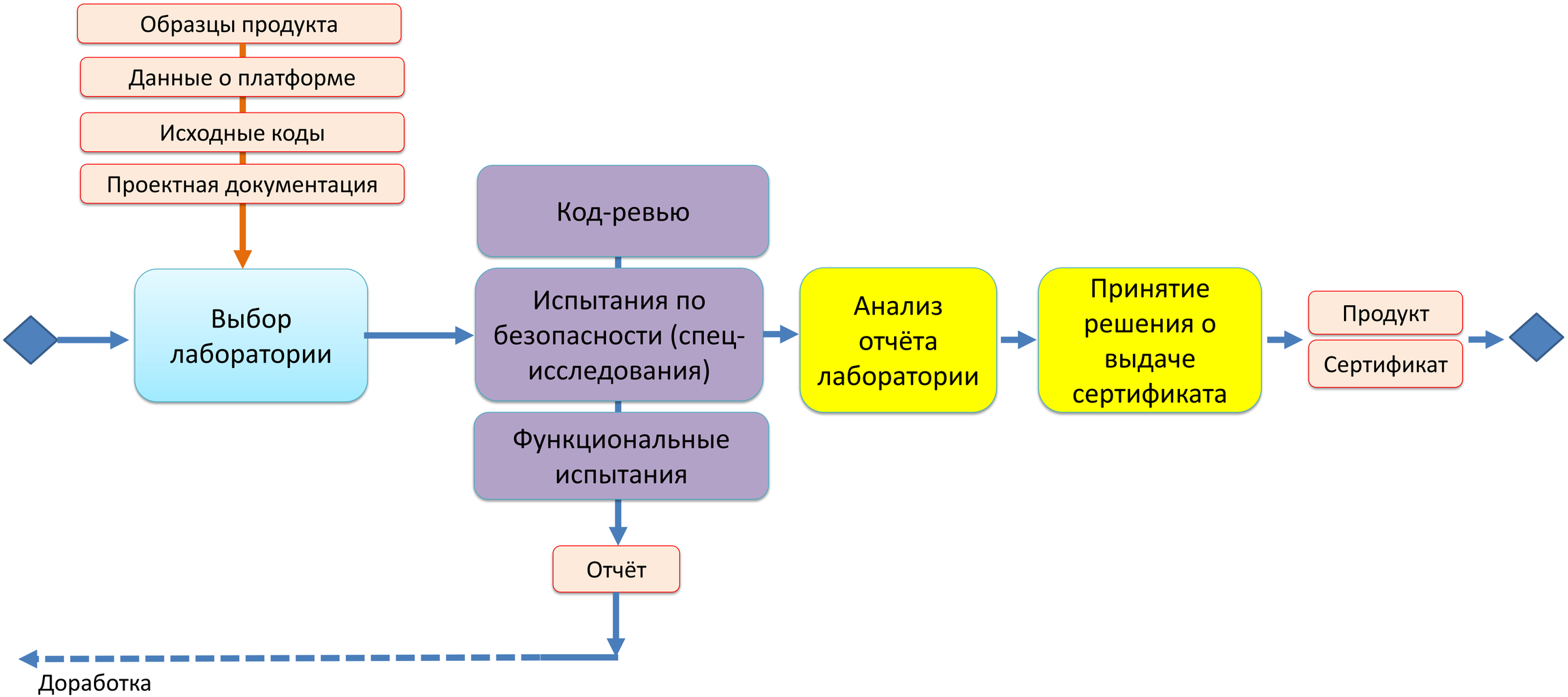
Разработка программного обеспечения ЗМК (основной цикл разработки)



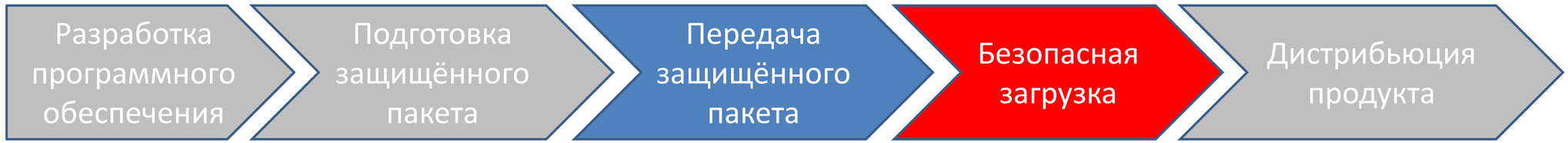
Разработка программного обеспечения ЗМК (интеграция с платформой)



