ГЛОССАРИЙ

Корпоративный Глоссарий ADVENT SYSTEMS | Технологии ACS, Кибербезопасности, Идентификации | +**RUST LANG DEVELOPMENT**

www.advent-systems.com | www.advent-id.com | www.advent-lab.tech | +74992130058 | info@sprx.ru

Глоссарий:

● **Active RFID transponder (RFID Активный транспондер (Активная метка))**
Транспондер, содержащий батарею или другой источник электроэнергии, обеспечивающий энергию, необходимую для передачи информации и получения данных.

● **AC (Переменный ток)**
Это тип электрического тока, который периодически непрерывно меняет своё направление. Переменный ток обычно используется в источниках питания. Другой тип тока — постоянный ток (DC).

● **Access Event (Событие доступа)**
Событие доступа — это выполнение определённого действия в системе контроля доступа. В системах могут быть определены десятки различных типов событий. Примеры событий доступа: «Доступ разрешён», «Доступ запрещён», «Дверь открыта слишком долго» и «Дверь открыта принудительно».

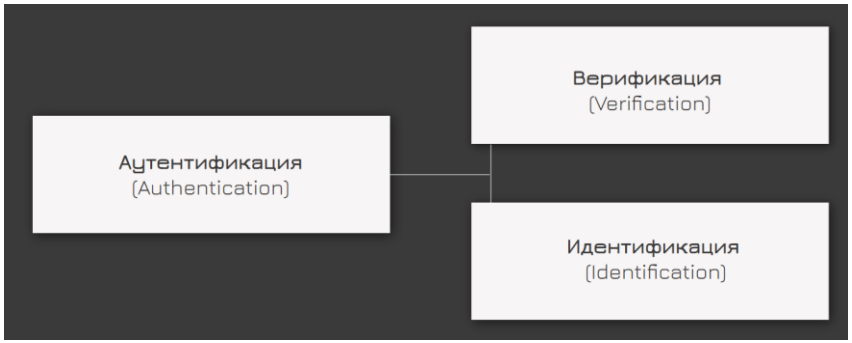
● **Active Infrared (Активные ИК датчики)**
Активные инфракрасные датчики излучают инфракрасное излучение в зону своей активации. Когда эти лучи отражаются от поверхностей и возвращаются обратно к датчику, датчик измеряет их. Когда человек или препятствие попадает в обычную зону активации, параметры отраженных инфракрасных лучей изменяются, что приводит к срабатыванию датчика, например, открыванию или закрыванию двери.

● **Agile Reader (ADVENT использует термин – Мультиформатный считыватель)**
Общий термин, который обычно относится к считывателю RFID, который может считывать метки, работая на разных частотах или используя разные методы связи между метками и считывателями.

● **Air Interface Protocol (Протокол RFID считывания)**
Правила (Стандарты), регулирующие взаимодействие тегов (меток) и считывателей.

● **Authentication (Аутентификация)**
Аутентификация — это процесс определения того, является ли кто-то (или что-то) тем, за кого себя выдаёт. В контексте вычислений учётные данные, предоставленные лицом, запрашивающим доступ к ресурсу, сравниваются с данными поставщика этого ресурса. Если учётные данные совпадают, пользователю предоставляется доступ к ресурсу. Аутентификация пользователя осуществляется на основе того, что он знает (например, пароль), того, что у него есть (например, токен безопасности) или того, кем он является (например, отпечаток пальца). Эти методы могут комбинироваться для обеспечения более высокого уровня безопасности, известного как двухфакторная или многофакторная аутентификация.

По сути Аутентификация (Authentication) - это базовый термин, означающий **Верификацию (Verification)** или **Идентификацию (Identification)** (важно понимать разницу терминов, так как от этого зависит принятие правильного решения при конфигурации и построении системы!).



Он Иван?

Кто Он такой?

● **Amplifier (Усилитель дальности)**
Усилитель. Дальность считывания этикетки можно расширить с помощью конденсаторов.

● **Amplitude (Амплитуда)**
Максимальное абсолютное значение периодической кривой, измеренное вдоль ее вертикальной оси (высота волны, говоря простым языком).

● **Amplitude modulation (Амплитудная модуляция)**
Изменение амплитуды радиоволны. Более высокая волна интерпретируется как 1, а нормальная — как ноль. Изменяя волну, RFID-метка может передавать считывателю строку двоичных цифр. Компьютеры могут интерпретировать эти цифры как цифровую информацию. Метод изменения амплитуды известен как амплитудная манипуляция (АМК). **Amplitude Change keying:** Изменение амплитуды волны для передачи данных, хранящихся на метке.

● **Antenna RFID (RFID Антенна)**
Проводящий элемент, позволяющий транспондеру отправлять и принимать данные. Пассивные низкочастотные (125 кГц) и высокочастотные (13,56 МГц) транспондеры обычно имеют спиральную антенну, которая, взаимодействуя со спиральной антенной считывателя, образует магнитное поле. Антенны транспондеров УВЧ-диапазона могут иметь различную форму. Считыватели также оснащены антеннами, излучающими радиоволны. Электромагнитная энергия от антенны считывателя «собирается» транспондером и используется для питания микрочипа, который затем передаёт обратно свои сигналы.
Антенна метки — это проводящий элемент, позволяющий метке отправлять и принимать данные. Пассивные низкочастотные (135 кГц) и высокочастотные (13,56 МГц) метки обычно оснащены спиральной антенной, которая, взаимодействуя со спиральной антенной считывателя, образует магнитное поле. Антенны меток УВЧ-диапазона могут иметь различную форму. Считыватели также оснащены антеннами, излучающими радиоволны. Радиочастотная энергия от антенны считывателя «собирается» антенной и используется для питания микрочипа, который затем изменяет электрическую нагрузку на антенну, отражая собственные сигналы.

● **Antenna Gain (Коэффициент усиления)**
С технической точки зрения, коэффициент усиления — это отношение мощности, необходимой на входе эталонной антенны без потерь, к мощности, подаваемой на вход данной антенны для создания в заданном направлении той же напряжённости поля на том же расстоянии. Коэффициент усиления антенны обычно выражается в децибелах, и чем выше коэффициент усиления, тем больше мощность выходной энергии. Антенны с более высоким коэффициентом усиления смогут считывать метки с большего расстояния.

● **Anti-Collision (Антиколлизия)**
Антиколлизия - Общий термин, используемый для обозначения методов предотвращения помех, создаваемых радиоволнами одного устройства другим. Алгоритмы предотвращения столкновений также используются для считывания нескольких меток в одном поле считывателя.

● **AoA / AoD**
Угол прихода/угол ухода. Это шлюзы, содержащие антенную решетку; последняя может быть расположена в линейной геометрии (ULA — Uniform Linear Array) или в плоской геометрии (URA — Uniform Rectangle Array) и позволяет оценивать либо азимутальный угол, либо азимутальный угол и угол места относительно самого шлюза. AoA принимает и обрабатывает сигнал метки, в то время как AoD отправляет сигнал на метку.

● **API - Application Programming Interface (Программный интерфейс-приложение)**
API - Программный интерфейс приложения (интерфейс прикладного программирования): инструкции или инструменты форматирования, используемые разработчиком приложения для связывания и создания аппаратных или программных приложений.

API - по сути - набор классов, процедур, функций, структур или констант, которыми одна компьютерная программа может взаимодействовать с другой программой. Обычно входит в описание какого-либо интернет-протокола (например, RFC), программного каркаса (фреймворка) или стандарта вызовов функций операционной системы. Часто реализуется отдельной программной библиотекой или сервисом операционной системы. Используется программистами при написании всевозможных приложений.

● **Application Family Identifier (Архитектура идентификатора на основе Приложений)**
Метод Международной организации по стандартизации (ИСО) для классификации радиочастотной идентификации по приложениям, позволяющий использовать единый протокол радиоинтерфейса для нескольких приложений.

● **ASCII**
Стандарт кодировки для сопоставления символов уникальным числовым кодам. Используется в большинстве современных компьютеров для передачи данных, вводимых с клавиатуры. ASCII — базовая кодировка, содержащая всего 256 символов. Для отображения расширенных наборов символов используются дополнительные стандарты, такие как UTF-8 или Unicode.

● **Asymmetric algorithm (криптография)**
Криптографический алгоритм использует два разных, но математически связанных ключа — один открытый и один закрытый. Открытый ключ, которым можно поделиться с кем угодно, используется для шифрования данных. Закрытый ключ, который необходимо хранить в секрете, необходим для расшифровки данных. Безопасность алгоритма заключается в сложности разложения больших целых чисел, являющихся произведением двух больших простых чисел. Хотя их легко умножить, время, необходимое для определения исходных простых чисел из суммы, непомерно велико даже для современных компьютеров.

● **Asymmetric encryption (криптография)**
Асимметричное шифрование — это метод шифрования данных, использующий два ключа: открытый и закрытый. Открытый ключ используется для шифрования данных и может быть широко распространен и открыт. Закрытый ключ используется для расшифровки данных, зашифрованных открытым ключом. Открытый и закрытый ключи — это очень большие числа, связанные определенной функцией, что делает вычисление одного из них крайне сложным, даже если известен другой.Цель асимметричного шифрования — защита данных при передаче; его принципы также нашли применение в цифровых подписях.

● **Attack Signature (Сигнатура атак)**
Файл, содержащий последовательность данных, используемую для идентификации атаки на сеть, обычно использующей уязвимость операционной системы или приложения. Такие сигнатуры используются системой обнаружения вторжений (IDS) или межсетевым экраном для обнаружения вредоносной активности,

направленной на систему.

● **Attack Surface (Объекты атаки) (Зона атаки)**
Количество потенциально уязвимых объектов в компьютерной системе. Этот термин применяется при оценке ресурсов, необходимых для защиты конкретной сети или устройства. Важной задачей информационной безопасности является сокращение количества уязвимых точек при сохранении функциональности системы. Поверхность атаки может быть уменьшена, например, путём отключения открытых, неиспользуемых портов.

● **Attack Vector (Векторная атака)**
Векторная атака — это путь, метод или средство, с помощью которых киберпреступники проникают в целевую систему. Векторы атаки могут включать инструменты и действия киберпреступников, а также человеческий фактор или уязвимые технологии потенциальной жертвы и её подрядчиков. Совокупность всех возможных векторов атаки в системе или организации называется поверхностью атаки.

● **Attenuation (Урезание сигнала)**
Уменьшение энергии. См. затухание сигнала.

● **Attenuator (Принцип резистора – устройство для усечения сигнала RFID)**
Устройство, подключаемое к линии передачи (коаксиальному кабелю), которое снижает мощность радиочастотного сигнала при его прохождении по кабелю от считывателя к антенне. Аттенюаторы обычно работают, рассеивая радиочастотную энергию в виде тепла.

● **Authentication (Аутентификация)**
Проверка личности человека, объекта или процесса. В RFID этот термин используется в двух значениях. Для бесконтактных смарт-карт и других платёжных систем считыватель должен убедиться, что транспондер является допустимым устройством в системе. То есть, кто-то не использует неавторизованное устройство для совершения мошенничества. Также обсуждается использование технологии EPC для аутентификации продуктов как способа снижения риска подделок.

● **Authorization (Авторизация)**
Авторизация — это процесс предоставления пользователю или группе пользователей определённых разрешений, прав доступа и привилегий в компьютерной системе. Разница между авторизацией, аутентификацией и идентификацией: Авторизацию не следует путать с идентификацией и аутентификацией пользователя. Обычно она происходит после завершения этих процессов. Предположим, пользователь хочет получить доступ к определённому документу в корпоративном облаке. Сначала он вводит логин своей учётной записи, и система проверяет, есть ли этот логин в её базе данных. Это и есть идентификация. Если логин существует, система запросит у пользователя пароль, вычислит его хеш и проверит, соответствует ли он хешу в базе данных. Это и есть аутентификация. Если логин и пароль верны, система проверит, имеет ли пользователь право читать и изменять запрошенный документ, и если да, предоставит ему доступ к файлу. Это и есть авторизация.

● **Air Gap (Физическое разделение блоков оборудования)**
Метод изоляции раздела компьютерной сети, в котором установка входящих/исходящих соединений запрещена на физическом, аппаратном или программном уровне. Простейший способ создания Air Gap - обеспечить полное отсутствие связи между сегментом и внешними хостами. Однако изоляция может быть достигнута и с помощью программно-аппаратных платформ, шифрующих и туннелирующих исходящий трафик. Воздушный зазор — один из вариантов обеспечения безопасности критически важных компьютерных сетей и отдельных устройств.

● **Amplification Attack (Кибератака «с усилением»)**
Тип кибератаки, включающий усиление исходного действия для вызова отказа в обслуживании в целевой системе. В отличие от стандартных DDoS-кампаний, усиление подразумевает асимметричный ответ со стороны зараженной машины: помимо маскировки IP-адреса злоумышленника, оно также отправляет жертве

пакет данных большего размера, чем первоначально полученный. Усиление может использовать различные типы интернет-пакетов, включая DNS, UDP и ICMP.

● **APT (Advanced Persistent Threats) (Высокотехнологичные скоординированные непрекращающиеся кибератаки)**
Этот термин применяется к скоординированным, скрытым, постоянным атакам на конкретные организации — в отличие от спекулятивных, изолированных, случайных инцидентов, составляющих основную часть киберпреступной активности. Как правило, APT-атаки осуществляются государственными органами. Такие атаки используют высокотехнологичное вредоносное ПО для нарушения защиты организации. APT-атаки — это частный случай целенаправленных атак.

● **Backdoor (Тип троянов с возможностью удаленного управления устройством / уязвимость с доступом к коду)**
Один из самых опасных типов троянов. Бэкдоры предоставляют автору или оператору трояна возможность удалённого управления компьютером жертвы. В отличие от легитимных утилит удалённого управления, они устанавливаются, запускаются и работают незаметно, без согласия или ведома пользователя. После установки бэкдорам можно приказать отправлять, получать, выполнять и удалять файлы, собирать конфиденциальные данные с компьютера, вести журнал активности на нём и многое другое.

● **Backporting (Бэкпортирование)**
Перенос фрагмента кода, модуля или другой части программы на более старую версию программного обеспечения. В области информационной безопасности бэкпортирование означает попытку установки исправления, закрывающего уязвимость в текущей сборке приложения, поверх более ранней версии. Это требуется, когда поставщик выпускает обновление безопасности только для текущей версии приложения (например, когда устаревшая версия больше не поддерживается).

● **Backscatter (Обратное рассеяние)**
Метод связи между пассивными метками (метками, не использующими батареи для передачи сигнала) и считывателями. RFID-метки, использующие технологию обратного рассеяния, отражают радиоволны от считывателя обратно к считывателю, обычно на той же несущей частоте. Отражённый сигнал модулируется для передачи данных.

● **Backward compatible (Обратная совместимость)**
Совместимо с системами предыдущего поколения.

● **Beacon (Метка-маяк)**
Активная или полуактивная RFID-метка, запрограммированная на активацию и передачу сигнала с заданными интервалами.

● **Behavioral analysis (Поведенческий анализ)**
Это относится к методу определения вредоносности приложения на основе его действий. Если приложение совершает что-либо, выходящее за рамки «допустимых», его работа ограничивается.

● **Behavioral Biometric Characteristic (Поведенческая биометрическая характеристика)**
Поведенческая биометрическая характеристика: биометрическая характеристика, которая определяется и формулируется с течением времени, а не основывается главным образом на биологии. Все биометрические характеристики в некоторой степени зависят как от поведенческих, так и от биологических характеристик. Примеры биометрических модальностей, для которых могут доминировать поведенческие характеристики, включают положение пальца или руки при биометрическом сканировании, распознавание подписи и динамику нажатия клавиш.

● **Benchmarking (Бенчмаркинг)**
Бенчмаркинг: процесс сравнения измеренной производительности со стандартным, общедоступным эталоном.

● **Binary Code (Бинарный код)**
Этот термин применяется к скомпилированным инструкциям, содержащимся в исполняемом файле. Двоичный код не читается человеком и может быть понятен процессору компьютера только во время выполнения программы.Исходный код, напротив, состоит из операторов, созданных программистом с помощью текстового редактора. Исходный код читается человеком, то есть любым, кто понимает соглашения, используемые в данном языке программирования («C», «C++» и т. д.), но не может быть выполнен процессором компьютера до тех пор, пока он не будет скомпилирован.

● **Binning (Биннинг)**
Биннинг: процесс синтаксического анализа (изучения) или классификации данных для ускорения и / или улучшения биометрического сопоставления.

● **BioAPI**
BioAPI - интерфейс прикладного программирования биометрии: определяет интерфейс прикладного программирования и интерфейс поставщика услуг для стандартного интерфейса биометрической технологии. BioAPI позволяет легко устанавливать, интегрировать или заменять биометрические устройства в общей архитектуре системы.

● **Biological Biometric Characteristic (Биологические неприобретенные биометрические характеристики)**
Биологические биометрические характеристики: биометрические характеристики, основанные в первую очередь на анатомических или физиологических характеристиках, а не на приобретенном поведении. Все биометрические характеристики в некоторой степени зависят как от поведенческих, так и от биологических характеристик. Примеры биометрических модальностей, для которых биологические характеристики могут доминировать, включают рисунок вен пальца, отпечаток пальца и рисунок вен руки.

● **Biometric Data (Биометрические данные)**
Биометрические данные: универсальный термин для компьютерных данных, созданных в ходе биометрического процесса. Он включает необработанные данные сенсорного сканирования, биометрические образцы, модели, шаблоны и / или результаты оценки сходства. Биометрические данные используются для описания информации, собранной во время процесса регистрации, проверки или идентификации, но не применяются к информации конечного пользователя, такой как имя пользователя, личные данные и авторизация доступа.

● **Biometric Sample (Биометрический образец)**
Биометрический образец: информация или компьютерные данные, полученные от устройства биометрического датчика. Образцы - изображения лица или отпечатка пальца.

● **Biometric System (Биометрическая система)**
Биометрическая система: несколько отдельных компонентов (таких как датчик, алгоритм сопоставления и отображение результатов), которые в совокупности составляют полностью работоспособную систему. Биометрическая система - это автоматизированная система, способная:
1. Получать биометрические шаблоны у конечного пользователя.
2. Извлекать и обрабатывать биометрические данные из этого образца.
3. Сохранять извлеченные данные в базе данных.
4. Сравнение биометрических данных с данными, содержащимися в одной или нескольких справочных источниках.
5. Определение того, насколько хорошо они совпадают, и указание того, была ли проведена идентификация или проверка личности.

Биометрическая система может быть компонентом более крупной системы.

● **Biometrics (Биометрия)**
Биометрия: общий термин, используемый в качестве альтернативы для описания характеристики или процесса.
В качестве характеристики:
Измеримые биологические (анатомические и физиологические) и поведенческие характеристики, которые можно использовать для автоматического распознавания.

Как процесс:
Автоматизированные методы распознавания человека на основе измеримых биологических (анатомо-физиологических) и поведенческих характеристик.

● **Bistatic Reader (Бистатичный считыватель)**
Бистатический RFID-считыватель или считыватель использует одну антенну для передачи радиочастотной энергии на RFID-метку и другую антенну для приема энергии, отраженной от метки.

● **Black Hat (Киберпреступник)**
Термин, используемый в хакерской культуре для описания типичного киберпреступника, который использует свои знания и навыки для совершения преступных действий: взлома программ/сайтов, кражи данных, шифрования информации с целью получения выкупа. В отличие от этичных хакеров (белых хакеров), которые помогают повысить безопасность систем, чёрные хакеры мотивированы исключительно материальной выгодой или общественным признанием.

● **BLE**
BLE (Bluetooth Low Energy) — это технология активного отслеживания на основе радиочастот. Она работает в диапазоне 2,40 ГГц и позволяет обнаруживать объекты с метками BLE в окружающей среде.

● **Bluetooth BR/EDR**
Классический Bluetooth также называется BR/EDR (Basic Rate/Enhanced Data Rate). Разница с BLE существенная: классический Bluetooth используется для непрерывной потоковой передачи данных, тогда как BLE данные передаются пакетами.

● **Blue Screen of Death (BSoD) («Синий экран смерти компьютера»)**
Название сообщения о системной ошибке Windows, возникающей после обнаружения кода по умолчанию или повреждения системного файла, приводящего к остановке работы системы. При возникновении ошибки на экране компьютера появляется белый текст на синем фоне. Текст содержит информацию, помогающую определить причину ошибки.● **Bluesnarfing**
Несанкционированный доступ к данным мобильного устройства через Bluetooth-соединение. Атаки Bluesnarfing появились на ранних этапах развития технологий передачи данных и подразумевали подключение к стороннему телефону или планшету без разрешения владельца устройства. В большинстве современных операционных систем и клиентских программ сопряжение двух узлов по Bluetooth требует подтверждения с обеих сторон, поэтому атаки Bluesnarfing в настоящее время встречаются крайне редко. Однако, в системах СКУД на основе BT / BT Mesh технологий часто отсутствует элемент авторизации подключаемого устройства, что создает дополнительные риски для технологий доступа.

● **Bootkit (Буткит)**
Это вредоносная программа, предназначенная для загрузки на максимально ранних этапах процесса загрузки, чтобы контролировать все этапы запуска операционной системы, изменяя системный код и драйверы до загрузки антивируса и других компонентов безопасности. Вредоносная программа загружается из главной загрузочной записи (MBR) или загрузочного сектора. По сути, буткит — это руткит, загружаемый до загрузки операционной системы.

● **[RUST] Borrowing (Заимствование) (&mut | &)**

Определение: Заимствование позволяет временно использовать значение без передачи владения. В Rust есть два типа заимствования: изменяемое **(&mut)** и неизменяемое **(&)**.

```
fn main() {
    let mut s = String::from("hello");
    let r1 = &s; // Неизменяемое заимствование
    let r2 = &mut s; // Изменяемое заимствование
    r2.push_str(", world");
    println!("{}", r1); // Ошибка: r1 не может использоваться после &mut
}
```

Нюансы: В Rust действует правило: **либо одно изменяемое заимствование, либо любое количество неизменяемых, но не оба одновременно.** Это предотвращает гонки данных **(data races)**.

Предупреждение: Нарушение правил заимствования приведёт к ошибке компиляции. Всегда проверяйте области видимости ссылок.