Сертификация продукта



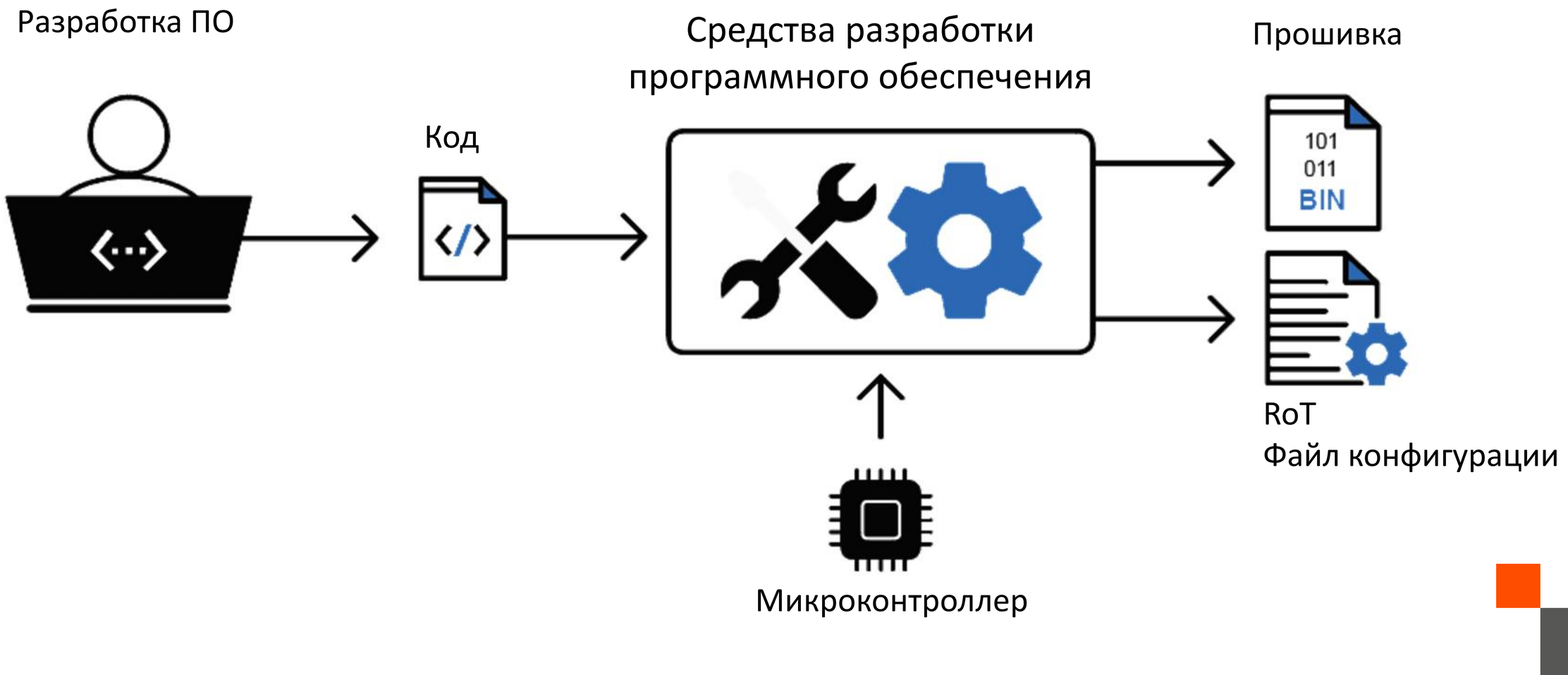
Безопасный провижининг



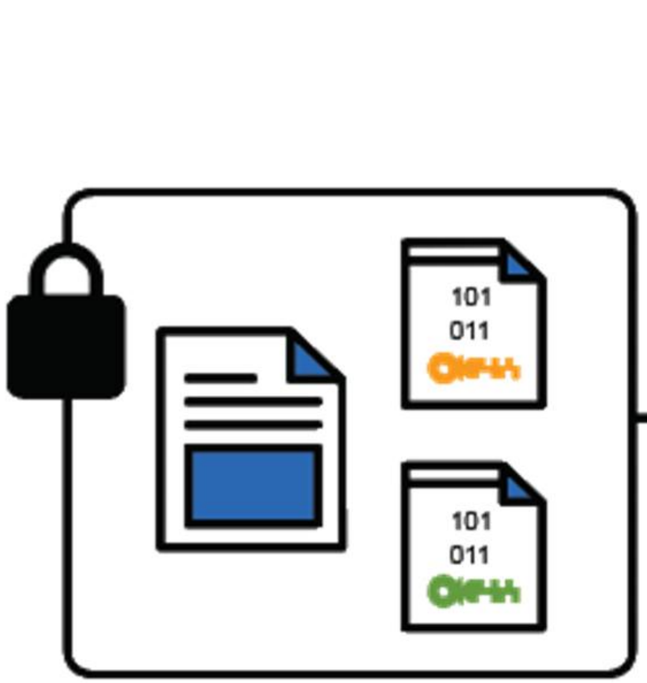
- **Безопасный провижининг (Secure Provisioning)** – технология безопасной загрузки прошивки в защищённый