● **«Breakglass» (Разбивное число)**
Разбивное стекло — это тип аварийного аварийного выхода. Оно представляет собой небольшую стеклянную пластину, которая при разбивании активирует аварийный выход. После разбивания стекло необходимо заменить, прежде чем его можно будет использовать снова.

● **Brute-Force**
Это метод подбора пароля (или ключа, используемого для шифрования сообщения), который включает в себя систематический перебор всех возможных комбинаций символов до тех пор, пока не будет найдена правильная. Это может занять очень много времени, поэтому одним из вариантов является атака по словарю, хотя она работает только в том случае, если в качестве пароля использовалось обычное слово, а не комбинация букв, цифр и небуквенно-цифровых символов.

● **Buffer Overflow (Переполнение буфера)**
Ошибка в компьютерной программе, возникающая при попытке поместить в память блок данных, превышающий объем выделенного для него пространства.

● **Bug Bounty (Метод или программа поиска уязвимостей)**
Программа, стимулирующая поиск ошибок и уязвимостей в программном обеспечении. Разработчики приложений и сетевых платформ обычно объявляют премии за выявление уязвимостей, чтобы выявлять проблемы безопасности.

● **CAN (Controller Area Network)**
Протокол передачи данных, используемый для объединения в сеть разнородных устройств и обеспечения надёжной связи между ними. Чаще всего он представляет собой шину, поддерживающую передачу пакетов данных, доступных для приёма всеми подключёнными узлами.

● **Capacitor (Конденсатор)**
Элемент электрической цепи, используемый для временного хранения заряда. Конденсатор обычно состоит из двух металлических пластин, разделённых и изолированных друг от друга диэлектриком.

● **Capture (Сканирование)**
Сканирование: процесс сбора биометрической данных шаблона у человека с помощью датчика.

● **Card Cloning (Клонирование карт)**
В сфере контроля доступа клонирование карт — это процесс перехвата сигналов RFID-карт доступа и их копирования, как правило, в вредоносных целях. Наибольшему риску клонирования подвергаются низкочастотные карты доступа и карты без встроенной защиты или шифрования.

● **Card Printer (Принтер карт)**
Специализированный электронный принтер-кодировщик, используемый для печати и персонализации пластиковых карт, обычно изготавливаемых из ПВХ, для таких целей, как удостоверения личности, банковские карты, пропуска, ID и Smart-карты.

● **Card Operating System (Операционная система работы с картами)**
Программное обеспечение, хранящееся в микросхеме смарт-карты, которое управляет основными функциями карты, такими как связь с терминалом, управление безопасностью и управление данными в файловой системе смарт-карты.

● **Carrier Frequency (Несущая частота)**
Частота, используемая для передачи данных.

● **Carrier wave (Несущая волна)**
Радиоволна определённой частоты, модулированная или изменённая каким-либо образом для передачи данных. Амплитуда несущей волны может быть увеличена, например, для обозначения единицы или нуля в двоичном коде.

● **CC | Continuous Current (Постоянный ток)**
В запирании CC означает постоянный ток (continuous current) или постоянный ток. Это тип постоянного тока (DC), интенсивность которого не меняется с течением времени. Он позволяет системам поддерживать постоянный уровень напряжения независимо от колебаний тока.

● **Circular-polarized antenna** (Антенна круговой поляризации)
Антенна считывателя УВЧ-диапазона, излучающая радиоволны по круговой диаграмме направленности. Такие антенны используются в ситуациях, когда ориентация транспондера относительно считывателя меняется. Поскольку волны распространяются по круговой диаграмме, у них больше шансов достичь антенны.

● **Chain of Trust (Цепь Доверия)**
Метод проверки компонентов компьютерной системы для обеспечения её безопасности и целостности. Цепочка доверия работает по принципу: если элемент A считает элементы B и C надёжными, последние также считают друг друга безопасными. Если цепочка доверия имеет иерархическую структуру, корневые элементы гарантируют достоверность подчинённых. Цепочка доверия составляет основу системы SSL-сертификации, в которой удостоверяющие центры подтверждают безопасность веб-ресурсов.

● **Challenge Response (Ответ на вызов)**
Ответ на вызов: метод, используемый для подтверждения присутствия человека путем получения прямых ответов от него. Ответы могут быть как произвольными, так и непроизвольными. В добровольном ответе конечный пользователь сознательно отреагирует на то, что предлагает система. При непроизвольной реакции тело конечного пользователя автоматически реагирует на раздражитель. Ответ на запрос может использоваться для защиты системы от атак.

● **Check-in unit (Блок регистрации)**
Блок регистрации: книжный ларек со встроенным сканером.

● **Checkbit (Проверка битов)**
Контрольный бит, или бит чётности, добавляется к строке двоичного кода для обнаружения ошибок и проверки целостности данных. Значение контрольного бита — 0 или 1, в зависимости от количества единиц в строке. Наиболее распространённый тип проверки требует, чтобы количество единиц в строке было чётным: если количество уже чётное, контрольный бит устанавливается в 0; если оно нечётное, контрольный бит устанавливается в 1, чтобы сделать сумму чётной. Существует ещё один тип проверки, требующий нечётного количества единиц, но он встречается гораздо реже.

● **Checksum (Проверка суммы)**
Значение, полученное в результате применения криптографической хеш-функции к фрагменту данных, обычно к одному файлу. Сравнение сгенерированной контрольной суммы с суммой, предоставленной источником файла, помогает убедиться в подлинности копии файла и отсутствии ошибок.
В технологиях RFID - Код, добавляемый к содержимому блока данных, хранящегося на микрочипе RFID, который можно проверить до и после передачи данных с метки на считыватель, чтобы определить, были ли данные повреждены или утеряны. Циклический контроль избыточности является одной из форм проверки контрольной суммы.

● **Chip (Чип / Микропроцессор)**
Программируемый цифровой электронный компонент (также называемый микропроцессором), предназначенный для реализации функций центрального процессора (ЦП) в одной полупроводниковой интегральной схеме (ИС). Несколько микросхем могут выполнять функции ЦП в компьютерной системе, встраиваемой системе или карманном устройстве.

● **Chipless RFID tag (Безчиповая RFID-метка)**
RFID-метка, не зависящая от кремниевого микрочипа. В некоторых бесчиповых метках вместо кремниевых микрочипов используются пластик или проводящие полимеры. В других бесчиповых метках используются материалы, отражающие часть излучаемых радиоволн. Компьютер делает снимок отраженных волн и использует его как отпечаток пальца для идентификации объекта с меткой. Компании экспериментируют со встраиванием в бумагу волокон, отражающих радиочастотное излучение, для предотвращения несанкционированного фотокопирования некоторых документов. Бесчиповые метки, использующие встроенные волокна, имеют один недостаток при использовании в цепочке поставок: одновременно можно считывать только одну метку.

● **Cipher (Шифр)**
- Алгоритм, описывающий правила преобразования исходного текста сообщения в набор символов, непонятный стороннему наблюдателю. Он предписывает последовательность чётко определённых действий, которые необходимо выполнить для шифрования/расшифровки сообщения. Сложность шифра зависит от вспомогательной информации, называемой ключом. Без ключа преобразование зашифрованной информации в открытый текст крайне затруднительно или вообще невозможно.

● **Claim of Identity (Подтверждение идентичности)**
Подтверждение идентичности: заявление о том, что человек является или не является источником ссылки в базе данных. Претензии могут быть положительными (я в базе данных), отрицательными (я не в базе данных) или конкретными (я конечный пользователь 123 в базе данных).

● **Closed Loop Systems (Системы замкнутого цикла)**
Системы RFID-отслеживания, установленные внутри компании. Поскольку отслеживаемый объект никогда не покидает контроль компании, ей не нужно беспокоиться об использовании технологий, основанных на открытых стандартах.

● **Closed-set Identification (Закрытая идентификация)**
Закрытая идентификация Относится к биометрической задаче, в которой известно, что неопознанный человек находится в базе данных биометрических характеристик, и система пытается определить его / ее личность.

● **CMC - Cumulative Match Characteristic (Совокупная характеристика соответствия)**
CMC - кумулятивная характеристика соответствия: метод демонстрации измеренных показателей точности биометрической системы, работающей в закрытой задаче идентификации. Шаблоны сравниваются и ранжируются на основе их сходства. CMC показывает, как часто шаблон человека появляется в рейтингах (1, 5, 10, 100 и т.д.), в зависимости от коэффициента соответствия. CMC сравнивает ранг (1, 5, 10, 100 и т.д.) со скоростью идентификации.

● **Code Injection (Внедрение кода)**
- Внедрение кода — это манипуляция уязвимой программой с целью выполнения произвольного кода, при котором вредоносный код внедряется в запущенный процесс уязвимой программы. Это возможно, когда программа позволяет небезопасным пользовательским данным (например, из-за отсутствия проверки границ) стать частью исполняемого кода, что часто приводит к запуску системной оболочки. Обратите внимание, что вредоносный код выполняется с теми же привилегиями, что и уязвимая программа.

● **COM file**
- Исполняемый файл в операционных системах DOS и Windows. COM-объекты имеют простую структуру и хранят данные, содержимое стека и код в одном сегменте. COM-файлы в значительной степени уступили место улучшенному формату EXE.

● **Collision (Коллизия)**
Случай, в котором шифрование или хэш-функция создает одинаковый результат для двух или более заданных наборов входных данных.

● **Common (Общий контакт)**
В релейном соединении общий контакт — это подвижная часть. Он часто обозначается как COM или C. Когда реле выключено, общий контакт подключен к нормально замкнутому (НЗ) контакту. При включении реле общий контакт перемещается с НЗ (Нормально замкнутый) на нормально разомкнутый (Открытый) (НО) контакт.

● **Conducted power (Проводящая или Кондуктивная мощность)**
Кондуктивная мощность — это мощность радиочастотного излучения, передаваемая системой RFID на антенну. Обычно она рассчитывается или измеряется в месте соединения кабеля с антенной. В США правила Федеральной комиссии по связи требуют максимальной кондуктивной мощности 1 Вт.

● **Conductive ink (Электропроводящие чернила)**
Тип чернил, способных проводить сигнал, обычно содержащий порошкообразное серебро и углерод. С помощью проводящих чернил компании могут рисовать или печатать схемы на различных материалах. Проводящие чернила обеспечивают дешёвый способ печати печатных плат, например, на бумаге.

● **Contactless Smart Card (Бесконтактная Смарт-карта)**
Неловкое название для кредитной карты или карты лояльности, содержащей RFID-чип для передачи информации на считыватель без необходимости проведения через него. Такие карты ускоряют процесс оплаты, предоставляя покупателям больше удобства.

● **COS (Card Operation System) (Операционная система Карты)**
Операционная система управления карты.

● **Covert (Скрытые или тайные операции/действия)**
Скрытый: случай, когда биометрические образцы или данные карт собираются в месте, неизвестном посторонним. Пример скрытой среды может включать контрольно-пропускной пункт аэропорта, где изображения лиц пассажиров фиксируются и сравниваются со списком наблюдения без их ведома.

● **CPU – Central Processor Unit (Микропроцессор)**
Мозг компьютера, который управляет всеми остальными частями компьютера.

● **Cryptographic Co-Processor (Криптографический Со-процессор)**
Специальные схемы, выполняющие криптографические вычисления, такие как модульная арифметика и вычисления с большими целыми числами. Эти схемы добавляются к стандартному ядру процессора и поэтому называются сопроцессорами.

● **Cyclic Redundancy Check (Циклическая проверка избыточности битов)**
Метод проверки данных, хранящихся на RFID-метке, чтобы убедиться, что они не были повреждены или часть из них не утеряна.

● **Cypher (Шифр)**
Шифр (или шифр) — это метод преобразования сообщения с целью его сокрытия и сохранения смысла. В системах контроля доступа шифрование AES — это тип шифра, обычно используемый для защиты передачи данных через интернет. Подробнее о шифровании AES.

● **CC – Common Criteria (Общий Критерий)**
Общие критерии оценки безопасности информационных технологий (далее — Common Criteria или **CC**) — это международный стандарт (ISO/IEC 15408) для сертификации компьютерной безопасности. В настоящее время он имеет версию 2022, редакция 1. Общие критерии — это структура, в которой пользователи компьютерных систем могут указывать свои функциональные требования и требования к обеспечению безопасности (**SFR** и **SAR** соответственно) в Целевом показателе безопасности (**ST**) и могут быть взяты из Профилей защиты (**PP**). В отличие от стандарта FIPS 140[4], Common Criteria не приводит списка требований по безопасности или списка особенностей, которые должен содержать продукт. Вместо этого он описывает инфраструктуру (framework), в которой потребители компьютерной системы могут описать требования, разработчики могут заявить о свойствах безопасности продуктов, а эксперты по безопасности определить, удовлетворяет ли продукт заявлениям.).

● **C&C Server (Command and Control Server) (Сервер оперативного и командного управления)**
Сервер, который помогает мошеннику управлять ботнетом и отправлять вредоносные команды его участникам, регулировать шпионское приложение.

● **Compiler (Компиллятор)**
Утилита для преобразования исходного кода программы в команды для выполнения процессором. Компиляторы создают исполняемый файл на основе алгоритма, описанного с помощью языка программирования. Скомпилированные программы нельзя изменять; изменять и перекомпилировать можно только исходный код.

● **Contactless Smart Card (Бесконтактная Смарт Карта)**
Технология RFID может быть интегрирована в кредитные карты (или любые другие карты), что позволит осуществлять бесконтактную передачу информации.

● **Crack (Программа – Взломщик)**
Вредоносная программа, предназначенная для взлома системы безопасности программного обеспечения. Взломщик может изменять исполняемые файлы, библиотеки или настройки приложения, генерировать и подменять лицензионные ключи, а также выполнять другие действия, позволяющие обойти алгоритм

проверки ключей и обеспечить несанкционированный запуск приложения. Он может открыть доступ к полному функционалу приложения или продлить срок подписки на привязанные к нему онлайн-сервисы.

● **Credential Stuffing (Подбор данных)**
Тип атаки методом подбора паролей по словарю, используемый для взлома компьютерных систем и онлайн-сервисов. При использовании метода подбора паролей вместо распространённых комбинаций имени пользователя и пароля киберпреступники используют настоящие учётные данные, украденные со стороннего ресурса. Злоумышленники рассчитывают на то, что многие пользователи используют одни и те же учётные данные для разных онлайн-аккаунтов.

● **CER - Crossover Error Rate: (See EER - Equal Error Rate) (Частота перекрестных ошибок)**
CER - частота перекрестных ошибок

● **[RUST] Crate (Крейт) (.rs)**

Определение: Крейт — это единица компиляции в Rust, которая может быть библиотекой или исполняемым файлом. Корневой файл крейта — это обычно main.rs (для бинарного крейта) или lib.rs (для библиотеки).

```
// main.rs
mod my_module {
    pub fn hello() {
        println!("Hello from module!");
    }
}

fn main() {
    my_module::hello();
}
```

Нюансы: Крейты управляются через Cargo, инструмент сборки Rust. Внешние крейты подключаются через Cargo.toml.

● **Cross-Site Request Forgery, CSRF/XSRF (Браузерный метод кибератаки на основе запроса)**
Вид атаки, при котором киберпреступники используют ограничения протокола HTTP. При открытии страницы пользователем активируется вредоносный код, который заставляет браузер жертвы отправлять определённый запрос к веб-сервису мошенников (например, под видом загрузки изображения), чтобы мошенники могли использовать его в своих целях.

● **Cryptographic Algorithm (Криптографический алгоритм)**
Набор правил, используемых для кодирования информации таким образом, чтобы её могли прочитать только уполномоченные лица. Они позволяют создать шифротекст, который можно прочитать только после расшифровки.

● **Cryptographic Key (Криптографический ключ)**
Секретная последовательность символов, используемая криптографическим алгоритмом для преобразования обычного текста в зашифрованный текст и наоборот.

● **Cryptography (Криптография)**
Практика и изучение методов безопасной связи в присутствии третьих лиц. В более общем смысле, криптография занимается разработкой и анализом протоколов, препятствующих чтению личных сообщений третьими лицами или общественностью; различные аспекты информационной безопасности, такие как конфиденциальность данных, целостность данных, аутентификация и безотказность, играют ключевую роль в современной криптографии. Современная криптография находится на стыке математики, информатики и электротехники. Криптография применяется в таких областях, как банковские карты, компьютерные пароли и электронная коммерция.

● **CRYPTO-1**
Crypto-1 — проприетарный алгоритм шифрования, созданный NXP Semiconductors для использования в RFID-картах стандарта Mifare (Classic). Не является безопасным.
Crypto-1 является потоковым шифром, очень похожим на предшествующий Hitag2. Crypto-1 состоит из:
> одного 48-битного сдвигового регистра с обратной связью для хранения секретного состояния
> линейной функции
> двухуровневой нелинейной функции 20-в-1
> 16-битного регистра сдвига с линейной обратной связью, который используется при аутентификации. Некоторыми картами может использоваться как ГПСЧ.

● **Cryptor (Криптор)**
Криптор — это инструмент, предназначенный для обфускации кода вредоносного ПО, что затрудняет его обнаружение с помощью сигнатурного сканера. Иногда этот термин используется как синоним термина «криптовредиент».

● **CTE (Constant Tone Expression) (Передача постоянного тона)**
CTE означает Constant Tone Expression (выражение постоянного тона) и представляет собой чистый синусоидальный сигнал, помещаемый в конце модулированного сигнала и передаваемый метками в AoA (или передаваемый AoD в метки). Оценка CTE позволяет рассчитать азимут линии, соединяющей шлюз и метку, и, следовательно, направление, в котором можно найти метку.

● **CVSS (Common Vulnerability Scoring System) (Система оценки общей уязвимости)**
Открытый стандарт оценки серьёзности уязвимостей. CVSS был разработан Национальным консультативным советом по инфраструктуре США (NIAC). В создании и обновлении стандарта участвовали коммерческие компании, включая Microsoft и Cisco. Система поддерживается Форумом групп реагирования на инциденты и обеспечения безопасности (FIRST).

● **DAC (Дискреционный контроль доступа (СКУД))**
Дискреционный контроль доступа. Один из трёх основных методов контроля доступа, DAC, представляет собой систему, ориентированную на пользователя и наилучшим образом подходящую для небольших помещений. Конечный пользователь определяет права доступа, вручную и напрямую предоставляя доступ отдельным лицам. Это может быть выдача карты доступа или сообщение PIN-кода.

● **Daisy Chain (Гирляндная цепь)**
В электротехнике гирляндное соединение — это метод соединения устройств, при котором компоненты соединяются либо в кольцо, либо последовательно. Название происходит от сравнения с несколькими цветами ромашки, связанными в цепочку.

● **Data Field (Поле данных)**

Область памяти в микросхемах RFID, предназначенная для хранения определённого типа информации. Поля данных могут быть защищены (см. ниже) или перезаписаны, поэтому поле данных может содержать информацию о том, куда следует отправить объект. При изменении пункта назначения новая информация записывается в поле данных.

● **Data Field Protection (Защита блока критически важных данных)**
Возможность предотвращения перезаписи данных, хранящихся в определённой области памяти RFID-микрочипа. Компании могут захотеть защитить поле данных, в котором хранится электронный код продукта, который не меняется в течение всего срока службы продукта, с которым он связан

● **Data Loss Prevention (DLP) (Методы или Система защиты конфиденциальных данных)**
Набор методов и инструментов для обнаружения и предотвращения попыток кражи конфиденциальных данных. DLP-системы анализируют исходящий трафик и блокируют передачу конфиденциальной информации. Средства предотвращения утечек используют два основных метода: Анализ на основе формальных атрибутов — специальных тегов, заранее проставленных в документах. Анализ содержимого документов на основе ключевых слов, фраз и других элементов.

● **Data Retention (Сохранение данных)**
Способность микрочипа сохранять информацию, хранящуюся в EEPROM. RFID-метки и другие микрочипы обычно могут хранить данные в течение 10 лет и более, но срок хранения данных зависит от температуры, влажности и других факторов.

● **Data Synchronization (Синхронизация данных)**
Гармонизация информации между торговыми партнерами обеспечивает одинаковость основных данных во всех системах торговых партнеров.

● **Data Transfer rate (Параметр передачи данных)**
Количество символов, которое может быть передано с RFID-метки на считыватель за заданное время. Скорость передачи данных также используется для количественной оценки скорости считывания информации с RFID-метки считывателями. Это отличается от скорости считывания, которая определяет количество меток, которые могут быть считаны за заданный период времени.

● **dBi (Коэфициент усиления)**
Коэффициент усиления антенны по сравнению с изотропной антенной, то есть антенной, которая излучает энергию равномерно во всех направлениях. Типичная дипольная антенна имеет коэффициент усиления 2,2 dBi по сравнению с изотропной антенной.

● **DC | Direct Current (Постоянный ток)**
DC (постоянный ток) — это электрический ток, текущий только в одном направлении. Постоянный ток обычно используется в низковольтных электроприборах. Другой тип тока — переменный ток (AC).

● **DDoS (Distributed Denial of Service) attack (Кибератака с целью срыва работы системы посредством перегрузки потока данных)**
DDoS-атака (распределенный отказ в обслуживании) — это тип DoS-атаки, при котором целевой сервер, сервис или сеть перегружаются трафиком, исходящим из нескольких источников (например, группы устройств). Как и любая другая DoS-атака, цель DDoS-атаки — сделать систему жертвы недоступной.

● **DES (Data Encryption Standard) (Стандарт шифрования данных)**
DES (англ. *Data Encryption Standard*) — алгоритм для симметричного шифрования, разработанный фирмой IBM и утверждённый правительством США в 1977 году как официальный стандарт (FIPS 46-3). Размер блока для DES равен 64 битам. В основе алгоритма лежит сеть Фенистила с 16 циклами (раундами) и ключом,

имеющим длину 56 бит. Алгоритм использует комбинацию нелинейных (S-блоки) и линейных (перестановки E, IP, IP-1) преобразований. Для DES рекомендовано несколько режимов:
> ECB (англ. ***e****lectronic* ***c****ode* ***b****ook*) — режим «электронной кодовой книги» (простая замена);
> CBC (англ. ***c****ipher* ***b****lock* ***c****haining*) — режим сцепления блоков;
> CFB (англ. ***c****ipher* ***f****eed* ***b****ack*) — режим обратной связи по шифротексту;
> OFB (англ. ***o****utput* ***f****eed* ***b****ack*) — режим обратной связи по выходу;
> Counter Mode (CM) — режим счётчика.
Это преобразование над векторами (блоками), представляющими собой левую и правую половины регистра сдвига. В алгоритме DES используются прямое преобразование сетью Фейстеля в шифровании (см. Рис.1) и обратное преобразование сетью Фейстеля в расшифровании (см. Рис.2). Схема шифрования алгоритма DES указана на Рис.3.
Исходный текст — блок 64 бит. Процесс шифрования состоит из начальной перестановки, 16 циклов шифрования и конечной перестановки.

● 2DES / 3DES (Стандарт шифрования данных на основе 2 / 3 прогонов ключей)
Чтобы увеличивать криптостойкость DES, появляются несколько вариантов: double DES (2DES), triple DES (3DES), DESX, G-DES.
Методы 2DES и 3DES основаны на DES, но увеличивают длину ключей (2DES — 112 бит, 3DES — 168 бит) и поэтому увеличивается криптостойкость.
> **DES-EEE3:** Шифруется три раза с 3 разными ключами.
> **DES-EDE3:** 3DES операции шифровка-расшифровка-шифровка с 3 разными ключами.
> **DES-EEE2** и DES-EDE2: Как и предыдущие, за исключением того, что первая и третья операции используют одинаковый ключ.

● Deadbolt (Засов)
Засов — это элемент некоторых запирающих систем. Он состоит из засова (обычно металлического), который вручную перемещается ключом или ручкой, чтобы запереть или отпереть дверь.

● Decibel
Единица измерения, используемая для выражения отношения двух величин, включая коэффициент усиления антенны, потери в кабеле и выходную мощность считывателя. Формула для децибела: дБ = 10 log (P1/P2). Проще говоря, дБ представляет собой разницу в интенсивности излучаемого сигнала или мощности, где 0 дБ — опорная величина, 3 дБ — удвоенная интенсивность 0 дБ, 10 дБ — умноженная на 10 интенсивность, а 20 дБ — умноженная на 100 интенсивность и так далее. (См. также дБи, дБм и дБВт.).

● Decompiler (Декомпиллятор)
Компьютерная программа, которая принимает на вход исполняемый файл и пытается создать высокоуровневый компилируемый исходный файл, выполняющий те же функции. Декомпиляторы обычно не полностью восстанавливают исходный код и могут значительно различаться по степени читаемости результатов. Используется в основном как инструмент для обратной разработки программного обеспечения.

● Decryption (Расшифровка)
Расшифровка — это процесс преобразования сообщения, скрытого с помощью шифрования, обратно в его исходную, читаемую форму.

● Default Deny (Отказ по умолчанию)
Сценарий контроля приложений, означающий запрет любого приложения, не упомянутого в списках разрешенных приложений, подготовленных администратором. Для этого требуется заранее составить список разрешенных рабочих приложений.

● **Demodulation (Демодуляция)**
В радиопередаче демодуляция — это процесс преобразования радиоволн, отправленных передатчиком, обратно в пригодную для использования форму.

● **DES (Data Encryption Standard)** Спецификация шифрования данных, созданная IBM в начале 1970-х годов. DES (Data Encryption Standard) — это симметричный шифр: для шифрования и дешифрования данных используется один и тот же ключ. Он также является блочным шифром: преобразует блоки открытого текста фиксированной длины в блоки шифротекста той же длины.

● **Deserialization**
Восстановление исходного объекта из последовательности битов, полученной в результате сериализации. Такие методы обычно используются для передачи многомерных массивов данных в виде одномерного массива – текстового или двоичного файла.

● **De-Tune**
Антенны УВЧ настроены на приём RFID-волн определённой длины от считывателя, подобно тому, как тюнер автомобильного радио меняет антенну для приёма сигналов разных частот. Когда антенна УВЧ находится близко к металлу или металлическому предмету, её настройка может быть нарушена, что приведёт к снижению производительности.

● **DevSecOps (Development, Security and Operations)**
DevSecOps (разработка, безопасность и эксплуатация) — это методология, которая дополняет практику DevOps и включает в себя не только тесное сотрудничество между командами DevOps, но и применение лучших практик безопасности на каждом этапе жизненного цикла программного обеспечения.

● **Dielectric Constant (Константа Диэлектрика)**
Мера способности материала накапливать заряд при приложении электрического поля, или его «ёмкость». Если материал имеет высокую диэлектрическую проницаемость, он отражает больше радиочастотной энергии и сильнее расстраивает антенну, что затрудняет его обнаружение. Примерами материалов с низкой диэлектрической проницаемостью являются сухая бумага (2), пластик (большинство из них имеют диэлектрическую проницаемость от 2 до 4) и стекло (от 5 до 10). Диэлектрическая проницаемость воды меняется: при комнатной температуре она равна 80, вблизи кипения — 55, а при замерзании — 3,2.

● **Digital Certificate (Цифровой сертификат)**
Цифровое сообщение, содержащее идентификационные данные компании или организации, ее открытый ключ и подпись этих данных от центра сертификации (центра доверия), подтверждающую правильность этих данных.

● **Digital Signal Processor (Цифровой Сигнальный процессор)**
Как особый тип микропроцессора, преобразующего изменения аналоговых сигналов в цифровую информацию. Цифровые сигнальные процессоры используются в считывателях RFID.

● **Digital Signature (Цифровая «Сигнатура»)**
Криптографический протокол, обеспечивающий подлинность и целостность цифровых данных. Цифровая подпись обычно реализуется путём шифрования хеш-значения защищаемых данных с помощью закрытого ключа.

● **Digital ID (Цифровое ID)**
Цифровое представление предмета, включая уникальный код предмета (цифровой идентификатор), а также полный журнал событий и атрибутов, связанных с физическим предметом.

● **Diffie-Hellman Protocol (Протокол Деффи-Хеллмана)**
Протокол Диффи–Хеллмана — это криптографический протокол, позволяющий сторонам генерировать общий секретный криптографический ключ, обмениваясь данными по незащищённому каналу. Полученный ключ может быть использован как для шифрования, так и для дешифрования сообщений с помощью симметричных алгоритмов.

● **Dipole antenna (Дипольная антенна)**
В радио и телекоммуникациях наиболее распространённым типом антенны является дипольная антенна. Дипольная антенна обычно состоит из двух одинаковых проводящих элементов. В RFID-транспондере эти два элемента соединены с микрочипом.

● **DIN (DIN рейка)**
DIN-рейка — это стандартизированный тип металлической рейки, обычно крепящейся к стене и предназначенной для установки электрооборудования. DIN расшифровывается как Deutsches Institut für Normung (Немецкий институт нормирования), немецкая организация, которая первоначально опубликовала спецификации для этого типа реек.

● **Direct-to-Card (DTC) (Печать прямым образом на Карте)**
Direct-to-Card (DTC): технология печати, при которой печатающая головка наносит чернила непосредственно на поверхность карты. Эта технология обычно оставляет небольшую рамку по краю.

● **Disassembler (Дисассемблер)**
Утилита для преобразования исполняемого файла программы в исходный код на низкоуровневом языке ассемблера. Дизассемблирование восстанавливает текст программы для понимания её работы. Специалисты по информационной безопасности используют дизассемблеры для изучения вредоносных программ.

● **Discovery Services (Сервисы поиска данных)**
Компонент архитектуры EPCglobal, состоящий из набора сервисов, которые позволяют пользователям находить данные, хранящиеся в отдельных компаниях, связанных с определенным электронным кодом продукта. Сервис именования объектов является одним из компонентов сервисов обнаружения.

● **[RUST] | Diverges (Расходящиеся функции без возвращения)**
Для функций, которые не возвращают управление («расходящихся»), в Rust есть специальный синтаксис:

```
fn diverges() -> ! {
  panic! ("Эта функция не возвращает управление!");
}
```

panic! — это макрос, как и **println!()**, который мы встречали ранее. В отличие от println!(), panic!() вызывает остановку текущего потока исполнения с заданным сообщением. Поскольку эта функция вызывает остановку исполнения, она никогда не вернёт управление. Поэтому тип её возвращаемого значения обозначается знаком ! и читается как «расходится».
Если добавить функцию diverges() и запустить её, то вы получите следующее сообщение:

thread ‘<main>’ panicked at ‘Эта функция не возвращает управление!’, hello.rs:2

● **DMZ (Demilitarized zone) («Демилитаризованная» Зона)**
Контур локальной сети, изолированный от других блоков локальной сети, доступный из интернета)
Часть локальной сети, доступная из Интернета и изолированная от других ресурсов. Хотя она находится внутри общей компьютерной системы организации, она является относительно незащищённой областью и выполняет функцию своего рода буфера. Этот сегмент может использоваться для размещения веб-серверов или внешних хранилищ данных. Узлы в защищённой части локальной сети могут взаимодействовать с узлами DMZ через межсетевой экран.

● **DLL (Dynamic Link Library) (Динамическая библиотека связей)**
Универсальная подпрограмма, вызываемая основным приложением по мере необходимости. Как правило, библиотеки DLL реализуют стандартные, часто используемые функции, одинаковые для нескольких программ. Использование библиотек DLL экономит место на диске и упрощает код программы. Библиотеки DLL, содержащие набор объектов (например, значков) и возвращающие один из них по запросу программы, называются библиотеками ресурсов.

● **Door Station (Dynamic Link Library) (Станция двери)**
В системах аудио- или видеодомофонии или домофонии вызывная панель (иногда называют - dvor station) — это компонент, расположенный в точке входа с незащищённой стороны двери. Именно к ней подходят посетители и инициируют вызов на монитор.

● **Dry inlay (Инлей) (Сухой инлей)**
Сухой инлей (без клеевого слоя) представляет собой изделие, состоящее из антенны с микрочипом, расположенным поверх слоя неклейкой плёнки. Она представляет собой результат первого производственного процесса. В зависимости от достигнутого сочетания могут быть получены различные характеристики для различных областей применения.

● **D-Prime (D’)**
D-Prime (D ’): статистическая мера того, насколько хорошо система может различать сигнал и отсутствие сигнала.

● **Duplex printing (Двусторонняя печать)**
Двусторонняя печать: возможность автоматической печати на обеих сторонах карты за один цикл печати.

● **Duty cycle (Рабочий цикл)**
Продолжительность времени, в течение которого считывающее устройство может излучать энергию. Согласно правилам многих государств, считывающее устройство может быть включено только 10% времени.

● **Dye Sublimation Printing (Сублимационная цифровая печать)**
Сублимационная печать: цветные изображения переносятся на поверхность карты посредством нагрева, что позволяет получить насыщенные цвета и плавные градиенты. Обычно изображения защищаются прозрачным слоем для долговечности.

● **Edge server**
Компьютер для запуска промежуточного программного обеспечения или приложений, расположенный на периферии сети, где цифровой мир встречается с реальным. Периферийные серверы устанавливаются на складах, в распределительных центрах и на заводах, а не в штаб-квартирах компаний.

● **EDI - Electronic Data Interchange**
Эффективная изотропно излучаемая мощность. Измеряется выходная мощность антенн RFID-считывателей, используемых в США и других странах. EIRP обычно

выражается в ваттах.

● **EDR – Emergency Door Release (Аварийное открытие двери)**
EDR означает аварийное открывание дверей (Emergency Door Release). Это устройства, которые мгновенно отключают питание дверных замков при срабатывании в случае чрезвычайной ситуации. Они часто имеют яркую расцветку и срабатывают либо при разбивании стекла, либо при нажатии на кнопку с функцией

● **EEPROM (ПЗУ)**
Electrically Erasable Programmable Read-Only Memory \ Электрически стираемое перепрограммируемое ПЗУ
Метод хранения данных на микрочипах. Обычно байты можно стирать и перепрограммировать по отдельности. RFID-метки с EEPROM дороже заводских меток, где номер записывается в кремний при изготовлении чипа, но они обеспечивают большую гибкость, поскольку конечный пользователь может записать идентификационный номер в метку непосредственно перед её использованием.

● **Egress (Выход / Покидание объекта)**
Выход — это действие по выходу или оставлению какого-либо места или территории.

● **Electromagnetic Lock (Электромагнитный замок)**
Электромагнитный замок (также известный как магнитный замок) состоит из электромагнита и якорной пластины. При подаче питания на электромагнит создаётся магнитное поле, и между магнитом и якорной пластиной возникает сильное притяжение. При отключении питания магнитное поле исчезает, и дверь можно открыть. Магнитные замки просты в установке и отличаются прочностью и долговечностью.

● **Electronic Lock (Электромеханический замок)**
Электрозамок состоит из небольшого двигателя, который активируется электрическим импульсом. Этот импульс приводит в действие подвижные части, например, засов, запирающий дверь.

● **Electronic Strike (Электромеханическая защелка)**
Электромеханическая защёлка — это тип замка. Она состоит из защёлки с электроприводом и лицевой пластины. При активации защёлки, будь то подача или снятие питания, защёлка разблокируется, и дверь можно открыть. Электромеханические защёлки — это экономичные и долговечные замки.

● **Electronic Pedigree («Электронная родословная»)**
Защищённый файл, в котором хранятся данные о каждом перемещении продукта по цепочке поставок. Электронные родословные могут помочь снизить риск подделки лекарств и других товаров. EPCglobal ратифицировала стандарт электронных родословных для отрасли.

● **Encryption (Шифрование)**
Логическая трансформация данных таким образом, что их могут расшифровать и прочитать только те, кому они предназначены. В системах RFID шифрование используется для защиты информации, хранящейся на микросхеме транспондера, или для предотвращения перехвата связи между меткой и считывателем. В биометрии Шифрование используется для защиты данных шаблонов и процесса передачи информации посредством коммуникационных протоколов между считывателем и Сервером.

● **End-to-End Encryption (Сквозное Шифрование)**

Сквозное шифрование относится к системам, которые шифруют передачу данных между всеми компонентами системы. Этот уровень шифрования означает, что все точки передачи данных в системе надёжны и не могут быть легко перехвачены и расшифрованы.

● **[RUST] Enumerate (Перечисление) (emumerate( ))**

Если вы хотите отслеживать число прошедших итераций, используйте функцию **.enumerate()**.

с Интервалами:

```
    for (i, j) in (5..10) .enumerate() {
    println! ("i = {} и j = {}", i, j);
}
```

Выводит:

```
i = 0 и j = 5
i = 1 и j = 6
i = 2 и j = 7
i = 3 и j = 8
i = 4 и j = 9
```

Не забудьте написать скобки вокруг интервала.

с Итераторами:

```
    let lines = "привет\nмир\nhello\nworld".lines ();
    for (linenumber, line) in lines.enumerate() {
    println!("{}: {}", linenumber, line);
}
```

Выводит:

```
0: привет
1: мир
2: hello
3: world
```

● **EPC Reader (Считыватель EPC)**

RFID-считыватель, соответствующий стандартам EPCglobal, включая протокол радиоинтерфейса и протокол низкоуровневого считывания.

● **Etching (Вытравливание)**

Одна из нескольких технологий, используемых для изготовления антенны транспондера. В этом процессе антенна вытравливается из алюминия или других металлов с помощью химического раствора.

● **FailSAFE | Fail Safe Devices (Устройства с защитой от перебоев с открытие при перебоях)**
Замки с защитой от перебоев в электропитании разблокируются при отключении питания. Чтобы заблокировать дверь и предотвратить доступ, необходимо подать питание на замок.

● **FailSEC | Fail Secure Devices (Устройства с защитой от перебоев с запиранием)**
Замки с защитой от перебоев запираются при отключении питания. При подаче питания замок разблокируется, и дверь можно свободно открыть.

● **Failover (Отказоустойчивость)**
Отказоустойчивость — это встроенная в системы процедура обеспечения надёжности, обеспечивающая поддержание нормальной работы в случае сбоя или отказа. Отказоустойчивость обычно подразумевает использование дублирующей системы, которая активируется при обнаружении проблемы.

● **FC | Facility Code (Код Объекта)**
Код объекта отображается в строке двоичного кода, используемого для идентификации учётных данных доступа в соответствии с протоколом Wiegand. Двоичные коды, передаваемые между компонентами, делятся на несколько разделов, один из которых является общим для всех учётных данных на одном сайте или в одной системе. Это код объекта.

● **FAR / FMR / FRR / FTA / FTE**
> FAR - False Acceptable Rate: коэффициент ложного пропуска, вероятность ложной идентификации, то есть вероятность того, что система биоидентификации по ошибке признает подлинность пользователя, не зарегистрированного в системе. FAR - False Acceptance Rate: статистика, используемая для измерения биометрических характеристик при выполнении задачи проверки. Процент случаев ложного принятия системой, когда одно лицо неправильно сопоставлено с существующими биометрическими данными другого человека. Пример: Фрэнк утверждает, что он Джон, и система проверяет заявку.
> FMR - False Match Rate: вероятность того, что система неверно сравнивает входной образец с несоответствующим шаблоном в базе данных.
> FRR – False Rejection Rate: ошибка отказа при сверке. FRR - False Rejection Rate: статистика, используемая для измерения биометрических характеристик при выполнении задачи проверки. Процент случаев ложного отклонения системой. Ложный отказ происходит, когда человек не соответствует его / ее существующему биометрическому шаблону. Пример: Джон утверждает, что он Джон, но система ошибочно отклоняет это утверждение.
> FTA - Failure to Acquire: Неспособность биометрической системы собрать и / или извлечь полезную информацию из биометрического образца.
> FTE – Failure to Enroll: Отказ от регистрации: отказ биометрической системы сформировать надлежащую ссылку для регистрации для конечного пользователя. К типичным сбоям относятся конечные пользователи, которые не обучены должным образом предоставлять свои биометрические данные, датчик не собирает информацию правильно или полученные данные датчика недостаточного качества для разработки шаблона.

● **Factory Reading (Данные «Заводского программирования»)**
Некоторые устройства, предназначенные только для чтения, должны иметь идентификационный номер, записанный в кремниевый микрочип во время его изготовления. Процесс записи номера в чип называется заводским программированием. Эти данные нельзя перезаписать или изменить.

● **Far Field Communication (FFC)**
Антенны RFID-считывателей излучают электромагнитное излучение (радиоволны). Если RFID-метка находится за пределами одной полной длины волны от считывателя, она находится в «дальнем поле». Если она находится в пределах одной полной длины волны, она находится в «ближнем поле». Сигнал в дальнем поле затухает пропорционально квадрату расстояния от антенны, а сигнал в ближнем поле затухает пропорционально кубу расстояния от антенны. Таким образом, пассивные RFID-системы, использующие связь в дальнем поле (обычно системы УВЧ и СВЧ), имеют большую дальность считывания, чем системы, использующие связь в ближнем поле (обычно низко- и высокочастотные системы).

● **Fixed Card Reader (Фиксированный считыватель карт)**
RFID-считыватель, устанавливаемый на стену, дверной проем, ворота, стол, полку или другую стационарную или неподвижную конструкцию, позволяющий ИТ-системам считывать уникальные идентификационные номера и другое содержимое RFID-транспондеров.

● **[RUST] | For … in (Циклы «для диапазона») (for…in)**
Цикл **for** нужен **для повторения блока кода определённое количество раз**. Циклы **for** в Rust работают немного иначе, чем в других языках программирования. Например, в Си-подобном языке цикл **for** выглядит так:

```
   for x in 0..10 {
   println!("{}", x); // x: i32
}
```

Можно представить цикл более абстрактно:

for переменная in выражение {
***   тело_цикла***
}

Выражение — это **итератор**. Итератор возвращает серию элементов, где каждый элемент будет являться одной итерацией цикла. Значение этого элемента затем присваивается переменной, которая будет доступна в теле цикла. После окончания тела цикла, берётся следующее значение итератора и снова выполняется тело цикла. Когда в итераторе закончатся значения, цикл for завершается.
В нашем примере, **0..10** — это выражение, которое **задаёт начальное и конечное значение, и возвращает итератор**. Обратите внимание, что конечное значение не включается в него. В нашем примере будут напечатаны числа от 0 до 9, но не будет напечатано 10.
В Rust намеренно нет цикла **for** в стиле C. Управлять каждым элементом цикла вручную сложно, и это может приводить к ошибкам даже у опытных программистов на C.

● **Fork («Форк»)**
Программный продукт, созданный на основе кода другой системы и представляющий собой ветвь разработки исходного проекта. Большинство форков выпускаются под свободной лицензией с открытым исходным кодом.

● **Fraud (Злонамеренные незаконные действия)**
Умышленный обман с целью получения несправедливой или незаконной выгоды или лишения жертвы законного права. В контексте кибербезопасности кибермошенничество обычно представляет собой попытку обмана жертвы с целью получения доступа к ее банковскому счету, доступ в помещения или технические манипуляции, связанные с незаконным обогащением, либо в интересах враждебных структур.

● **Free space (Свободная зона)**
Термин, используемый для описания дальности считывания RFID-метки, которая ни к чему не прикреплена.

● **Frequency (Частота)**
Число повторений полной волны в течение одной секунды. 1 Гц равен одной полной волне за одну секунду. 1 кГц равен 1000 волн в секунду. RFID-метки обычно

используют низкие, высокие или сверхвысокие частоты. Каждая частота имеет свои преимущества и недостатки, которые делают её более подходящей для одних приложений, чем для других.
Частота электромагнитных волн, используемых для передачи данных от транспондера к считывателю в системе RFID. Стандартные частоты передачи для систем RFID: низкая частота (LF / НЧ, 125–148,5 кГц), высокая частота (HF / ВЧ, 13,56 МГц) и сверхвысокая частота (UHF / СВЧ, 400 МГц–1 ГГц). Отличительными характеристиками являются стоимость производства отдельных компонентов, скорость передачи данных и эффективный радиус действия.

● **Full Disk Encryption (FDE)**
Полное шифрование диска (FDE) или шифрование всего диска — это способ защиты информации путём шифрования всех данных на диске, включая временные файлы, программы и системные файлы. Некоторые системы полного шифрования диска оставляют загрузочный сектор диска незашифрованным, другие также шифруют его. После инициализации FDE вся информация автоматически шифруется при записи на диск и расшифровывается при чтении, если у пользователя есть разрешение.