микроконтроллер при серийном производстве продукции на разных производственных площадках

Безопасный провижининг

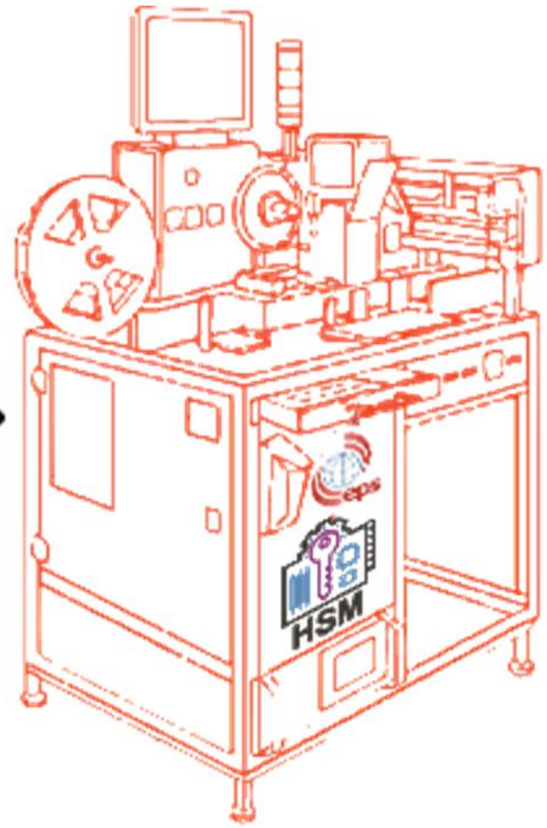
- Secure Provisioning тесно связан с процессом разработки программного обеспечения