● **[RUST] | Function (Функции) (fn…)**

Функции широко распространены в коде Rust. Вы уже познакомились с одной из самых важных функций в языке: функцией `main`, которая является точкой входа большинства программ. Вы также видели ключевое слово `fn`, позволяющее объявлять новые функции.

Код Rust использует *змеиный регистр (snake case)* как основной стиль для имён функций и переменных, в котором все буквы строчные, а символ подчёркивания разделяет слова. Вот программа, содержащая пример определения функции:

```
   fn main() {
   println! ("Hello, world!");

   another_function();
}

   fn another_function() {
   println! ("Another function.");
}
```

Для определения функции в Rust необходимо указать fn, за которым следует имя функции и набор круглых скобок. Фигурные скобки указывают компилятору, где начинается и заканчивается тело функции.

Мы можем вызвать любую функцию, которую мы определили ранее, введя её имя и набор скобок следом. Поскольку в программе определена another_function, её можно вызвать из функции main. Обратите внимание, что another_function определена *после* функции main в исходном коде; мы могли бы определить её и раньше. Rust не важно, где вы определяете свои функции, главное, чтобы они были определены где-то в той области видимости, которую может видеть вызывающий их код.

Мы можем определить функции, имеющие *параметры*, которые представляют собой специальные переменные, являющиеся частью сигнатуры функции. Когда у функции есть параметры, необходимо предоставить ей конкретные значения этих параметров. Технически конкретные значения называются *аргументы*, но в

повседневном общении люди обычно используют слова *параметр* и *аргумент* как взаимозаменяемые либо для переменных в определении функции, либо для конкретных значений, передаваемых при вызове функции.

В этой версии another_function мы добавляем параметр:

```
   fn main() {
   another_function(5);
}

fn another_function (x: i32) {
   println! ("The value of x is: {x}");
}
```

Объявление another_function содержит один параметр с именем x. Тип x задан как i32. Когда мы передаём 5 в another_function, макрос println! помещает 5 на место пары фигурных скобок, содержащих x в строке формата.

В сигнатурах функций вы *обязаны* указывать тип каждого параметра. Это намеренное решение в дизайне Rust: требование аннотаций типов в определениях функций позволяет компилятору в дальнейшем избежать необходимости использовать их в других местах кода, чтобы определить, какой тип вы имеете в виду. Компилятор также может выдавать более полезные сообщения об ошибках, если он знает, какие типы ожидает функция.

> Инструкции выполняют какое-либо действие и не возвращают значения
> Выражения вычисляются до результирующего значения

Каждая программа на Rust имеет по крайней мере одну функцию — main:

```
   fn main() {
}
```

Ключевое слово **fn** объявляет функцию. За ним следует её имя, пустые круглые скобки (поскольку эта функция не принимает аргументов), а затем тело функции, заключённое в фигурные скобки. Вот функция **foo:**

```
   fn foo() {
}
```

Итак, что насчёт аргументов, принимаемых функцией? Вот функция, печатающая число:

```
fn print_number (x: i32) {
    println! ("x равен: { }", x);
}
```

Вот полная программа, использующая функцию print_number:

```
fn main() {
    print_sum(5, 6);
}

fn print_sum(x: i32, y: i32) {
    println!("сумма чисел: {}", x + y);
}
```

Аргументы разделяются запятой — и при вызове функции, и при её объявлении.
В отличие от let, вы *должны* объявлять типы аргументов функции.

Как насчёт возвращаемого значения? Вот функция, которая прибавляет один к целому:

```
fn add_one(x: i32) -> i32 {
    x + 1
}
```

Функции в Rust возвращают ровно одно значение, тип которого объявляется после «стрелки». «Стрелка» представляет собой дефис (-), за которым следует знак «больше» (>). Заметьте, что в функции выше нет точки с запятой. Если бы мы добавили её, мы бы получили ошибку либо return ; буз ошибки.

> Указатели на Функции

Можно объявить имя, связанное с функцией:

```
let f: fn (i32) -> i32;
```

f — это имя, связанное с указателем на функцию, которая принимает в качестве аргумента i32 и возвращает i32. Например:

```
fn plus_one(i: i32) -> i32 {
    i + 1
}

// без вывода типа
let f: fn(i32) -> i32 = plus_one;

// с выводом типа
let f = plus_one;
```

Теперь мы можем использовать f, чтобы вызвать функцию:

```
# fn plus_one(i: i32) -> i32 { i + 1 }
# let f = plus_one;
let six = f(5);
```

> Функции с возвращаемыми значениями:
Функции могут возвращать значения коду, который их вызывает. Мы не называем возвращаемые значения, но мы должны объявить их тип после стрелки (->). В Rust возвращаемое значение функции является синонимом значения конечного выражения в блоке тела функции. Вы можете раньше выйти из функции и вернуть значение, используя ключевое слово return и указав значение, но большинство функций неявно возвращают последнее выражение. Вот пример такой функции:

```
fn five() -> i32 {
    5
}
fn main() {
    let x = five();
    println!("The value of x is: {x}");
}
```

В коде функции five нет вызовов функций, макросов или даже инструкций let — есть только одно число 5. Это является абсолютно корректной функцией в Rust. Заметьте, что возвращаемый тип у данной функции определён как -> i32. Попробуйте запустить этот код. Вывод будет таким:

```
The value of x is: 5
```

Значение 5 в five является возвращаемым функцией значением, поэтому возвращаемый тип - i32. Рассмотрим пример более детально. Здесь есть два важных момента: во-первых, строка let x = five(); показывает использование возвращаемого функцией значения для инициализации переменной. Так как функция five возвращает 5, то эта строка эквивалентна следующей:

```
let x = 5;
```

Во-вторых, у функции five нет параметров и определён тип возвращаемого значения, но тело функции представляет собой одинокую 5 без точки с запятой, потому что это выражение, значение которого мы хотим вернуть.

● **Gait: An individual's manner of walking (Индивидуальная манера движения)**
Походка: индивидуальная походка. Эта поведенческая характеристика находится на стадии исследования и разработки автоматизации.

● **Gateway (Шлюз)**
Шлюз соединяет одну сеть с другой. Например, интернет-шлюз контролирует доступ к интернету.

● **General Purpose Input/Output (Порты ввода и вывода данных)**
Порты на считывателе предназначены для взаимодействия с периферийным оборудованием самого считывателя. Например, устройство, такое как Intercom, Датчики или Сенсор QR-кода, могут быть подключены к порту ввода общего назначения (GPI), чтобы при пересечении луча сенсора объектом считыватель начинал считывание.
В качестве другого примера, исполнительное устройство может быть подключено к порту вывода общего назначения (GPO), чтобы при считывании метки (карты) открывался шлюз или автоматическая дверь.

● **Grabber (Граббер)**
Инструмент для копирования контента сайта по определённым параметрам. Как и веб-скрейперы, грабберы собирают информацию, которую затем можно проанализировать с помощью парсера. Также Граббером называют приложение или устройство, копирующее данные устрйств СКУД, считывателей или приложений мобильного доступа.

● **Handshake (Защищенный сеанс связи между Клиентом и Сервером в сети)**
Протокол для установления защищённого соединения между клиентом и сервером. Он предназначен для проверки подлинности и безопасности устанавливаемого канала и включает в себя ряд процедур контроля для предотвращения несанкционированного доступа. В процессе установления соединения запрашиваются данные аутентификации (в WPA2) или информация о сертификате безопасности (в SSL/TLS), генерируются ключи шифрования сеанса и проверяется целостность передаваемых пакетов данных.

● **Hamming Distance (Растояние Хамминга)**
Расстояние Хэмминга: количество несоответствующих цифр в строке двоичных цифр; используется для измерения несходства. Расстояния Хэмминга используются во многих алгоритмах распознавания радужной оболочки глаза Даугмана.

● **Hash (Хэш данных)**
Хеш или хеш-сумма — это последовательность символов фиксированной длины, полученная путём преобразования исходных данных (чисел, текста, файла и т. д.) с

помощью специального математического алгоритма; эта последовательность однозначно соответствует исходным данным, но не позволяет их восстановить. Процесс преобразования данных в хеш называется хешированием, а алгоритм хеширования — хеш-функцией. Большинство распространённых хеш-функций возвращают большие числа в шестнадцатеричном виде.

● **Hashing (Хэширование)**
Математический алгоритм, преобразующий случайный набор данных в строку фиксированной длины, состоящую из букв и цифр. Функция преобразования называется криптографической хеш-функцией, а результат преобразования — хешем. Алгоритм используется для сохранения паролей, обнаружения вредоносных программ и т. д.

● **Heap Overflow Attack (Атака на динамический блок кучи)**
Кибератака, при которой вносятся изменения в данные, временно хранящиеся в динамической памяти. Указатели соседних ячеек памяти перезаписываются, что позволяет потенциальным злоумышленникам получить доступ к информации, которая в противном случае была бы им недоступна, или выполнить произвольный код на устройстве. Целевые атаки такого типа сложно осуществить, поскольку практически невозможно предсказать, какие данные станут доступны после изменения относительной адресации.

● **Heap Spraying (Распыление «кучи»)**
Техника, используемая в эксплойтах, заключается в записи определённой последовательности байтов в различные места кучи — памяти, выделенной для использования программами. Эта техника напоминает распыление краски на стену, чтобы сделать её полностью одного цвета: подобно стене, куча «распыляется» так, чтобы содержащиеся в ней байты равномерно распределялись по всей её «поверхности» памяти.

● **Heaven's Gate («Небесные врата»)**
Метод запуска 64-битного кода в 32-битном процессе. Киберпреступники используют его для маскировки вредоносного ПО и обхода сканеров безопасности для доставки вредоносной нагрузки. Этот метод использует 64-битный обработчик, встроенный в 32-битный процесс Windows для обеспечения совместимости. Впервые описанный в середине 2000-х годов хакером под псевдонимом Рой Г. Бив, он использовался в нескольких вредоносных кампаниях.

● **HEX Value (Шестнадцатеричное значение)**
Шестнадцатеричное значение.

● **Hoax (Мистификация)**
Мистификация — это ложное предупреждение о вирусе или другом вредоносном коде.Обычно мистификация принимает форму электронного письма или другого сообщения, предупреждающего читателя об опасном новом вирусе и предлагающего ему передать это сообщение дальше.

● **Holding Force (Удерживающая сила)**
Удерживающая сила — это мера прочности замка. Измеряемая в килограммах или фунтах, удерживающая сила показывает вес, необходимый для взлома замка.

● **Honeypot (Медовый горшочек)**
В современном цифровом понимании «ханипот» — это ловушка в виде уязвимой или неправильно настроенной системы (например, маршрутизатора или виртуальной машины, эмулирующей его), намеренно открытой для атак из интернета. Находясь под постоянным контролем специалистов по информационной безопасности, ханипоты служат приманкой для киберпреступников. Основная задача «ханипота» — сбор информации об инструментах и тактике злоумышленников. «Ханипоты» относятся к пассивным средствам защиты. Чтобы привлечь киберпреступников, исследователи обычно намеренно используют такие меры, как защита «ханипотов» слабыми паролями или оставление часто используемых портов открытыми.

● **Hooking (Хукинг)**
Методы, используемые для изменения или дополнения поведения операционной системы, приложений или других программных компонентов путём перехвата вызовов функций, сообщений или событий, передаваемых между программными компонентами. Код, обрабатывающий такие перехваченные вызовы функций, события или сообщения, называется хуком.

● **Hopping Code (Переменный код)**
Переменный код, также известный как плавающий код, — это технология безопасности, используемая в радиочастотной передаче. Каждый раз при передаче сигнала между передатчиком и приёмником код, используемый для предоставления доступа, меняется. Это более безопасная технология, чем система с фиксированным кодом, поскольку код нельзя использовать повторно, а в случае перехвата он больше не может быть использован для получения доступа.

● **Hypervisor (Гипервизор)**
Программный продукт или аппаратное решение для управления виртуальными машинами, работающими в рамках одной системы. Гипервизор обеспечивает взаимодействие операционных систем в каждой из изолированных сред так, как если бы они были установлены на отдельных компьютерах.

● **IaaS (Infrastructure as a Service) (Инфраструктура как Сервис)**
Модель обслуживания для аутсорсинга удалённого оборудования с заданными параметрами. В рамках модели IaaS провайдер предоставляет клиенту доступ к виртуальному серверу с установленной операционной системой или к нескольким хостам, объединённым в сеть. Пользователь может работать с облачными ресурсами как с обычными компьютерами, включая установку программного обеспечения, назначение прав доступа и настройку параметров ОС.

● **Identification (Идентификация)**
Идентификация: задача, при которой биометрическая система ищет в базе данных ссылку, соответствующую представленному биометрическому образцу, и, если она найдена, возвращает соответствующую личность. Биометрические данные собираются и сравниваются со всеми ссылками в базе данных. Идентификация является «закрытой», если известно, что человек существует в базе данных. При «открытой» идентификации, иногда называемой «списком наблюдения», наличие человека в базе данных не гарантируется. Система должна определить, есть ли человек в базе данных, а затем вернуть его личность.

● **Identification Rate (Уровень Идентификации)**
Уровень идентификации: скорость, с которой человек в базе данных правильно идентифицирован.

● **[RUST] If… (Выражения if…)**
Выражение if позволяет выполнять части кода в зависимости от условий. Вы задаёте условие, а затем указываете: "Если это условие выполняется, выполните этот блок кода. Если условие не выполняется, не выполняйте этот блок кода".

Для изучения выражения if создайте новый проект под названием *branches* в каталоге *projects*. В файл *src/main.rs* поместите следующий код:

```
fn main() {
    let number = 3;
    if number < 5 {
        println!("condition was true");
    } else {
        println!("condition was false");
    }
```

```
}
```

Условие начинается с ключевого слова if, за которым следует условное выражение. В данном случае условное выражение проверяет, имеет ли переменная number значение меньше 5. Сразу после условного выражения внутри фигурных скобок мы помещаем блок кода, который будет выполняться, если результат равен true. Блоки кода, связанные с условными выражениями, иногда называют *ветками*, как и ветки в выражениях match, которые мы обсуждали в разделе "Сравнение догадки с секретным числом" главы 2.
Это необязательно, но мы также можем использовать ключевое слово **else**, которое мы используем в данном примере, чтобы предоставить программе альтернативный блок выполнения кода, выполняющийся если результат вычисления будет ложным. Если не указать выражение else и условие будет ложным, программа просто пропустит блок if и перейдёт к следующему фрагменту кода.

● **[RUST] Else…if… (Выражения Else… if…)**

Можно использовать несколько условий, комбинируя **if** и **else** в выражении else if. Например:

```
fn main() {
    let number = 6;
    if number % 4 == 0 {
        println! ("number is divisible by 4");
    } else if number % 3 == 0 {
        println!("number is divisible by 3");
    } else if number % 2 == 0 {
        println!("number is divisible by 2");
    } else {
        println! ("number is not divisible by 4, 3, or 2");
    }
}
```

● **[RUST] if… let (if в инструкции let)**

Поскольку if является выражением, его можно использовать в правой части инструкции let для присвоения результата переменной.

```
fn main() {
    let condition = true;
    let number = if condition { 5 } else { 6 };
    println!("The value of number is: {number}");
}
```

The value of number is: 5

Вспомните, что блоки кода вычисляются последним выражением в них, а числа сами по себе также являются выражениями. В данном случае, значение всего выражения if зависит от того, какой блок выполняется. При этом значения, которые могут быть результатами каждого из ветвей if, должны быть одного типа.

● **Induction communication (Индуктивная связь)**
В системе RFID, использующей индуктивную связь, антенны считывателя и транспондера имеют катушки, которые вместе образуют магнитное поле. Транспондер черпает энергию из этого поля. Микрочип использует эту энергию для изменения электрической нагрузки на антенне транспондера. Эти изменения улавливаются антенной считывателя и преобразуются в поток данных, содержащий уникальный серийный номер транспондера.

● **Ingress (Вход)**
Вход — это действие по вхождению или проходу в какое-либо место или область.

● **Infrared (Инфракрасный)**
Инфракрасный. Технологии использования ИК-спектра.

● **Interlock (Шлюз)**
В системе контроля доступа блокировка — это система, которая делает состояние одной двери зависимым от состояния другой. Например, в условиях чистого помещения пользователь не сможет отпереть и открыть дверь во внутреннюю зону, пока дверь во внешнюю зону не будет закрыта и заперта.

● **Inlay (Инлей)**
Микрочип RFID, прикреплённый к антенне и установленный на подложке. Инлей — это, по сути, необработанные RFID-этикетки в ПВХ листе. Иногда их также называют инлетами.

● **ID-1 Format (Формат ID-1)**
Формат ID-1: стандартные размеры карты 85,60 × 53,98 мм, используемые для большинства финансовых и идентификационных карт.

● **IDS (Intrusion Detection System) (Система обнаружения вторжений)**
Система обнаружения вторжений (IDS) — это программный продукт или устройство, которое обнаруживает несанкционированную и вредоносную активность в компьютерной сети или на отдельном хосте.

● **IEEE 802.1x (Стандарт Сетевой авторизации и делегирования прав доступа)**
Стандарт, описывающий процесс авторизации и разграничения прав доступа в большинстве современных сетевых систем. Спецификация IEEE 802.1x запрещает клиенту, не прошедшему процедуру аутентификации, полноценно использовать маршрутизатор. Стандарт разработан и поддерживается Институтом инженеров электротехники и электроники (IEEE).

● **Internet of Things (IoT) (Интернет вещей)**
Термин «Интернет вещей» (часто сокращённо IoT) используется для описания повседневных объектов, подключенных к Интернету и способных автоматически

собирать и передавать данные без участия человека. Интернет вещей охватывает любые физические объекты (не только традиционные компьютеры), которым можно назначить IP-адрес и которые могут передавать данные: бытовая техника, электросчётчики (или другие счётчики), автомобили, камеры видеонаблюдения и даже люди (например, сердечные имплантаты).
Интернет вещей представляет собой сеть физических устройств, транспортных средств, зданий и других предметов, оснащенных электроникой, программным обеспечением, датчиками, исполнительными механизмами или пассивными RFID-метками, а также сетевым подключением, которое позволяет этим объектам собирать и обмениваться данными.

● **Interoperability (В нашей сфере означает – Технологическую совместимость устройств и программных модулей)**
В вычислительной технике этот термин обозначает возможность обмена информацией и её использования разрозненными программными системами. В RFID этот термин обычно относится к способности меток и считывателей разных производителей взаимодействовать друг с другом.

● **IPv6**
Усовершенствованная версия IPv4 для передачи информации между устройствами в локальных и глобальных сетях. 128-битный идентификатор обеспечивает IPv6 большее адресное пространство. Адреса IPv6 состоят из восьми групп символов, разделённых двоеточием. Формат позволяет опускать нулевые группы и заменять их двойным двоеточием (например, адрес f9e:0:0:0:0:0:0:63ca можно записать как f9e::63ca).

● **ISO 11784**
Международный стандарт, определяющий частоты, скорость передачи данных, битовое кодирование и структуру данных транспондеров, используемых для идентификации животных.

● **ISO 7816**
ISO/IEC 7816 — стандарт относится к смарт-картам (в первую очередь контактным). Описывает форму карты, контактов, их расположение и назначение; протоколы обмена и некоторые аспекты работы с данными. Не смотря на то, что стандарт для «контактных» смарт-карт, для RFID (бесконтактных) идентификаторов он тоже применим, в частности для некоторых типов NFC-меток (карт), Mifare Plus и Desfire-интерфейсов.
> Стандарт можно назвать базовым для всех смарт-карт. Существует российская копия стандарта в виде ГОСТ Р ИСО/МЭК 7816.

● **Стандарт ISO 14443**
Все продукты «Семейства MIFARE» базируются на ISO 14443 Type A 13,56 МГц стандарте бесконтактных смарт-карт
> **ISO/IEC 14443** — стандарт, описывающий частотный диапазон, метод модуляции и протокол обмена данными посредством бесконтактных пассивных карт (RFID) ближнего радиуса действия на «магнитно-связанных индуктивностях». Стандарт можно отнести к первому поколению RFID-стандартов. Впоследствии вошёл в стандарт ISO/IEC 18000-3 и NFC-стандарты (ISO/IEC 18092 и ISO/IEC 21481). Российский аналог Стандарта - **(ГОСТ Р ИСО/МЭК 14443)**.
> Стандарт состоит из 4 частей, обозначенных цифрами 1 - 4. Например, ISO 14443-3:
- часть 1 определяет физические нормативы карт и условия нормальной работы;
- часть 2 определяет радиочастотные параметры и методы модуляции;
- часть 3 определяет протокол инициализации обмена (в основном это процедура антиколлизии — разделения нескольких карт в поле считывателя);
- часть 4 определяет протокол обмена данными.

> Стандарт содержит несколько разных протоколов обмена, обозначаемых буквами (A, B, и т. д.). Например ISO 14443A или ISO 14443A-4 (с указанием части). Первоначально стандарт содержал только протоколы A и B, однако добавлены несколько протоколов.
> Стандарт определяет использование свободного (нелицензируемого) диапазона частот 13,56 МГц с амплитудной модуляцией и девиацией 850 кГц. В России нормативным документом для использования этого частотного диапазона являются Приложения 4 и 9 к решению ГКРЧ от 7 мая 2007 года N07-20-03-001

● **Стандарт ISO 15961/2**
ISO 15961/2 Стандарты 15961 и 15962 определяют протоколы как для интерфейса приложений, так и для правил кодирования меток

и этикеток.

● **Стандарт ISO 18000**
ISO 18000-3 Для управления объектами семейство стандартов ISO 18000 является единственным важным стандартом на будущее.
ISO/IEC 18000-1 Часть 1: Общие параметры радиоинтерфейса для глобально принятых частот
ISO/IEC 18000-3 Часть 3: Параметры радиоинтерфейса связи на частоте 13,56 МГц.

● **iSCSI**
Компьютерный протокол для передачи команд SCSI по протоколу TCP/IP. iSCSI используется для создания распределённых систем хранения данных и динамического подключения дополнительного дискового пространства к серверу. Хранилище может располагаться на значительном расстоянии от сервера и взаимодействовать с центром управления по локальной сети или через Интернет.

● **JSON (JavaScript Object Notation)**
Открытый стандарт представления данных, основанный на JavaScript. Он используется для передачи структурированной информации и объектов данных в компьютерных программах. JSON использует методы организации данных, общие для большинства языков программирования: пары «ключ-значение» и упорядоченный список объектов. Отличительной особенностью стандарта является его понятный для человека текст и относительная простота по сравнению с XML.

Веб-токен JSON (JWT) - Интернет-стандарт для создания данных с дополнительной подписью и / или дополнительным шифрованием, полезная нагрузка которых содержит JSON, который утверждает определенное количество запросов. Токены подписываются с использованием личного секрета или открытого / закрытого ключа. Например, сервер может сгенерировать токен с утверждением «зарегистрирован как администратор» и предоставить его клиенту. Затем клиент может использовать этот токен, чтобы доказать, что он вошел в систему как администратор. Токены могут быть подписаны закрытым ключом одной стороны (обычно сервером), чтобы эта сторона могла впоследствии проверить, является ли токен законным. Если другая сторона каким-либо подходящим и заслуживающим доверия способом владеет соответствующим открытым ключом, она также может проверить легитимность токена. Маркеры спроектированы так, чтобы быть компактными, безопасными для URL и особенно удобными для использования в контексте единого входа (SSO) веб-браузера.

● **Kernel («Ядро»)**
Термин «ядро» относится к ядру операционной системы, поддерживающему все остальные операции. В отличие от этого, термин «оболочка» используется для описания пользовательского интерфейса.

● **KeeLoq®**
KeeLoq® — это запатентованный блочный шифр, который является одним из наиболее распространенных методов шифрования для передачи радиочастот.

● **Keylogger (Злонамеренные действия по перехвату нажатий клавиш клавиатуры или устройств)**
Используется третьей стороной для получения конфиденциальных данных (данных для входа в систему, паролей, номеров кредитных карт, PIN-кодов и т. д.) путём перехвата нажатий клавиш. Бэкдор-трояны обычно оснащены встроенным кейлоггером; конфиденциальные данные передаются удалённому киберпреступнику для использования в целях незаконного заработка или получения несанкционированного доступа.

● **Keystroke Dynamics (Динамика нажатия клавиш)**
Динамика нажатия клавиш: биометрический метод, в котором для распознавания используется частота набора текста человеком.

● **Kill Command (Команда деактивации чипа RFID-метки / Карты)**
Команда, отправленная на метку для полной деактивации её.

● **KRACK (Key Reinstallation Attack)**
Атака KRACK — это кибератака, направленная на взлом Wi-Fi-канала с шифрованием WPA2. На третьем этапе четырёхсторонней проверки криптоключа злоумышленник может повторно отправить секретную последовательность и снизить безопасность соединения. Атака KRACK может привести к перехвату трафика, подмене или передаче клиенту поддельных пакетов данных.

● **Lamination (Ламинирование)**
Ламинирование: добавление защитного слоя (пластиковой пленки) для повышения долговечности карты и ее устойчивости к подделке или выцветанию.

● **Latent Fingerprint (Скрытый отпечаток)**
Скрытый отпечаток пальца: «изображение» отпечатка пальца, оставленное на поверхности, к которой прикоснулся человек. Перенесенный отпечаток остается за счет контакта поверхности с гребнями трения, обычно вызванного масляными остатками, производимыми потовыми железами пальца.

● **Legic**
Стандарт разработан швейцарской компанией LEGIC Identbase AG. LEGIC – принадлежит HID
*см. детальное описание стандарта в каталоге **Идентификаторы RFID доступа.**

● **[RUST] Lifetime (Время жизни) (‘a)**

Определение: Время жизни — это аннотация ('a), указывающая, как долго ссылка остаётся действительной. Оно необходимо для обеспечения безопасности ссылок в функциях и структурах.

```
fn longest<'a>(x: &'a str, y: &'a str) -> &'a str {
    if x.len() > y.len() { x } else { y }
}

fn main() {
    let s1 = String::from("long");
    let s2 = String::from("short");
    let result = longest(&s1, &s2);
    println!("Longest: {}", result);
}
```

Нюансы: Компилятор **Rust** автоматически выводит времена жизни в большинстве случаев, но явные аннотации нужны, когда связи между входными и выходными ссылками неоднозначны.

● **Linear-Polarized antenna (Антенна линейной поляризации)**
Антенна, передающая радиоволны от считывателя в одной ориентации или полярности. Это увеличивает возможное расстояние считывания и может обеспечить

большую глубину проникновения через плотные материалы. Транспондеры, предназначенные для использования с антенной считывателя с линейной поляризацией, должны быть направлены в направлении, соответствующем антенне считывателя, для считывания. (См. также антенна с круговой поляризацией.).

● **Liveness Detection (Определение «живого» паттерна)**
Привязка к конкретному живому человеку: метод, используемый для подтверждения того, что отправленный биометрический образец принадлежит конечному пользователю. Метод определения живучести может помочь защитить систему от некоторых типов атак «спуфинга».

● **L2TP**
Протокол для организации виртуальных частных сетей (VPN). Обеспечивает туннелирование трафика в сетях на базе технологий IP, X.25, Frame Relay и ATM. В отличие от других спецификаций VPN, L2TP не имеет собственных средств шифрования данных и часто используется совместно с протоколом IPSec. L2TP/IPSec поддерживается большинством современного оборудования и считается одним из самых безопасных стандартов передачи данных. Недостатком L2TP является низкая скорость работы по сравнению с OpenVPN.

● **Living Off the Land (LOTL)**
Атака «Жизнь за счёт земли» (LotL) представляет собой кибератаку, при которой злоумышленники используют легитимное программное обеспечение и функции, доступные в системе, для совершения вредоносных действий. Жизнь за счёт земли означает выживание за счёт того, что можно добыть, добыть или вырастить в природе. Операторы кибератак LotL ищут в целевых системах инструменты, такие как компоненты операционной системы или установленное программное обеспечение, которые они могут использовать для достижения своих целей. Атаки LotL часто классифицируются как безфайловые, поскольку они не оставляют никаких артефактов.

● **[RUST] Loop (Циклы «петля») [RUST] (loop)**

Бесконечный цикл **(loop)** — простейшая форма цикла в Rust. С помощью этого ключевого слова можно организовать цикл, который продолжается, пока не выполнится какой-либо оператор, прерывающий его. Бесконечный цикл в Rust выглядит так:

```
loop {
    println! ("Зациклились!");
}
```

> Возвращение значений из циклов

Одно из применений loop - это повторение операции, которая может закончиться неудачей, например, проверка успешности выполнения потоком своего задания. Также может понадобиться передать из цикла результат этой операции в остальную часть кода. Для этого можно добавить возвращаемое значение после выражения break, которое используется для остановки цикла. Это значение будет возвращено из цикла, и его можно будет использовать, как показано здесь:

```
fn main() {
    let mut counter = 0;
    let result = loop {
        counter += 1;
```

```
        if counter == 10 {
            break counter * 2;
        }
    };
    println!("The result is {result}");
}
```

Перед циклом мы объявляем переменную с именем counter и инициализируем её значением 0. Затем мы объявляем переменную с именем result для хранения значения, возвращаемого из цикла. На каждой итерации цикла мы добавляем 1 к переменной counter, а затем проверяем, равняется ли 10 переменная counter. Когда это происходит, мы используем ключевое слово break со значением counter * 2. После цикла мы ставим точку с запятой для завершения инструкции, присваивающей значение result. Наконец, мы выводим значение в result, равное в данном случае 20.

● **MAC (Media Access Control) (Заводской Идентификатор сетевого устройства MAC)**
Уникальный идентификатор сетевого устройства, использующего соединение на основе одного из стандартов IEEE 802, например, Ethernet, Wi-Fi или Bluetooth. Он устанавливается на заводе и предназначен для однозначной идентификации каждого узла, а также для обеспечения точной маршрутизации пакетов в широковещательных сетях. ● **Транзакционный MAC-код.** Функция транзакционного MAC-кода надежно проверяет подлинность транзакции. Например, в системах с несколькими поставщиками услуг, использующими один и тот же кошелек и клиринговую службу, функция транзакционного MAC подтверждает клиринговой палате подлинность транзакций между поставщиками услуг и клиентами. Ее также можно использовать для проверки транзакций в многочисленных офлайн-сценариях или в случаях, когда возможности онлайн-аутентификации в режиме реального времени недоступны для операторов системы.

● **MAC filtering (MAC фильтр)**
Метод ограничения доступа к компьютерной сети посредством настройки разрешений устройству на основе MAC-адреса – уникального идентификатора сетевой карты. Эксперты по информационной безопасности отмечают, что фильтрация MAC-адресов мало способствует повышению безопасности сети, поскольку идентификатор может быть изменён в течение текущего сеанса (MAC-спуфинг). Запрос с поддельным MAC-адресом обрабатывается маршрутизатором как легитимный, даже если он отправлен с устройства, постоянный MAC-адрес которого занесён в стоп-лист.

● **MAC flooding (MAC – атака)**
Кибератака, направленная на компрометацию данных, передаваемых на устройство, подключенное к сетевому коммутатору. Метод основан на переполнении таблицы MAC-адресов устройств и соответствующих им сегментов сети. Эта база данных необходима для более точной маршрутизации пакетов, что позволяет осуществлять выборочную рассылку клиентам на коммутаторе.

● **MAC spoofing (MAC – спуфинг)**
Метод временного изменения MAC-адреса устройства для обхода блокировок доступа на уровне идентификатора сети (фильтрация MAC-адресов). Подмена адреса в большинстве систем не требует особых хакерских навыков и выполняется с помощью команды ipconfig или аналогичной. Как правило, настроенный таким образом MAC-адрес остаётся действительным до перезапуска устройства.

● **Man in the Disk (MitD)**
Тип кибератаки на устройства под управлением Android, при котором вредоносное ПО, установленное на смартфоне или планшете, атакует приложение через файлы, расположенные во внешнем хранилище. В отличие от внутренней памяти, приложения в этом пространстве не изолированы друг от друга, поэтому киберпреступники могут заменять или изменять хранящиеся там временные файлы или обновления.

● **Matching (Сверка)**
Сверка (Сопоставление): процесс сравнения биометрического образца с ранее сохраненным шаблоном и оценка уровня сходства (разница или расстояние Хэмминга). Затем системы принимают решения на основе этой оценки и ее отношения (выше или ниже) к заданному порогу.

● **[RUST] Match (Сопоставление)**

Простого **if/else** часто недостаточно, потому что нужно проверить больше, чем два возможных варианта. Да и к тому же условия в else часто становятся очень сложными. Как же решить эту проблему?
В Rust есть ключевое слово **match**, позволяющее заменить группы операторов **if/else** чем-то более удобным. Смотрите:

```
let x = 5;

match x {
    1 => println! ("один"),
    2 => println! ("два"),
    3 => println! ("три"),
    4 => println! ("четыре"),
    5 => println! ("пять"),
    _ => println! ("что-то ещё"),
}
```

match принимает выражение и выбирает одну из ветвей исполнения согласно его значению. Каждая *ветвь* имеет форму значение => выражение. Выражение ветви вычисляется, когда значение данной ветви совпадает со значением, принятым оператором match (в данном случае, x). Эта конструкция называется match (сопоставление), потому что она выполняет сопоставление значения неким «шаблонам». Глава «Шаблоны» описывает все шаблоны, которые можно использовать в match.
Так в чём же преимущества данной конструкции? Их несколько. Во-первых, ветви match *проверяются на полноту*. Видите последнюю ветвь, со знаком подчёркивания (_)? Если мы удалим её, Rust выдаст ошибку:

```
error: non-exhaustive patterns: `_` not covered
```

Другими словами, компилятор сообщает нам, что мы забыли сопоставить какие-то значения. Поскольку x — это целое число, оно может принимать разные значения — например, 6. Однако, если мы убираем ветвь _, ни одна ветвь не совпадёт, поэтому такой код не скомпилируется. _ — это «совпадение с любым значением». Если ни одна другая ветвь не совпала, совпадёт ветвь с _. Поскольку в примере выше есть ветвь с _, мы покрываем всё множество значений x, и наша программа скомпилируется.
match также является выражением. Это значит, что мы можем использовать его в правой части оператора let или непосредственно как выражение:

```
let x = 5;

let numer = match x {
    1 => "one",
    2 => "two",
    3 => "three",
```

```
    4 => "four",
    5 => "five",
    _ => "something else",
};
```

Иногда с помощью match можно удобно преобразовать значения одного типа в другой.

● **Memory Block (Блок памяти)**
Наименьшая единица памяти на RFID-метке (карте), которая может быть заблокирована независимо от других частей памяти.

● **Memory chip (Чип памяти)**
Интегральная схема, используемая в качестве устройства хранения данных.

● **Memory Scrapper (Скраппер памяти)**
Вредоносная программа, сканирующая оперативную память зараженных устройств (обычно POS-терминалов) для кражи конфиденциальных данных. Чаще всего скрейперы охотятся за номерами банковских карт и PIN-кодами. Уязвимы только карты с магнитной полосой. EMV-чипы защищены от этого типа атак.

● **Microcontroller (Микроконтроллер)**
Интегральная схема, включающая микроконтроллер, то есть высокоинтегрированную компьютерную микросхему, содержащую все компоненты, составляющие контроллер. Микроконтроллеры часто используются в приложениях, где безопасность и целостность данных имеют решающее значение, поскольку они обладают сложными функциями шифрования и позволяют выполнять сложные операции с данными.

● **Middleware (в контексте RFID-систем)**
Этот термин обычно используется для обозначения программного обеспечения, размещаемого на сервере между RFID-считывателями и ИТ-системами. Промежуточное программное обеспечение используется для фильтрации данных и передачи только полезной информации корпоративным приложениям. Некоторые виды промежуточного программного обеспечения также могут использоваться для удалённого управления и контроля считывателей в сети.

● **Mifare | Mifare Plus | Mifare DESFire**
MIFARE — торговая марка семейства бесконтактных смарт-карт. Торговая марка объединяет несколько типов микросхем смарткарт, микросхемы считывателей и продукты на их основе. Владельцем торговой марки является NXP Semiconductors.
Считается наиболее распространённой торговой маркой бесконтактных смарт-карт в мире: продано более 10 млрд смарт-карт и 150 млн считывателей.
*см. детальное описание стандарта в каталоге **Идентификаторы RFID доступа.**

● **Mimic (Мимик / Клон)**
Мимик: представление действующего биометрического показателя в попытке обманным путем выдать себя за кого-то, кроме отправителя.

● **Minutia(e) Point:**
Контрольная точка(е) Точка: характеристики гребня трения, которые используются для индивидуализации изображения отпечатка пальца. Контрольные точки - это точки, в которых гребни трения начинаются, заканчиваются или разделяются на два или более гребня. Во многих системах отпечатков пальцев мелкие детали (в отличие от изображений) сравниваются для распознавания.

● **MOV (Металлооксидный варистор)**
MOV (металлооксидный варистор) — это компонент, который используется в некоторых электронных устройствах для поддержания безопасного и стабильного уровня напряжения даже при непредвиденных скачках напряжения. Поглощая энергию и рассеивая её в виде тепла, MOV предотвращает повреждение уязвимых компонентов системы.

● **Multiplexer (Мультиплексер) (в нашем случае – двухантенный считыватель 125KHz + 13,56MHz)**
Электронное устройство, позволяющее считывателю использовать более одной антенны. Каждая антенна сканирует поле в заданном порядке. Это сокращает количество считывателей, необходимых для покрытия заданной зоны, например, ворот дока, и предотвращает взаимные помехи между антеннами.

● **Multiple Access schemes (Множественные схемы доступа)**
Методы увеличения объёма данных, передаваемых по беспроводной связи в одном частотном спектре. Некоторые RFID-считыватели используют технологию множественного доступа с временным разделением (TDMA), что означает, что они считывают метки в разное время, чтобы избежать помех друг другу.

● **[RUST] | Moving Semantic (Семантика перемещения):**
Rust гарантирует, что существует *ровно одно* связывание какого-либо ресурса. Например, если у нас есть вектор, то мы можем присвоить этот вектор другому имени:

```
let v = vec![1, 2, 3];
let v2 = v;
```

Но, если после этого мы попытаемся использовать **v**, то получим ошибку:

```
let v = vec![1, 2, 3];
let v2 = v;
println!("v[0] = {}", v[0]);
```

Ошибка выглядит следующим образом:

```
error: use of moved value: `v`
println!("v[0] = {}", v[0]);
                     ^
```

То же самое произойдёт, если мы определим функцию, которая принимает владение, и попробуем использовать значение после того, как мы передали это значение в качестве аргумента в эту функцию:

```
fn take (v: Vec <i32>) {
    // что будет здесь не очень важно
}

let v = vec! [1, 2, 3];

take(v);
println!("v [0] = {}", v [0]);
```

Та же самая ошибка: **«use of moved value»** («используется перемещённое значение»). Когда мы передаём право владения куда-то ещё, мы как бы говорим, что мы «перемещаем» то, на что ссылаемся. При этом не нужно указывать какую-либо специальную аннотацию, Rust делает это по умолчанию.

> Подробности:

Причина, по которой мы не можем использовать значение после того, как мы его переместили, неочевидна, но очень важна. Когда мы пишем код вроде этого:

```
let v = vec! [1, 2, 3];
let v2 = v;
```

Первая строка создаёт некоторые данные для вектора в стеке, **v**. Данные самого вектора, однако, сохраняются в куче, и поэтому стековые данные содержат указатель на данные в куче. Когда мы перемещаем **v** в **v2**, то создаётся копия стековых данных для **v2**. Что будет означать, что два указателя ссылаются на расположенный в куче вектор. Такое поведение могло бы быть проблемой: оно нарушало бы гарантии безопасности Rust, привнося гонки по данным. Поэтому Rust запрещает использование v после того, как мы выполнили его перемещение.

Важно также отметить, что оптимизация может удалить саму копию байтов на стеке, в зависимости от обстоятельств. Так что это может быть не так уж неэффективно, как выглядит на первый взгляд.

● **Modulation (Модуляция)**
Изменение радиоволн, проходящих между считывателем и транспондером, таким образом, чтобы обеспечить передачу информации. Волны могут быть изменены различными способами, которые могут быть считаны считывателем и преобразованы в единицы и нули двоичного кода. Волны можно сделать выше или ниже (амплитудная модуляция) или сдвинуть вперёд (фазовая модуляция). Можно изменять частоту (частотная модуляция), а данные можно хранить в длительности импульсов (широтно-импульсная модуляция).

> Амплитудная модуляция (АМ): данные содержатся в изменениях амплитуды несущей.
> Частотная модуляция (ЧМ): данные содержатся в изменениях частоты несущей.
> Частотная манипуляция (ЧМн): данные содержатся в изменениях между двумя частотами несущей.
> Фазовая модуляция (ФМ): данные содержатся в изменениях фазы несущей.

> Длительно-импульсная модуляция (ШИМ): данные содержатся в длительности импульсов. Также известна как широтно-импульсная модуляция (ШИМ).
> Фазово-импульсная модуляция (ФИМ): данные содержатся в положении импульсов относительно опорной точки.
> Непрерывно-волновая модуляция (НВ): данные содержатся в несущей, которая включается и выключается.

● MySQL
Универсальная мультиплатформенная СУБД, основанная на реляционной модели. MySQL предназначена для небольших и средних приложений. Она предоставляет разработчикам выбор различных типов таблиц для решения различных задач. СУБД входит в состав нескольких серверных программных продуктов и распространяется как по свободной, так и по коммерческой лицензии. С 2010 года права на MySQL принадлежат Oracle.

● N/C | Normally Closed (Нормально закрытый)
N/C (НЗ) означает «нормально замкнутый». Это одно из двух состояний переключателей, датчиков или контактов реле в электронной системе. Другое — нормально разомкнутый (N/O). Н/З контакт остаётся замкнутым до тех пор, пока не наступит определённое условие, после чего он размыкается.

● N/O | Normally Open (Нормально открытый)
N/O (НО) означает «нормально разомкнутый». Это одно из двух состояний переключателей, датчиков или контактов реле в электронной системе. Другое — нормально замкнутый (НЗ). НО контакт остаётся разомкнутым до тех пор, пока не наступит определённое условие, после чего он замыкается.

● .NET (Фреймворк)
Фреймворк Microsoft, состоящий из среды разработки и модуля исполнения. Программы, написанные на **Visual C++, Visual Basic, Visual F#** и других поддерживаемых языках, транслируются в единый байт-код. Последний интерпретируется средой Common Language Runtime, которая либо запускает скомпилированный файл в своей виртуальной машине, либо передаёт его инструменту **NGen.** Такая архитектура позволяет разрабатывать универсальные приложения для любых устройств, на которых установлен фреймворк. Ключевой особенностью **.NET** является возможность использования библиотек, написанных на разных языках, в рамках одной программы. Платформа ориентирована на среду Windows и входит в дистрибутив этой операционной системы.

Что такое .NET?

.Net (читается как «дот нет») – это кроссплатформенная среда выполнения приложений. Проще говоря – это то, что позволяет запускаться нашим приложениям в системе **Microsoft Windows**. Кроссплатформенная – означает, что созданное приложение будет работать на всех процессорах и на всех операционных системах семейства Windows (за исключением самых ранних).
Более того! Те, кто уже имел дело с программированием, например, на **C++,** знает что под процессоры на разной платформе приходится «пересобирать» программы. Например программа, скомпилированная для **x64** не будет корректно работать на x86, а программа, собранная для x86 не сможет полностью показать свой потенциал работы на x64 системе.
Тут нам на помощь приходит **.Net framework**.
.Net Framework – это набор уже скомпилированных библиотек, откуда берутся методы и функции для запуска и разработки приложений. В разработке, на деле, нам придётся просто вызвать уже готовую функцию для того, чтобы она заработала. Большинство методов и функций, необходимых программисту, уже скомпилировано и лежит **в .Net framework** внутри системы. И каждая библиотека с функциями лежит в двух вариантах – для **x86** и для **x64**, так что о **«пересборке»** программы под разные платформы можно забыть! Созданная вами программа будет показывать свой полный потенциал на любой аппаратной («железе») и программной (операционной системе) платформе.

Как это всё работает?

Вспомним, что такое процесс компиляции – это перевод вашего кода, понятного человеку, в бинарный код, понятный компьютеру.