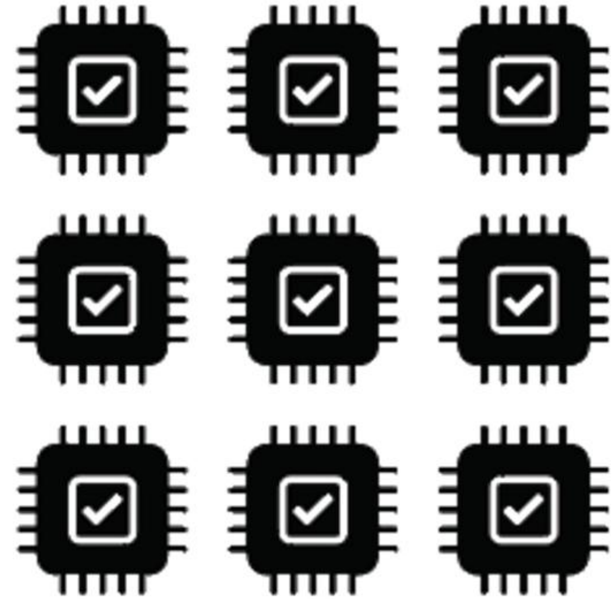
Безопасный провижининг



Зашифрованный пакет



Установка безопасной загрузки

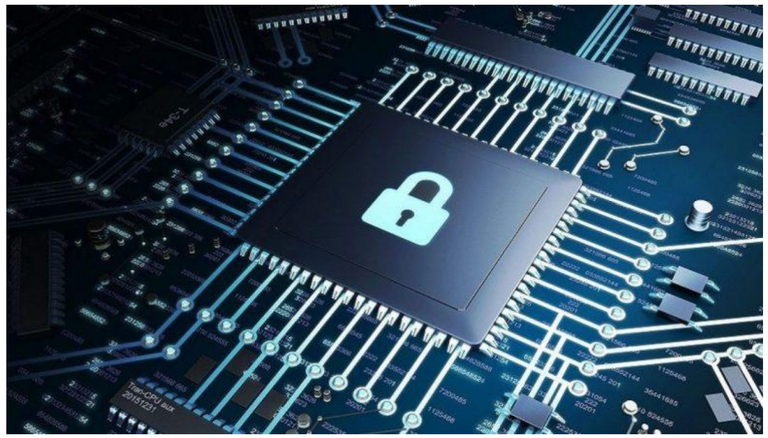


Микроконтроллеры с безопасной прошивкой

Безопасное кодирование

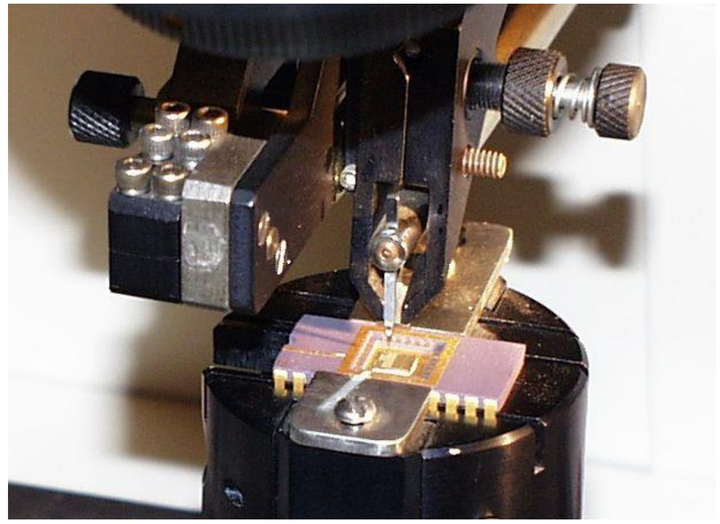
Безопасное кодирование

- Безопасное кодирование это практика разработки программного обеспечения таким образом, чтобы предотвратить случайное появление уязвимостей в системе безопасности
- При программировании защищённых микроконтроллеров к техникам безопасного кодирования также относят программные методы поддержки целостности кода и данных, а также защиты от атак по побочным каналам (технические каналы утечки информации)

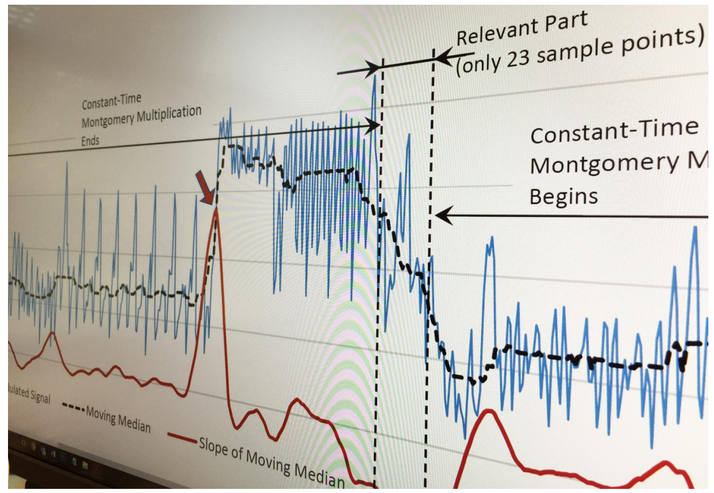


Атаки по побочным каналам

- Атаки по побочным каналам (SCA) направлены на извлечение секретов из чипа или системы посредством измерения и анализа физических параметров
- При тестировании защищённых микроконтроллеров SCA рассматривают как особый вид тестов на проникновение



- Атака зондированием (probing attack)
- Атака по времени (timing attack)
- Атака по ошибкам вычислений (fault–induction attack)
- Атака по энергопотреблению (power analysis attack)
- Атака по электромагнитному излучению (electromagnetic analysis attacks)



Аппаратные средства защиты

- Основные аппаратные меры противодействия:
  - Экранирование (пассивное и активное)
  - Зашумление питания
  - Датчики проникновения
  - Защита памяти и интерфейсов
- Дополнительные меры противодействия:
  - Сторожевой таймер
  - Проверка целостности
  - Маскировка/шифрование памяти
  - Рандомизация тактовой частоты

Безопасное кодирование

Рекомендации по функциональной безопасности (MISRA, и .т.д.)