В программировании на **.Net** компиляция и запуск приложений происходит следующим образом:
Код из любого языка преобразовывается в код, написанный на общем языке **(Common intermediate language или CIL)**. Этот язык является языком низшего уровня, похожего по синтаксису на язык ассемблер.
После, этот код передаётся так называемой исполняющей среде (**Common language runtime или CLR**), которая берёт функции и методы из **.Net Framework**
После этого конечный результат передаётся на процессор и выполняется программа.

● **Near-to-Edge (Почти «в край»)**
Почти в край: экономичный метод печати, при котором печать выполняется на расстоянии до 5 мм от края карты, не обеспечивая полный обрез.

● **NetBIOS**
Протокол связи для обнаружения устройств в локальной сети и передачи информации между ними. Каждому узлу назначается идентификатор **NetBIOS**, а **WINS-**сервер ведёт базу данных сопоставлений IP-адресов с именами NetBIOS. NetBIOS использует транспортные протоколы **UDP** и **TCP**.

● **Next Generation Firewall (NGFW)**
Межсетевой экран нового поколения (NGFW) — это межсетевой экран глубокой фильтрации, интегрированный с системой обнаружения вторжений (IDS) или системой предотвращения вторжений (IPS) и способный контролировать и блокировать трафик на уровне приложений.
Межсетевые экраны нового поколения также обеспечивают микросегментацию сети на основе приложений, а не только портов и IP-адресов. Обычно они представляют собой отдельные устройства, но существуют межсетевые экраны нового поколения в виде виртуальной машины или облачного сервиса.

● **NFC (Near Field Communication)**
Технология беспроводной передачи данных, позволяющая обмениваться данными в непосредственной близости без риска их потери или перехвата.
Преимуществами технологии являются короткое время соединения (менее одной десятой секунды) и отсутствие необходимости подключения к интернету.
Технология в основном используется для мобильных бесконтактных платежей.
Эта технология — простое расширение стандарта бесконтактных карт (ISO 14443), которое объединяет интерфейс смарт-карты и считывателя в единое устройство.
Устройство NFC может поддерживать связь и с существующими смарт-картами, и со считывателями стандарта ISO 14443, и с другими устройствами NFC и, таким образом, — совместимо с существующей инфраструктурой бесконтактных карт.
Существуют два режима: пассивный и активный. В пассивном режиме связи устройство-инициатор обеспечивает несущее поле, а целевое устройство отвечает посредством модулирования имеющегося поля. В этом режиме целевое устройство может вытягивать свою рабочую мощность из предоставленной Инициатором электромагнитной области, таким образом делая целевое устройство ретранслятором. В активном режиме связи и инициатор, и целевое устройство взаимодействуют путём поочерёдного создания своих собственных полей. Устройство дезактивирует своё радиочастотное поле в то время, как оно ожидает данных.
В активном режиме у обоих устройств должно быть электропитание.

Для передачи данных NFC использует два различных вида кодирования. Если активное устройство передаёт данные со скоростью 106 кбод, тогда используется модифицированный код Миллера со 100%-й модуляцией. Во всех других случаях используется манчестерское кодирование с коэффициентом модуляции 10 %.
Устройства NFC в состоянии одновременно и получать, и передавать данные. Таким образом, они могут контролировать радиочастотное поле и обнаруживать противоречия, если полученный сигнал не соответствует переданному.

● **NFT (Non-Fungible Token) (Невзаимозаменяемый токен)**
NFT (англ. non-fungible token, в переводе с англ. — «невзаимозаменя́емый то́кен»), также уника́льный то́кен — вид криптографических токенов, каждый экземпляр которых уникален (специфичен) и не может быть заменён или замещён другим аналогичным токеном, хотя обычно токены взаимозаменяемы по своей природе. Невзаимозаменяемый токен представляет собой криптографический сертификат цифрового объекта с возможностью передавать сертификат через механизм, применяемый в криптовалютах[4][5]. Сам по себе токен не является подтверждением права на владение цифровым активом в контексте законодательства об авторском праве. NFT не препятствует копированию объекта, он только закрепляет за владельцем «цифровую фишку», созданную на основе одного из экземпляров цифрового артефакта. Также нет препятствий для формирования нескольких разных токенов для одного и того же файла. Но если участники соглашаются, что NFT может формировать только реальный владелец и дальнейшая передача NFT происходит только в связи с передачей соответствующих прав на исходный цифровой объект, тогда NFT может выполнять роль маркера, указывающего текущего владельца объекта.

● **NIR (Near Infrared) (Околоинфракрасный)**
БИК (NIR) означает ближнее инфракрасное излучение (Near-Infrared Radiation). Это диапазоны излучения в инфракрасном спектре, наиболее близкие к красному видимому свету. В системах контроля доступа БИК используется в технологиях распознавания лиц для улучшения сканирования и анализа лиц в условиях низкой освещённости.

● **Noise (Электромагнитный шум)**
Шум: нежелательные компоненты в сигнале, которые ухудшают качество данных или мешают желаемым сигналам, обрабатываемым системой.

● **Non-Cooperative User (Не осведомленный о сборе данных пользователь)**
Пользователь, не сотрудничающий в рамках биометрической идентификации: человек, который не знает, что его биометрический образец собирается. Пример: путешественник, проходящий через линию безопасности в аэропорту, не подозревает, что камера фиксирует изображение его лица.

● **Non-volatile memory (Энергонезависимая память)**
Общий термин для обозначения памяти, сохраняющей своё содержимое после отключения питания. Примерами энергонезависимой памяти являются EPROM, EEPROM и FLASH.

● **Null Spot (Нулевая точка или «Слепая зона»)**
Область в поле зрения считывателя, которая не принимает радиоволны. По сути, это «слепая зона» считывателя. Это явление характерно для систем УВЧ (UHF).

● **Obfuscation (Обфускация / Запутывание)**
Процесс преобразования программного кода в форму, сложную для понимания человеком, при этом функции программы остаются прежними. Цели обфускации включают сохранение конфиденциальности разработки программы, а также защиту от киберпреступников, атакующих своих жертв, используя уязвимости кода. Программа, изменяющая код, называется обфускатором.

● **Ontology (Онтология)**
Онтология в информационных системах — это структура, состоящая из набора понятий и категорий, относящихся к определённой области знаний, а также информации об их свойствах и связях между ними. В контексте информационной безопасности можно говорить о киберонтологии или онтологии кибербезопасности.

● **One-to-many (1:N) (Поиск и сверка - Один из множества)**
Один ко многим: термин, используемый в биометрическом сообществе для описания системы, которая сравнивает одну ссылку со многими зарегистрированными ссылками для принятия решения. Эта фраза обычно относится к задачам идентификации или списков наблюдения.

● **One-to-one (1:1) (Поиск и сверка - Один к одному)**
Один к одному: термин, используемый в биометрическом сообществе для описания системы, которая сравнивает одну ссылку с другой зарегистрированной ссылкой для принятия решения. Эта фраза обычно относится к задаче проверки (хотя не все задачи проверки действительно взаимно однозначны), а задача идентификации может быть выполнена серией однозначных сравнений.

● **OpenID**
Открытый децентрализованный стандарт аутентификации пользователей. Создав учётную запись у провайдера OpenID, можно использовать одни и те же учётные данные для входа на различные онлайн-ресурсы, использующие OpenID. Более того, пользователь самостоятельно определяет, какую учётную запись использовать для входа и какая информация будет доступна сторонним сайтам.

OpenID Connect (OIDC) - это программный промежуточный протокол идентификации, созданный на основе платформы OAuth 2.0. Это позволяет сторонним приложениям проверять личность конечного пользователя и получать базовую информацию профиля пользователя. OIDC использует веб-токены JSON (JWT), которые можно получить с помощью потоков запросов, соответствующих спецификациям OAuth 2.0.

● **[RUST] Operators vs. Experssions (Операторы и Выражения)**

Rust — в первую очередь язык, ориентированный на **выражения**. Есть только два типа операторов, а всё остальное является выражением.

А в чём же разница? Выражение возвращает значение, в то время как оператор - нет. Вот почему мы получаем здесь **«not all control paths return a value»**: оператор `x + 1;` не возвращает значение. Есть два типа операторов в Rust: **«операторы объявления»** и **«операторы выражения»**. Все остальное — выражения. Давайте сначала поговорим об операторах объявления.
Оператор объявления — это связывание. В некоторых языках связывание переменных может быть записано как выражение, а не только как оператор. Например, в Ruby:

```
x = y = 5
```

Однако, в Rust использование `let` для связывания *не является* выражением. Следующий код вызовет ошибку компиляции:

```
let x = (let y = 5); // expected identifier, found keyword `let`
```

Здесь компилятор сообщил нам, что ожидал увидеть выражение, но `let` является оператором, а не выражением.
Обратите внимание, что присвоение уже связанной переменной (например: `y = 5`) является выражением, но его значение не особенно полезно. В отличие от других языков, где результатом присваивания является присваиваемое значение (например, `5` из предыдущего примера), в Rust значением присваивания является пустой кортеж `()`.

```
let mut y = 5;
let x = (y = 6)  // x будет присвоено значение `()`, а не `6`
```

Вторым типом операторов в Rust является *оператор выражения*. Его цель - превратить любое выражение в оператор. В практическом плане, грамматика Rust ожидает, что за операторами будут идти другие операторы. Это означает, что вы используете точку с запятой для отделения выражений друг от друга. Rust выглядит как многие другие языки, которые требуют использовать точку с запятой в конце каждой строки. Вы увидите её в конце почти каждой строки кода на Rust.

Из-за чего мы говорим «почти»? Вы это уже видели в этом примере:

```
fn add_one(x: i32) -> i32 {
    x + 1
}
```

Наша функция объявлена как возвращающая i32. Но если в конце есть точка с запятой, то вместо этого функция вернёт (). Компилятор Rust обрабатывает эту ситуацию и предлагает удалить точку с запятой.

> Досрочный возврат из Функции:

А что насчёт досрочного возврата из функции? У нас есть для этого ключевое слово return:

```
fn foo(x: i32) -> i32 {
    return x;

    // дальнейший код не будет исполнен!
    x + 1
}
```

return можно написать в последней строке тела функции, но это считается плохим стилем:

```
fn foo(x: i32) -> i32 {
    return x + 1;
}
```

● **OPSEC (Operations Security) (Операции по безопасности)**
OPSEC (сокращение от Operations Security) — это процесс выявления и защиты критически важной информации. Принципы OPSEC были первоначально разработаны военными США для предотвращения утечки разнородных фрагментов данных, которые при объединении могут раскрыть более крупный фрагмент секретной информации. Сегодня эти принципы используются как специалистами по информационной безопасности для снижения риска утечки конфиденциальной информации, так и киберпреступниками для избежания обнаружения.

● **Oscillation (Колебание)**
Колебание — это процесс возвратно-поступательного движения между двумя точками в регулярном ритме. При радиопередаче радиоволны многократно колеблются от самой высокой точки до самой низкой, распространяясь по воздуху.

● **OSDP**
OSDP (Open Supervised Device Protocol) — это стандарт связи для управления доступом, разработанный для улучшения совместимости в сфере электронных систем безопасности.

● **OSINT**
OSINT (разведка на основе открытых источников) — это раздел разведки, который анализирует информацию о людях или организациях из общедоступных источников.

Специалисты по кибербезопасности собирают информацию из открытых источников для: Оценки безопасности объекта и определения поверхности атаки для более эффективного противодействия угрозам; Выявления утечек данных; Выявления готовящихся угроз, их источников и векторов; Расследования и атрибуции киберинцидентов.

● **OTA (Over the Air) (Беспроводной принцип обновления данных)**
Метод обновления программного обеспечения, включающий отправку последних версий через Wi-Fi или мобильную сеть передачи данных. Ключевой особенностью технологии является централизованная рассылка установочного пакета широкому кругу получателей. OTA-обновление проще для пользователей, чем предыдущие технологии обновления программного обеспечения мобильных устройств, поскольку от них требуется только подтвердить установку.

● **Overlay / Topcoat (Защитное покрытие при печати карт)**
Покрытие/верхнее покрытие: прозрачная защитная панель, наносимая в процессе печати для защиты отпечатанных изображений от износа и выцветания.

● **[RUST] | Ownership (Владение)**

Владение - это самая уникальная особенность Rust, которая имеет глубокие последствия для всего языка. Это позволяет Rust обеспечивать безопасность памяти без использования сборщика мусора, поэтому важно понимать, как работает владение. В этой главе мы поговорим о владении, а также о нескольких связанных с ним возможностях: заимствовании, срезах и о том, как Rust раскладывает данные в памяти.

Определение: Владение — это ключевая концепция Rust, которая управляет тем, как память выделяется и освобождается. Каждое значение в Rust имеет "владельца" (owner), и когда владелец выходит из области видимости, память автоматически освобождается.

```
fn main() {
    let s = String::from("hello"); // s становится владельцем строки
    // Здесь s владеет памятью
} // s выходит из области видимости, память освобождается
```

Нюансы: Владение предотвращает ошибки вроде двойного освобождения памяти или использования указателей после освобождения. Передача владения (move) происходит при присваивании или передаче в функцию, если тип не реализует трейт `Copy`.
Связанные имена имеют одну особенность в Rust: они «владеют» тем, с чем они связаны. Это означает, что, когда имя выходит за пределы области видимости, ресурс, с которым оно связано, будет освобождён.

Например:

```
fn foo() {
    let v = vec! [1, 2, 3];
}
```

Когда v входит в область видимости, создаётся новый **Vec<T>**. В данном случае вектор также выделяет из кучи пространство для трёх элементов. Когда v выходит из области видимости в конце **foo(),** Rust очищает все, связанное с вектором, даже динамически выделенную память. Это происходит детерминировано, в конце области видимости.

● **PaaS (Platform as a Service) (Платформа как Сервис)**
Сервисная модель предоставления облачной платформы с определённым набором функций. PaaS-провайдер предоставляет клиенту полностью настроенную платформу (аппаратное обеспечение, ОС, СУБД), на которой может быть развернуто всё необходимое прикладное программное обеспечение. В отличие от модели IaaS, инфраструктурой управляет провайдер, а при передаче платформы на аутсорсинг точные характеристики обычно не критичны и могут не оговариваться. Более того, вычислительные мощности могут динамически меняться в зависимости от потребностей конкретного пользователя.

● **[RUST] Packages and Crates (Пакеты и Крейты)**
> Крейт — это наименьший объем кода, который компилятор Rust рассматривает за раз. Даже если вы запустите rustc вместо cargo и передадите один файл с исходным кодом (как мы уже делали в разделе «Написание и запуск программы на Rust» Главы 1), компилятор считает этот файл крейтом. Крейты могут содержать модули, и модули могут быть определены в других файлах, которые компилируются вместе с крейтом, как мы увидим в следующих разделах.Крейт может быть одним из двух видов: бинарный крейт или библиотечный крейт. *Бинарные крейты* — это программы, которые вы можете скомпилировать в исполняемые файлы, которые вы можете запускать, например программу командной строки или сервер. У каждого бинарного крейта должна быть функция с именем main, которая определяет, что происходит при запуске исполняемого файла. Все крейты, которые мы создали до сих пор, были бинарными крейтами.

> Библиотечные крейты не имеют функции main и не компилируются в исполняемый файл. Вместо этого они определяют функциональность, предназначенную для совместного использования другими проектами. Например, крейт rand, который мы использовали в Главе 2 обеспечивает функциональность, которая генерирует случайные числа. В большинстве случаев, когда Rustaceans говорят «крейт», они имеют в виду библиотечный крейт, и они используют «крейт» взаимозаменяемо с общей концепцией программирования «библиотека».

> Корневой модуль крейта — это исходный файл, из которого компилятор Rust начинает собирать корневой модуль вашего крейта (мы подробно объясним модули в разделе «Определение модулей для контроля видимости и закрытости»).

> Пакет — это набор из одного или нескольких крейтов, предоставляющий набор функциональности. Пакет содержит файл ***Cargo.toml***, в котором описывается, как собирать эти крейты. На самом деле Cargo — это пакет, содержащий бинарный крейт для инструмента командной строки, который вы использовали для создания своего кода. Пакет Cargo также содержит библиотечный крейт, от которого зависит бинарный крейт. Другие проекты тоже могут зависеть от библиотечного крейта Cargo, чтобы использовать ту же логику, что и инструмент командной строки Cargo.

Пакет может содержать сколько угодно бинарных крейтов, но не более одного библиотечного крейта. Пакет должен содержать хотя бы один крейт, библиотечный или бинарный. Давайте пройдёмся по тому, что происходит, когда мы создаём пакет. Сначала введём команду cargo new:

```
$ cargo new my-project
Created binary (application) `my-project` package
$ ls my-project
Cargo.toml
src
$ ls my-project/src
main.rs
```

После того, как мы запустили cargo new, мы используем **ls**, чтобы увидеть, что создал Cargo. В каталоге проекта есть файл *Cargo.toml*, дающий нам пакет. Также есть каталог *src*, содержащий *main.rs*. Откройте ***Cargo.toml*** в текстовом редакторе и обратите внимание, что в нём нет упоминаний о ***src/main.rs***. Cargo следует соглашению о том, что ***src/main.rs*** — это корневой модуль бинарного крейта с тем же именем, что и у пакета. Точно так же Cargo знает, что если каталог пакета

содержит ***src/lib.rs***, пакет содержит библиотечный крейт с тем же именем, что и пакет, а ***src/lib.rs*** является корневым модулем этого крейта. Cargo передаёт файлы корневого модуля крейта в rustc для сборки библиотечного или бинарного крейта.

Здесь у нас есть пакет, который содержит только ***src/main.rs***, что означает, что он содержит только бинарный крейт с именем my-project. Если пакет содержит ***src/main.rs*** и ***src/lib.rs***, он имеет два крейта: бинарный и библиотечный, оба с тем же именем, что и пакет. Пакет может иметь несколько бинарных крейтов, помещая их файлы в каталог ***src/bin***: каждый файл будет отдельным бинарным крейтом.

● **Packer (Упаковщик)**
Упаковщики используются для сжатия файлов. Хотя это может быть сделано и по вполне законным причинам, например, для экономии места на диске или сокращения времени передачи данных, киберпреступники также используют упаковщики для обфускации кода.

● **Parcer («Парсер»)**
Программа или сервис для поиска данных по определённым правилам. Парсер обрабатывает информацию по заданным критериям и выдаёт её в структурированном виде. Входными данными могут быть ключевая фраза или любая последовательность символов, а также характеристики объекта, такие как тип, размер файла или геолокация.
В отличие от API, парсинг не требует получения разрешения от владельца ресурса-донора на использование данных. В результате, контент, полученный с помощью парсера, часто сталкивается с юридическими проблемами. В сфере информационной безопасности парсеры используются для анализа логов и поиска следов активности вредоносных программ.

● **Parity Bit (Бит четности (правильнее: Бит Паритета))**
Бит чётности, или контрольный бит, добавляется к строке двоичного кода для обнаружения ошибок и проверки целостности данных. Значение бита чётности принимает значение 0 или 1 в зависимости от количества единиц в строке. Наиболее распространённый тип чётности требует, чтобы количество единиц в строке было чётным: если количество уже чётное, бит чётности устанавливается в 0; если же оно нечётное, бит чётности устанавливается в 1, чтобы сумма была чётной. Существует ещё один тип чётности, требующий нечётного количества единиц, но он встречается гораздо реже.

● **Passive Infrared Sensor (Пассивный инфракрасный датчик)**
Пассивные инфракрасные датчики используют естественное инфракрасное излучение, испускаемое всеми существами и предметами, для обнаружения присутствия. После установки датчик регистрирует постоянный уровень инфракрасного излучения окружающей среды. Когда человек или препятствие попадает в зону срабатывания, этот уровень изменяется, что приводит к срабатыванию реле датчика и выполнению действия.

● **Passive transponder (Пассивный транспондер)**
Транспондер, который собирает энергию, излучаемую электромагнитным полем считывателя, для питания чипа и передачи сигналов. Поскольку батарейка не требуется, такие транспондеры, как правило, не требуют обслуживания.

● **[RUST] Pattern Matching (Сопоставление с образцом) (match | if let)Определение:** Сопоставление с образцом — это мощный механизм для работы с данными через конструкции вроде match и if let.

```
fn main() {
    let number = Some(7);
    match number {
        Some(n) if n > 0 => println!("Positive: {}", n),
```

```
        Some(n) => println!("Non-positive: {}", n),
        None => println!("None"),
    }
}
```

Нюансы: match требует исчерпывающего покрытия всех вариантов, что делает код надёжным.

● **Peer-to-Peer (P2P)**
Термин «одноранговая сеть» может применяться к сетевой системе, в которой нет выделенного сетевого сервера и где каждая машина выполняет функции как сервера, так и клиента. Сегодня термин «P2P» чаще применяется к временному соединению, используемому пользователями, работающими с одним и тем же приложением, что позволяет им обмениваться файлами на компьютерах друг друга.

● **Pharming («Фарминг»)**
Вид кибератаки, направленный на скрытое перенаправление пользователей на фишинговый ресурс, принадлежащий злоумышленнику. Фарминг заключается в подмене IP-адреса легитимного сайта с помощью вредоносного ПО, установленного на компьютере жертвы. Перенаправление осуществляется путём изменения файла hosts или настроек DNS-сервера. Список доменов, IP-адреса которых подлежат подмене, определяется злоумышленником.

● **Phase (Фаза)**
Часть полного цикла формы волны, измеренная от указанной контрольной точки.

● **Phase Jitter Modulation (Модуляция фазового дрожания)**
Разновидность фазовой манипуляции, разработанная компанией Magellan Technology, работающая на частоте 13,56 МГц и соответствующая стандарту ISO/IEC 18000 3 Mode 2. Технология PJM обеспечивает скорость записи данных до 424 килобит в секунду и скорость чтения данных до 106 кбит/с. Она особенно подходит для маркировки отдельных изделий в фармацевтической промышленности.

● **Phishing (Фишинг)**
Фишинг (хакерская атака, связанная с фишингом) — это вид интернет-мошенничества, целью которого является кража конфиденциальной информации, например, данных учётных записей или банковских карт, с помощью социальной инженерии. Типичная фишинговая атака включает отправку электронных писем или сообщений от имени реальной организации с «приманкой» в теме или тексте сообщения и ссылкой на страницу, запрашивающую данные. Киберпреступники, осуществляющие такие атаки, называются фишерами.

● **Pig Butchering («Разделка свиньи» сленг мошенников)**
Разделка свинины — это вид мошенничества, связанный с фиктивными инвестициями, часто в криптовалюте. Его характерные черты: Длительное общение между мошенником и жертвой — порядка нескольких месяцев;К рупные суммы; жертвы могут потерять сотни тысяч, а то и миллионы долларов.

● **PoC (Proof of Concept)**
Демонстрация осуществимости конкретного метода. PoC используется для проверки теоретических расчётов и гипотез на практике.В области информационной безопасности PoC подразумевает моделирование работы программного обеспечения или эксплойта с целью определения оптимальных способов защиты или потенциальной угрозы компрометации компьютерной системы.

● **Polymorphism (Полиморфизм вредоносной программы – «Те же функции, другая байтовая последовательность кода»)**
Свойство вредоносной программы, при котором каждая её новая копия, созданная посредством механизма самораспространения, имеет уникальную последовательность байтов на уровне исполняемого файла, но с теми же функциями, что и родительская копия. Полиморфизм используется для обхода

антивирусной защиты с помощью сравнения сигнатур или эвристического анализа. В настоящее время киберпреступники редко используют его, переключившись на шифрование кода и обфускацию.

● **Port (Порт)**
Номер, присваиваемый пакету данных для определения процесса, для которого он предназначен. Порты, по сути, используются для идентификации служб, работающих на устройстве, и обеспечения корректной адресации передаваемой информации.

● **Port TCP/IP**
Идентификатор пакета, передаваемого по протоколу TCP/IP. Номер порта используется для маршрутизации данных внутри одного хоста и определяет процесс, которому необходимо доставить конкретный пакет. Для портов TCP/IP зарезервировано определённое адресное пространство из 65 536 номеров. Идентификаторы делятся на три группы: Публичные порты, используемые для системных процессов
> Пользовательские (статические)
> Динамические, назначаемые на время сеанса определённого процесса.

● **Portal (reader) (Портальный считыватель)**
RFID-считыватель, обычно используемый на производственных или складских площадках. Для транспортировки маркированных предметов через портальный считыватель для сбора данных RFID-транспондера используются вилочные погрузчики или другие средства.

● **PostgreSQL**
PostgreSQL — свободная объектно-реляционная система управления базами данных (СУБД).
Существует в реализациях для множества UNIX-подобных платформ, включая различные BSD-системы, IRIX, Linux, macOS, Solaris/OpenSolaris, Tru64, QNX, а также для Microsoft Windows.
PostgreSQL базируется на языке SQL и поддерживает многие из возможностей стандарта SQL:2011 и ряд возможностей SQL:2016 в части работы с данными в формате JSON.
В PostgreSQL есть следующие ограничения:

Максимальный размер базы данных	Нет ограничений
Максимальный размер таблицы	32 Тбайт
Максимальный размер поля	1 Гбайт
Максимум записей в таблице	Ограничено размерами таблицы
Максимум полей в записи	250-1600, в зависимости от типов полей
Максимум индексов в таблице	Нет ограничений

Сильными сторонами PostgreSQL считаются:

- высокопроизводительные и надёжные механизмы транзакций и репликации;
- расширяемая система встроенных языков программирования: в стандартной поставке поддерживаются PL/pgSQL, PL/Perl, PL/Python и PL/Tcl; дополнительно можно использовать PL/Java, PL/PHP, PL/Py, PL/R, PL/Ruby, PL/Scheme, PL/sh и PL/V8, а также имеется поддержка загрузки модулей расширения на языке C;
- наследование;
- возможность индексирования геометрических (в частности, географических) объектов и наличие базирующегося на ней расширения PostGIS;
- встроенная поддержка слабоструктурированных данных в формате JSON с возможностью их индексации;

- расширяемость (возможность создавать новые типы данных, типы индексов, языки программирования, модули расширения, подключать любые внешние источники данных).

● Precision laser cutting (Тонкая лазерная вырезка)
Одна из нескольких технологий, используемых для изготовления антенны RFID-транспондера. Антенна вырезается лазером из алюминия или других металлов, что позволяет избежать использования химикатов, обычно используемых при травлении антенн. Излишки алюминия утилизируются.

● PRNG (Псевдо-Случайный генератор чисел)
PRNG означает генератор псевдослучайных чисел (Pseudo-Random Number Generator). PRNG — это алгоритм, генерирующий последовательность чисел, имитирующую генерацию случайных чисел. Это не совсем случайная последовательность, поскольку PRNG начинается с начального значения, из которого генерируется остальная часть последовательности. В системах управления доступом PRNG используются в скользящем коде KeeLoq®.

● Proxy Server (Прокси Сервер)
Прокси-сервер находится между пользователями сети и Интернетом.Когда кто-то в сети запрашивает веб-страницу через свой браузер, запрос проходит через прокси-сервер. Прокси-сервер проверяет свой кэш, чтобы определить, была ли эта страница запрошена ранее:если да, то прокси-серверу не требуется доступ в Интернет, что обеспечивает более быстрый доступ к кэшированным страницам.

● PSW Троян (Password – stealing Trojans)
Эти трояны предназначены для кражи паролей с компьютера жертвы (хотя некоторые из них крадут и другие типы информации: IP-адрес, регистрационные данные, данные почтового клиента и т. д.). Эта информация затем отправляется на адрес электронной почты, закодированный в теле трояна. Первые трояны PSW появились в 1990-х годах и представляли собой трояны AOL, ворующие пароли.

● Q
В протоколе радиоинтерфейса Gen 2 Q — это параметр, который считыватель использует для регулирования вероятности срабатывания метки. Считыватель подаёт команду меткам в раунде инвентаризации выбрать случайное число от нуля до 2^Q; метка может быть успешно отделена, если ни одна другая метка не выберет то же самое случайное число («коллизия»). Большие значения Q снижают вероятность коллизии, но требуют от считывателя больше времени на отделение.

● Quantum cryptography (Квантовая криптография)
Квантовая криптография — это наука о шифровании данных, основанная на законах квантовой механики. В отличие от традиционной криптографии, методы квантового шифрования используют физические свойства элементарных частиц для защиты и передачи данных.

● Quiet Tag («Тихая метка»)
RFID-метка, которую можно считать только изредка, когда выходной сигнал считывателя находится на полной мощности, или которую можно считать только на очень близком расстоянии.

● Race Condition («Ошибка гонки»)
Ошибка в программировании сложных приложений, при которой процессы выполняются не последовательно в предопределённом порядке, а меняются местами в зависимости от определённых условий или даже работают параллельно. В результате может возникнуть уязвимость, которая проявляется только в условиях эксплуатации, при которых два или более процессов вынуждены конкурировать друг с другом. Такую ошибку сложно обнаружить, но злоумышленники могут её использовать. Концепцию состояния гонки специалисты также называют «неопределённостью параллелизма».

● **RAM – Random Access Memory (ОЗУ – Оперативное Запоминающее устройство)**
Память временного хранения. Информация, хранящаяся в RAM удаляется при отключении питания.

● **ROM – Read Only Memory (ПЗУ – Постоянное Запоминающее устройство)**
Это форма памяти постоянного хранения.

● **Ransomware (Программа вымогатель)**
Программы-вымогатели — это вредоносное ПО, которое шифрует данные или блокирует доступ к ним, требуя от пользователя оплаты за разблокировку или расшифровку данных. Различные виды вредоносных программ атакуют настольные компьютеры и мобильные устройства.

● **Reader (Считыватель)**
Устройство чтения и записи для беспроводного считывания данных с чипа транспондера (карты).

● **Read Accuracy (Точность считывания)**
Этот термин обычно относится к проценту успешно считанных меток. Если в поле 1000 меток (карт) и 985 из них считаны успешно, точность считывания составляет 98,5%.

● **Reading Range (Дистанция считывания)**
Представляет собой область считывания, которую может охватить считыватель. Метки, находящиеся за пределами области считывания, не могут быть активированы полем, генерируемым считывателем, и, следовательно, не могут быть идентифицированы и прочитаны.

● **Retransfer printing (Ретрансферная печать)**
Обратный перенос (ретрансфер/печать высокой четкости): принтер сначала печатает на пленке, которая затем термически закрепляется на поверхности карты, обеспечивая покрытие «через край» без видимых границ.

● **RFID (Radio-Frequency Identification)**
Технология RFID основана на считывании и обработке радиосигналов, полученных от специальных меток, прикреплённых к объектам. Такие RFID-метки, состоящие из антенны и микрочипа, называются транспондерами. Антенна принимает сигнал от RFID-считывателя, передаёт его на микрочип и преобразует полученные электромагнитные волны в энергию, которую тот использует для обработки запроса и передачи ответа. Активные RFID-метки со встроенным источником энергии могут использоваться для увеличения дальности считывания .RFID широко применяется в торговле, транспорте, системах обработки багажа в аэропортах и других областях.

● **RFID система (Radio-Frequency Identification System)**
Система RFID состоит из транспондера и считывателя. Когда транспондер находится в зоне действия считывателя, он передаёт данные на считыватель по радиоволнам.

● **RSSI (Received Signal Strength Indication)**
Измерение мощности принимаемого радиосигнала. В RFID RSSI используется для определения расстояния до транспондера, поскольку сигнал сильнее у транспондера, расположенного ближе к антенне считывателя.

● **RTLS (Real Time Locating System) (Система обнаружения в реальном времени)**

Система автоматического определения и отслеживания местоположения объектов или людей в режиме реального времени, обычно в пределах здания или другой замкнутой зоны. Беспроводные метки RTLS крепятся к объектам или носят люди, и в большинстве систем RTLS фиксированные опорные точки получают беспроводные сигналы от меток для определения их местоположения методом триангуляции. Примерами систем определения местоположения в режиме реального времени являются отслеживание автомобилей на сборочной линии, поиск поддонов с товарами на складе или поиск медицинского оборудования в больнице.

● **RSA (Асимметричный алгоритм шифрования)**
RSA — это алгоритм шифрования с открытым ключом (или асимметричный алгоритм), впервые публично описанный в 1977 году Роном Ривестом, Ади Шамиром и Леонардом Адлеманом в Массачусетском технологическом институте (MIT). Первые буквы их фамилий образуют название «RSA». Сегодня RSA широко используется для защиты данных, например, в протоколах SSH, SSL и TLS, а также в программах (например, браузерах), которым необходимо защищать данные, передаваемые через Интернет.

● **RS-485**
RS485 — промышленный стандарт последовательной связи. Он устанавливает физический электрический интерфейс, обеспечивающий связь между устройствами. RS485 является усовершенствованием по сравнению со своим предшественником (RS232), поддерживая подключение нескольких устройств к одной шине, а не только между устройствами, и позволяя использовать кабели на больших расстояниях.

● **RSSI (Received Signal Strength Indicator)**
RSSI — индикатор уровня принятого сигнала, который указывает интенсивность ответного сигнала на шлюз.

● **RTLS (Real Time Location System)**
RTLS означает Real-Time Location System (система определения местоположения в реальном времени) и обозначает те системы отслеживания и идентификации, которые позволяют определять местоположение маркированных активов.

● **RUST lang** — мультипарадигменный компилируемый язык программирования общего назначения, сочетающий парадигмы функционального и процедурного программирования с объектной системой, основанной на типажах. Управление памятью осуществляется через механизм «владения» с использованием аффинных типов[англ.][10], что позволяет обходиться без системы сборки мусора во время исполнения программы. Rust *гарантирует* безопасную работу с памятью благодаря встроенной в компилятор системе статической проверки ссылок (*borrow checker*). Имеются средства, позволяющие использовать приёмы объектно-ориентированного программирования[11].

Ключевые приоритеты языка: безопасность, скорость и параллелизм. Rust пригоден для системного программирования, в частности, он рассматривается как перспективный язык для разработки ядер операционных систем[10]. Rust сопоставим по скорости и возможностям с Си/C++, однако даёт большую безопасность при работе с памятью, что обеспечивается встроенными в язык механизмами контроля ссылок. Производительности программ на Rust способствует использование «абстракций с нулевой стоимостью».

После нескольких лет активной разработки первая стабильная версия (1.0) вышла 15 мая 2015 года, после чего новые версии выходят раз в 6 недель. Для версий языка, вышедших после 1.0, заявлена обратная совместимость.

Разрабатывается с 2010-х годов сообществом Mozilla Research и финансировался фондом Mozilla Foundation. С 2020 года планировалась передача интеллектуальной собственности и процессов развития и финансирования языка в организацию Rust Foundation[15]. 8 февраля 2021 года пять компаний-учредителей (AWS, Huawei, Google, Microsoft и Mozilla) официально объявили о создании Rust Foundation[16][17].

Девять лет подряд с 2016 по 2024 год Rust занимает первое место в списке самых любимых языков программирования («Most loved programming languages» / «Most admired programming languages») по версии ежегодного опроса разработчиков Stack Overflow Developer Survey

● **Secure Element (SE) (Тип чипа для задач безопасности)**
Это чип, изначально защищённый от несанкционированного доступа и используемый для запуска ограниченного набора приложений, а также для хранения конфиденциальных и криптографических данных. Secure Element используется в смартфонах и планшетах, аппаратных криптокошельках и других устройствах. Чип может хранить и обрабатывать такую информацию, как PIN-коды, пароли, отпечатки пальцев, платёжную информацию и многое другое. Безопасность Secure Element. Ограниченный доступ к чипу обеспечивает надёжную защиту Secure Element. Во-первых, на него нельзя установить никакие программы (всё его программное обеспечение предустановлено). Во-вторых, только доверенные приложения (например, цифровые кошельки) и устройства (например, POS-терминалы) имеют доступ к чтению и/или записи на чип. Secure Element также предназначен для противодействия многим известным атакам, в частности, атакам по сторонним каналам. Технология Secure Element обеспечивает следующие возможности на аппаратном уровне: Обнаружение попыток взлома и модификации; Создание платформы Root of Trust (RoT) для систем шифрования; Предоставление защищенной памяти для хранения закрытых ключей шифрования, данных банковских карт и другой информации; Криптографически безопасная генерация случайных чисел; Генерация ключей — например, пар закрытого и открытого ключей для асимметричного шифрования.

● **Salt (Программная соль)**
Последовательность данных, добавляемых к криптоключу для предотвращения перебора. При шифровании двух идентичных наборов данных их хеш-функции совпадают, что может быть использовано киберпреступниками для взлома криптографического алгоритма. Соль решает эту проблему, делая каждую последовательность уникальной. При шифровании коротких блоков информации (например, пароля) соль увеличивает длину исходной строки, что ещё больше усложняет декодирование ключа. Соль может быть статической или генерируемой динамически. Последняя более устойчива к взлому.

● **Samba**
Кроссплатформенная программа для управления сетевой инфраструктурой. Она обеспечивает общий доступ к файлам и принтерам, а также может выступать в качестве контроллера домена, включая поддержку Active Directory. Samba реализует общедоступную (бесплатную) версию сетевого протокола SMB, разработанного Microsoft. Программное обеспечение пользуется популярностью у системных администраторов благодаря гибкости настроек и возможности их настройки как через графический интерфейс, так и через консоль. Samba разработана командой энтузиастов и распространяется как бесплатное программное обеспечение.

● **Sandbox (Песочница)**
В контексте компьютерной безопасности «песочница» обеспечивает строго контролируемую среду, в которой полудоверенные программы или скрипты могут безопасно запускаться в памяти (или с ограниченным доступом к локальному жесткому диску).

● **SAW (Радиочастотный сигнал -> в Ультразвуковой акустический в рамках RFID)**
Технология автоматической идентификации, при которой маломощные микроволновые радиочастотные сигналы преобразуются в ультразвуковые акустические сигналы пьезоэлектрическим кристаллическим материалом в транспондере. Изменения отраженного сигнала могут быть использованы для обеспечения уникальной идентификации.

● **Semi-Active Tag (Полуактивный транспондер (Метка))**
Аналогично активным меткам, но батарея используется для питания микросхемы, а не для передачи сигнала считывателю. Некоторые полупассивные метки находятся в спящем режиме до тех пор, пока их не разбудит сигнал считывателя, что экономит заряд батареи. Стоимость полупассивных меток может составлять от доллара и выше. Такие метки иногда называют метками с батарейным питанием.

● **Seed Code (Начальный код/Начальное значение)**
В генераторе псевдослучайных чисел (PRNG) начальный код или начальное значение — это начальное значение, из которого генерируется остальная часть последовательности в соответствии с используемым алгоритмом.

● **SHA (Стандарты Криптохэширования данных)**
Набор криптографических алгоритмов для хеширования паролей, генерации цифровых подписей и других целей. Они основаны на создании уникального одностороннего хеша для каждого отдельного блока информации. Включает в себя неиспользуемый в настоящее время алгоритм SHA-0, устаревший SHA-1 и семейство криптографических методов SHA-2. Последний представляет собой новый стандарт для установления защищённого соединения и включает различные варианты, различающиеся длиной подписи в битах.

● **Shadow Card (Теневая карта)**
Теневая карта — это метод дублирования физических карт доступа для упрощения администрирования системы. Если обычная карта пользователя утеряна, украдена или не возвращена, её можно легко удалить из системы с помощью соответствующей теневой карты.

● **Shearlock (Срезной замок)**
Срезной замок (или срезной замок) — это тип электрического замка, который повышает надёжность стандартного магнитного замка со штифтами или ригелями, выступающими в пластину якоря. Штифты добавляют к электромагниту физическое запирание, так что если по какой-либо причине питание магнита отключается, штифты остаются на месте и обеспечивают безопасность двери.

● **Shielding (Экранирование)**
Использует принцип клетки Фарадея, металлизированного листа, фольги или металлического барьера для предотвращения помех радиоканалу связи между считывателем RFID и транспондером.

● **Silent Commerce (Тихая коммерция)**
Этот термин охватывает все бизнес-решения, основанные на маркировке, отслеживании, датчиках и других технологиях, включая RFID, которые делают повседневные предметы интеллектуальными и интерактивными. В сочетании с постоянным и всеобъемлющим подключением к Интернету они формируют новую инфраструктуру, позволяющую компаниям собирать данные и предоставлять услуги без участия человека.

● **Singulation (Сингуляция)**
Способ, с помощью которого RFID-считыватель идентифицирует метку с определённым серийным номером среди нескольких меток в своём поле. Существуют различные методы выделения, но наиболее распространённым является «обход дерева», при котором запрашивается ответ от всех меток с серийным номером, начинающимся с 1 или 0. Если запрашивается ответ от нескольких меток, считыватель может запросить ответ от всех меток с серийным номером, начинающимся с 01, а затем от 010. Он продолжает это делать, пока не найдёт искомую метку.

● **Skimming (Скимминг)**
Считывание RFID-метки у человека без его ведома или тайное считывание метки.

● **Slap Fingerprint (Групповой набор отпечатков пальцев)**
Групповой образ отпечатков нескольких пальцев: отпечатки пальцев, полученные при одновременном нажатии четырьмя пальцами одной руки на сканер или карту отпечатков пальцев. Пощечины известны как одновременные простые оттиски четырьмя пальцами.

● **Slotted Antenna (Щелевая Антенна)**
Антенна, состоящая только из узкой щели, прорезанной в электрическом проводнике, соединённом с транспондером. Щелевые антенны обладают такой же чувствительностью к ориентации, как и диполи.

● **Smart Label (Умная RFID метка / Этикетка)**
Умная этикетка содержит RFID-метку. Она называется умной, потому что может передавать информацию об объекте, на котором она нанесена.

● **Software as a Service (SaaS)**
Модель лицензирования программного обеспечения, при которой клиенту предоставляется удалённый доступ к приложению, чаще всего размещённому на облачных ресурсах поставщика услуг. В рамках SaaS покупатель не несёт расходов на поддержку приложения, а оплачивает регулярную абонентскую плату. Обычно доступ осуществляется через браузер или тонкий клиент; однако некоторые варианты позволяют установить приложение локально и проверять статус подписки при запуске.

● **Solenoid Bolt (Соленоидный ригель)**
Соленоидный ригель — это тип электрического замка. Запирающий механизм приводится в действие электромеханическим ригелем, который вставляется в пластину якоря, чтобы надёжно запереть дверь. Соленоидные ригели могут быть как отказоустойчивыми с открытием двери при перебоях питания (FailSAFE), так и отказоустойчивыми с закрытием двери при перебоях питания (FailSECURE).

● **Spoofing (Спуфинг)**
Атака, фальсифицирующая передаваемые данные. Целью спуфинга может быть получение расширенных привилегий; он основан на обходе механизмов проверки с помощью поддельных запросов, имитирующих настоящие. Один из способов такой подмены включает использование поддельного HTTP-заголовка для получения доступа к скрытому контенту. Спуфинг также может быть направлен на обман пользователя, классическим примером чего является подмена адресов отправителей в сообщениях электронной почты.

● **Spyware (Шпионский софт)**
Тип программного обеспечения, которое тайно устанавливается на компьютер пользователя для сбора его данных. В отличие от вредоносных программ, шпионское ПО не наносит вреда операционной системе, программам и файлам.

● **SQL**
Специализированный язык программирования для реляционных баз данных. SQL используется для описания структуры больших массивов информации, их изменения и быстрого извлечения данных. Запросы к базе данных являются основным инструментом SQL. Они описывают действия, которые необходимо выполнить с конкретными данными. Система управления базами данных (СУБД) обрабатывает запросы и возвращает результаты.

● **SSL (Secure Socket Layer)**
Технология безопасной передачи данных между веб-сервером и браузером. Чаще всего используется с протоколом передачи гипертекста: после получения SSL-сертификата протокол http меняется на https.

● **Stack Overflow (Ошибка переполнения Стэка)**
Ошибка в компьютерной программе, возникающая из-за попытки записать в стек вызовов больше данных, чем он может хранить. Обычно размер стека ограничен и устанавливается при запуске программы, и переполнение этого адресного пространства приводит к завершению работы приложения. Киберпреступники используют такую ошибку для выполнения кода на устройстве или проведения атаки типа «отказ в обслуживании».

● **Stack Smashing (Переполнение буфера Стэка)**
Кибератака, основанная на переполнении буфера стека, методе, используемом для выполнения вредоносного кода на устройстве. Злоумышленник перезаписывает переменные, указатели или адреса возврата, чтобы получить контроль над уязвимым приложением. Успешная атака позволяет запустить в системе сторонний скрипт с правами взломанного приложения.