Спецификация на платформу

Рекомендации по безопасности от поставщика платформы

Контрмеры против атак по побочным каналам

Рекомендации регуляторов

Руководство по безопасному кодированию

- Руководство по безопасному кодированию подготавливается для каждого проекта в зависимости от аппаратной платформы, модели угроз нарушителя, известных на текущий момент атак

Безопасное кодирование

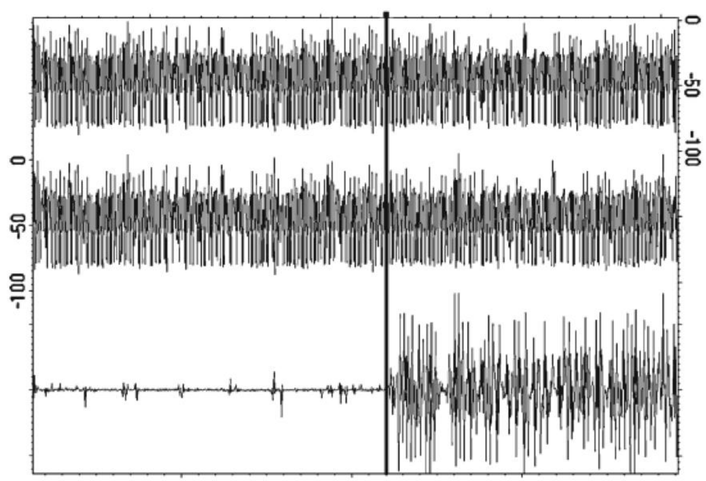
- Использование программных методов защиты от атак по побочным каналам позволяют существенно повысить безопасность устройства
- Программные методы защиты увеличивают размер кода, делают исходные коды менее читаемыми, уменьшают скорость выполнения операций
- Необходимо соблюдать баланс между аппаратными методами защиты, программными методами защиты и реальной опасностью атаки
- Рекомендуемые программные методы защиты от атак и процедуры безопасного кодирования указываются в документе проекта «Руководство безопасного кодирования»
- Анализ исходного кода (автоматический и ручной) проводится в соответствии с требованиями руководства безопасного кодирования

Основные рекомендации

- Активация и правильное использование аппаратных средств защиты
- Анализ целостности исполняемого кода при старте микроконтроллера
- Анализ целостности данных (переменных, структур) при каждом использовании
- Применение акселераторов и доверенных библиотек при выполнении криптографических операций
- Активация MPU, разграничение доступа к ресурсам с использованием привилегий исполняемого кода
- Контроль работы критических функций с использованием технологии контроля времени выполнения (Watchdog Timer)

Hiding — усложнение получения информации из побочных каналов

- **Уравнивание времени выполнения операций** – все этапы шифрования в устройстве должны выполняться за одинаковое время
- **Балансировка энергопотребления** – по возможности при проведении операций должны быть задействованы все аппаратные части устройства, на неиспользуемых частях следует проводить ложные вычисления
- **Устранение условных переходов** – устранение в алгоритме операций условного перехода, зависящего от входных данных или секретного ключа



Masking — усложнение интерпретации получаемой информации

- **Фиктивные операции (dummy)** – выполнение дополнительных вычислений со случайными данными и ключами
- **Рандомизация (randomization)** – случайная перестановка блоков данных
- **Ослепление (blinding)** – изменение входных данных алгоритма в какое-то непредсказуемое состояние с использованием функции маскирования

Обнаружение манипуляций

- **Двойная проверка** – чувствительные данные дважды передаются и проверяются
- **Сигнатуры функций** – использование специальных констант, отправляемых в функцию и возвращающихся из функций; неожиданное значение сигнатуры воспринимают как атаку
- **Обратные расчёты** – выполнение обратного преобразования и сравнение результата с входными данными

Спасибо!

Олег Дьяков
директор по инновационным проектам
АО "Аладдин Р.Д."
8 (915) 130 3943
www.aladdin.ru