● **Star Configuration (Принцип Звезды)**
В электронике и электропроводке звездообразная конфигурация или звездообразная сеть относится к сети, в которой все коммуникации между двумя компонентами должны проходить через центральную точку, концентратор или контроллер.

● **[RUST] Structures (Структуры) (struct …)**
Структуры похожи на кортежи, рассмотренные в разделе "Кортежи", так как оба хранят несколько связанных значений. Как и кортежи, части структур могут быть разных типов. В отличие от кортежей, в структуре необходимо именовать каждую часть данных для понимания смысла значений. Добавление этих имён обеспечивает большую гибкость структур по сравнению с кортежами: не нужно полагаться на порядок данных для указания значений экземпляра или доступа к ним.

Для определения структуры указывается ключевое слово `struct` и её название. Название должно описывать значение частей данных, сгруппированных вместе. Далее, в фигурных скобках для каждой новой части данных поочерёдно определяются имя части данных и её тип. Каждая пара `имя: тип` называется *полем*.

```
struct User {
    active: bool,
    username: String,
    email: String,
    sign_in_count: u64,
}
```

После определения структуры можно создавать её *экземпляр*, назначая определённое значение каждому полю с соответствующим типом данных. Чтобы создать экземпляр, мы указываем имя структуры, затем добавляем фигурные скобки и включаем в них пары `ключ: значение` (key: value), где ключами являются имена полей, а значениями являются данные, которые мы хотим сохранить в полях. Нет необходимости чётко следовать порядку объявления полей в описании структуры (но всё-таки желательно для удобства чтения). Другими словами, объявление структуры - это как шаблон нашего типа, в то время как экземпляр структуры использует этот шаблон, заполняя его определёнными данными, для создания значений нашего типа.

```
fn main() {
    let user1 = User {
        active: true,
        username: String::from("someusername123"),
        email: String::from("someone@example.com"),
        sign_in_count: 1,
    };
```

```
}
```

Чтобы получить конкретное значение из структуры, мы используем запись через точку. Например, чтобы получить доступ к адресу электронной почты этого пользователя, мы используем user1.email. Если экземпляр является изменяемым, мы можем поменять значение, используя точечную нотацию и присвоение к конкретному полю.

```
fn main() {
    let mut user1 = User {
        active: true,
        username: String::from("someusername123"),
        email: String::from("someone@example.com"),
        sign_in_count: 1,
    };

    user1.email = String::from("anotheremail@example.com");
}
```

Структуры (`struct`) — это один из способов создания более сложных типов данных. Например, если мы рассчитываем что-то с использованием координат 2D пространства, то нам понадобятся оба значения — `x` и `y`:

```
let origin_x = 0;
let origin_y = 0;
```

Структура позволяет нам объединить эти два значения в один тип с **`x`** и **`y`** в качестве имен полей:

```
struct Point {
    x: i32,
    y: i32,
}

fn main() {
    let origin = Point { x: 0, y: 0 }; // origin: Point

    println! ("Начало координат находится в ({}, {})", origin.x, origin.y);
}
```

Этот код делает много разных вещей, поэтому давайте разберём его по порядку. Мы объявляем структуру с помощью ключевого слова `struct`, за которым следует имя объявляемой структуры. Обычно, имена типов-структур начинаются с заглавной буквы и используют чередующийся регистр букв: название **PointInSpace** выглядит привычно, а **Point_In_Space** — нет.
Как всегда, мы можем создать экземпляр нашей структуры с помощью оператора `let`. Однако в данном случае мы используем синтаксис вида `ключ: значение` для установки значения каждого поля. Порядок инициализации полей не обязательно должен совпадать с порядком их объявления.
Наконец, поскольку у полей есть имена, мы можем получить к ним доступ с помощью операции `точка`: **origin.x.**
Значения, хранимые в структурах, неизменяемы по умолчанию. В этом плане они не отличаются от других именованных сущностей. Чтобы они стали изменяемы, используйте ключевое слово **mut:**

```
struct Point {
    x: i32,
    y: i32,
}

fn main() {
    let mut point = Point { x: 0, y: 0 };

    point.x = 5;

    println!("Точка находится в ({}, {})", point.x, point.y);
}
```

Этот код напечатает `Точка находится в (5, 0)`.
Rust не поддерживает изменяемость отдельных полей, поэтому вы не можете написать что-то вроде такого:

```
struct Point {
    mut x: i32,
    y: i32,
}
```

Изменяемость — это свойство имени, а не самой структуры. Если вы привыкли к управлению изменяемостью на уровне полей, сначала это может показаться непривычным, но на самом деле такое решение сильно упрощает вещи. Оно даже позволяет вам делать имена изменяемыми только на короткое время:

```
struct Point {
    x: i32,
    y: i32,
}

fn main() {
    let mut point = Point { x: 0, y: 0 };
```

```
    point.x = 5;

    let point = point; // это новое имя неизменяемо

    point.y = 6; // это вызывает ошибку
}
```

Структуры так же могут содержать &mut ссылки, это позволяет вам производить подобные преобразования:

```
struct Point {
    x: i32,
    y: i32,
}
struct PointRef<'a> {
    x: &'a mut i32,
    y: &'a mut i32,
}
fn main() {
    let mut point = Point { x: 0, y: 0 };
    {
        let r = PointRef { x: &mut point.x, y: &mut point.y };
        *r.x = 5;
        *r.y = 6;
    }
    assert_eq! (5, point.x);
    assert_eq! (6, point.y);
}
```

● **Sunstrate (Сабстрат)**
Материал-носитель, например пластиковая пленка, который является частью готового продукта (например, RFID-этикетка).

● **Symmetric encryption (Симметрическое шифрование)**
Симметричное шифрование — это метод шифрования данных, при котором для кодирования и декодирования информации используется один и тот же ключ. До появления первых асимметричных шифров в 1970-х годах это был единственный криптографический метод.
Как работают симметричные алгоритмы:
В общем случае, любой шифр, использующий один и тот же секретный ключ для шифрования и дешифрования, считается симметричным.
Например, если алгоритм заменяет буквы цифрами, отправитель сообщения и его получатель должны иметь одну и ту же таблицу соответствия. Первый шифрует сообщение с её помощью, а второй — расшифровывает.

● **Tailgating (Проводник-Нарушитель)**
В системе контроля доступа Tailgating (Проводник-Нарушитель) означает, что уполномоченное лицо получает доступ в зону ограниченного доступа, а затем за ним туда же следует другой человек, при этом дверь остаётся открытой после проверки учётных данных первого. Подмена понятий представляет собой проблему для систем безопасности по двум причинам: во-первых, она позволяет потенциально неавторизованным лицам получить доступ в зоны ограниченного доступа, а во-вторых, снижает точность отчётности и прозрачность системы.

● **Tag (Метка)**
Общее обозначение пассивного транспондера с корпусом, изготовленным по индивидуальному заказу. Большинство RFID-меток состоят как минимум из двух частей. Одна из них — интегральная схема (микрочип) для хранения и обработки информации, модуляции и демодуляции радиочастотного (РЧ) сигнала и других специализированных функций. Вторая — антенна для приёма и передачи сигнала.

● **Telnet (Незащищенный старый коммуникационный интерфейс)**
Сетевой текстовый протокол, предназначенный для установления терминального соединения между клиентской и серверной частями компьютерной системы.Telnet использует протокол TCP для передачи пакетов без шифрования трафика или других средств защиты. По соображениям безопасности использование этого стандарта разрешено только в сетях, полностью изолированных от внешних атак, поэтому он не применяется в современных операционных системах.

● **Tamper Evident Tag (TET)**
RFID-транспондер, который подает сигнал считывателю при открытии коробки, упаковки или контейнера.

● **Thermal Printhead (TPH) (Термохромная печатная головка)**
Термопечатающая головка: устройство в принтере, которое использует тепло для переноса чернил с ленты на карту, являющееся основой технологий прямой печати (DTC) и термопечати.

● **Thermal Transfer Printing (Термотрансферная печать)**
Термотрансферная печать: метод, при котором монохромная смола или цветная красящая лента нагреваются печатающей головкой и переносятся на поверхность карты. Термотрансферная печать использует чернила на основе твердой смолы или воска, которые подаются с ленты. Чернила расплавляются печатающей головкой и переносятся на поверхность карты в заданных точках. Этот метод позволяет получать четкий текст и штрихкоды, идеально подходящие для функциональных элементов, требующих высокой читаемости и четкости. Чернила наносятся на поверхность карты, создавая высокопрочные и устойчивые к царапинам изображения и текст. Пигментные термочернила, как правило, обладают изначальной устойчивостью к УФ-излучению, что способствует долговечности отпечатков.

● **TDMA (Time Division Multiple Access)**
Множественный доступ с временным разделением (TDMA): метод решения проблемы столкновения сигналов двух считывателей. Алгоритмы используются для обеспечения того, чтобы считыватели пытались считывать метки в разное время.

● **Template (Шаблон)**
Шаблон: цифровое представление отличительных характеристик человека, представляющее информацию, извлеченную из биометрического образца. Шаблоны используются при биометрической аутентификации в качестве основы для сравнения.

● **Threshold (Порог многосложности паттерна или кода паттерна в биометрии)**
Порог: настройка пользователя для биометрических систем, работающих в задачах проверки или открытой идентификации (список наблюдения). Принятие или

отклонение биометрических данных зависит от того, падает ли счет матча выше или ниже порогового значения. Порог регулируется так, что биометрическая система может быть более или менее строгой, в зависимости от требований любого конкретного биометрического приложения.

● **Token (Так и переводим – Токен и не путаем с «Ключами»)**
Средство идентификации пользователя или отдельного сеанса в компьютерных сетях и приложениях. Различают программные и аппаратные токены.
Программный токен обычно представляет собой зашифрованную последовательность символов, которая точно идентифицирует объект и определяет уровень привилегий. Он генерируется системой авторизации и привязывается к конкретному сеансу, сетевому клиенту или пакету данных.
Аппаратный токен — это устройство, на котором хранится уникальный пароль или который может быть сгенерирован в соответствии с определёнными правилами. Для аутентификации аппаратные токены могут быть физически подключены к компьютеру через коммуникационный порт или специальный считыватель.

● **TOR (The Onion Router) (Луковичный поисковик, страшная штука)**
Система специализированных серверов, обеспечивающая анонимное сетевое взаимодействие. Обычное соединение устанавливается напрямую между пользователем и сервером; TOR-соединение устанавливается через оверлейную сеть, состоящую из множества серверов, при этом данные шифруются на каждом сервере.

● **TLS (Transport Layer Security)**
Протокол для безопасной передачи данных в Интернете. Является развитием стандарта SSL и, по сути, дополнением к протоколу HTTP. Для создания безопасного соединения TLS использует симметричное и асимметричное шифрование данных, различные криптографические алгоритмы и сертификаты открытых ключей. TLS разработан и поддерживается IETF (Internet Engineering Task Force).

● **[RUST] Trait (Трейт) (trait | dyn trait)**
Определение: Трейт **(trait)** — это способ определения общего поведения для типов. Похож на интерфейсы в других языках, но с возможностью реализации по умолчанию.

```
trait Printable {
    fn print(&self) {
        println!("Default print");
    }
}
struct Item {
    name: String,
}
impl Printable for Item {
    fn print(&self) {
        println!("Item: {}", self.name);
    }
}

fn main() {
    let item = Item { name: String::from("Book") };
    item.print(); // Выведет: Item: Book
}
```

Нюансы: Трейты используются для обобщённого программирования (generics) и динамической диспетчеризации через dyn Trait.

● **Trailing Parity Bit (Бит Паритета)**
Бит чётности, или контрольный бит, добавляется к строке двоичного кода для обнаружения ошибок и проверки целостности данных. Значение бита чётности равно 0 или 1 в зависимости от количества единиц в строке. Наиболее распространённый тип чётности требует, чтобы количество единиц в строке было чётным: если количество уже чётное, бит чётности устанавливается в 0; если оно нечётное, бит чётности устанавливается в 1, чтобы сделать сумму чётной. Существует ещё один тип чётности, требующий нечётного количества единиц, но он встречается гораздо реже. Завершающий бит чётности размещается в конце двоичной строки.

● **Transponder (Транспондер)**
Транспондер — это носитель информации в виде вставки, этикетки или бирки. Это микроэлектронная схема, состоящая из микрочипа, соединённого с антенной на подложке-носителе.

● **Triple Pole (Триполь)**
В электрическом выключателе полюсом называется цепь, которой он управляет. Трёхполюсный выключатель может управлять тремя отдельными цепями, используя один и тот же выключатель.

● **Trojans (Трояны)**
Трояны — это вредоносные программы, которые выполняют действия, не разрешенные пользователем: удаляют, блокируют, изменяют или копируют данные, а также нарушают работу компьютеров и компьютерных сетей. В отличие от вирусов и червей, угрозы этой категории не способны копировать себя или самовоспроизводиться.

● **Type I Error (Ошибка Типа I)**
Ошибка типа I: ошибка, которая возникает в статистическом тесте, когда истинное утверждение (ошибочно) отклоняется. Например, Джон утверждает, что он Джон, но система ошибочно отклоняет это утверждение.

● **Type II Error (Ошибка Типа II)**
Ошибка типа II: ошибка, которая возникает в статистическом тесте, когда ложное утверждение (ошибочно) не отклоняется. Например: Фрэнк утверждает, что он Джон, и система проверяет его.

● **Type A Reference Interval (TARI)**
Длительность импульса энергии, передаваемого на метки UHF EPC Gen 2 для обозначения 0 в двоичном коде. Считыватели, совместимые с EPC Gen 2, используют кодирование импульсного интервала (PIE) для кодирования двоичных данных. Двоичный «0» обозначается коротким импульсом высокого уровня, за которым следует импульс низкого уровня такой же длительности. Длительность TARI может варьироваться от 6,25 до 25 микросекунд.

● **TypeScript** — язык программирования, представленный Microsoft в 2012 году и позиционируемый как инструмент, расширяющий возможности JavaScript. Разработчиком языка TypeScript является Андерс Хейлсберг, создатель ранее Turbo Pascal, Delphi и C#.

Спецификации языка открыты и опубликованы в соответствии с Соглашением о спецификациях Open Web Foundation (OWFa 1.0).
TypeScript обратно совместим с JavaScript и компилируется в последнюю очередь. Полученное после компиляции программы на TypeScript можно запускать в

любом браузере или использовать совместно с серверной платформой Node.js. Код экспериментального компилятора, транслирующего TypeScript в JavaScript, распространяется под лицензией Apache. Его идея ведется в публичном репозитории через сервис GitHub[10].

TypeScript отличается от JavaScript любого статического типа, поддержкой использования полноценных классов (как в традиционных объектно-ориентированных языках), а также поддержкой модулей подключения, что позволяет повысить скорость разработки, обеспечить читаемость, рефакторинг и повторное использование кода, помочь выявить ошибки поиска на этапах разработки и компиляции и, возможно, ускорить выполнение программ. Планируется, что полная обратная адаптация адаптации существующих приложений к новому языку программирования может перейти поэтапно, путем постепенного изменения типа.

На момент выпуска представлены файлы для восприятия расширенного синтаксиса TypeScript для Vim и Emacs, а также плагин для Microsoft Visual Studio. Следуя этим рекомендациям, разработчики составляют файлы с типовыми декларациями для некоторых популярных JavaScript-библиотек, в том числе jQuery.

● **UDP**
Один из транспортных протоколов, используемых для передачи данных в Интернете. UDP отправляет пакеты данных без проверки доступности получателя или целостности пакетов. Этот протокол менее надёжен, но быстрее, чем TCP. Он подходит для передачи больших объёмов данных в ситуациях, когда допустимы несколько потерянных пакетов и небольшое ухудшение качества, например, при потоковой передаче аудио и видео. UDP является базовым протоколом для многих приложений онлайн-игр.

● **UID (Unique Identification Code) (Как правило в системах 13,56MHz с возможностью хранения данных в памяти чипа, в 125KHz используется CSN Card Serial Number)**
Серийный номер, хранящийся на микрочипе, который однозначно идентифицирует транспондер.

● **Unicode**
Основной стандарт, используемый для кодирования символов в Интернете. Юникод определяет метод кодирования наборов символов (таких как алфавиты, знаки препинания, вспомогательные элементы) в шестнадцатеричном формате. Он поддерживает большое количество различных наборов символов, включая иероглифы и музыкальную нотацию.

● **Unsafe (Небезопасный код) [RUST] (unsafe)**
Определение: Блок unsafe позволяет обойти некоторые проверки безопасности Rust, например, для работы с сырыми указателями или вызова C-кода.

```rust
fn main() {
    let mut num = 5;
    let ptr = &mut num as *mut i32;
    unsafe {
        *ptr = 10; // Изменение через сырой указатель
    }
    println!("{}", num); // Выведет: 10
}
```

Нюансы: Использование unsafe требует осторожности, так как оно может привести к неопределённому поведению (UB).
Предупреждение: Используйте unsafe только там, где это действительно необходимо, и документируйте свои действия.

● **USB (Universal Serial Bus)**
USB обеспечивает стандарт «plug-and-play» для одновременного подключения множества периферийных устройств к компьютеру без необходимости использования отдельной платы-адаптера для каждого устройства.USB позволяет подключать к одному компьютеру до 127 устройств и обеспечивает быструю передачу данных.

● **Use-After-Free (Уязвимость освобождения памяти)**
Использование после освобождения (UAF) — уязвимость, связанная с некорректным использованием динамической памяти во время работы программы. Если после освобождения области памяти программа не очищает указатель на эту область, злоумышленник может использовать эту ошибку для взлома программы.

● **UTF-8**
Одна из самых распространённых кодировок, используемых в Интернете. UTF-8 — это, по сути, способ представления всех символов стандарта Unicode. Ключевой особенностью UTF-8 является её компактность: в отличие от первых таблиц Unicode, для представления одной кодовой точки требуется от 1 до 4 байтов. Наиболее часто используемые латинские буквы в этой кодировке занимают всего один байт.

● **VBS (Visual Basic Script)**
VBS — это язык сценариев, разработанный Microsoft.Как и JavaScript, он часто используется при разработке веб-страниц.Для решения определённых задач зачастую проще написать сценарий, чем использовать формальный язык программирования, такой как C или C++.Однако, как и в случае с формальной программой, VBS также можно использовать для создания вредоносного кода.Поскольку скрипт легко встраивается в HTML, киберпреступник может внедрить вредоносный скрипт в веб-страницу или в HTML-сообщение электронной почты:и когда кто-то посещает страницу или читает электронное письмо, скрипт запускается автоматически.

● **Vue.js** — JavaScript-фреймворк с открытым исходным кодом для создания пользовательских интерфейсов[6]. Легко интегрируется в проекты с использованием других JavaScript-библиотек. Может функционировать как веб-фреймворк для разработки одностраничных приложений в реактивном стиле.

Разработчики называют Vue.js прогрессивным и постепенно адаптируемым по сравнению с другими веб-фреймворками.

Это позволяет разработчику настроить структуру приложения в соответствии с собственными требованиями. Разработчики считают Vue.js более простым в освоении, чем AngularJS, поскольку API построен намного проще в освоении. В Vue.js можно использовать только знания JavaScript и HTML. Возможно применение Typescript. У **Vue.js** есть собственная официальная достаточно богатая документация на многих языках, выложенная на vuejs.org, которая может послужить примером в объяснении проектирования и разработки в браузере. В Vue.js реализуется шаблон MVVM, Vue.js предлагает возможность привязки данных на Javascript, так что вывод и ввод данных сопрягаются непосредственно с источником данных. Таким образом, режим *ручного* определения данных (например, через jQuery) из HTML-DOM *не* нужен. При этом нет необходимости в никаких дополнительных аннотациях, как в Knockout.js, объявленные в Vue-Element обычные переменные JavaScript включаются в качестве *реактивных* элементов.

Реактивность означает, что представление в модели MVC *изменяется* по мере изменения модели. В Vue разработчики просто привязывают представление к соответствующей модели, и Vue автоматически наблюдает за изменениями в модели и перерисовывает представление. Эта функция делает управление состоянием Vue довольно простым и интуитивно понятным.

Эффекты перехода

Vue предоставляет различные способы применения эффектов перехода при вставке, обновлении или удалении DOM. Включает следующие инструмент

- Автоматически применять CSS классы при переходах и анимации
- Вы можете работать со сторонними библиотеками анимации CSS, такими как Animate.css.
- Используйте JavaScript функции для перехвата перехода, чтобы напрямую управлять DOM
- Может использоваться в сочетании со сторонними библиотеками анимации JavaScript, такими как Velocity.js.

● **Watchlist (Список наблюдения)**
Список наблюдения: термин, который иногда называют открытой идентификацией, описывает одну из трех задач, которые выполняют биометрические системы. Отвечает на вопросы: есть ли этот человек в базе данных? Если да, то кто они? Биометрическая система определяет, совпадает ли биометрический шаблон человека с биометрическим шаблоном кого-то из списка наблюдения. Человек не претендует на личность, а в некоторых случаях вообще не взаимодействует с системой лично.

● **Watering hole («Водопой» методика)**
Стратегия целенаправленной атаки, при которой киберпреступники заражают веб-сайты, которые они считают плодородной почвой для потенциальных жертв, и ждут, когда вредоносное ПО попадёт на их компьютеры. Эта метафора из дикой природы подразумевает место, куда животные ходят на водопой, что делает их уязвимыми для хищников.

● **Web Shell (Веб шелл)**
Веб-шелл — это командная оболочка (программа или скрипт для управления устройством посредством команд), позволяющая удалённо управлять веб-сервером. Веб-шелл может использоваться для легитимных задач, но чаще всего применяется при кибератаках. Одним из самых известных и распространённых веб-шеллов является China Chopper.
Веб-шелл может быть написан на любом языке программирования, поддерживаемом целевым сервером. Обычно это распространённые языки, например, PHP.

● **[RUST] While (Циклы «Пока») (while … if)**

Цикл **while** — это ещё один вид конструкции цикла в Rust. Выглядит он так:

```
let mut x = 5; // mut x: i32
let mut done = false; // mut done: bool

while !done {
    x += x - 3;

    println!("{}", x);

    if x % 5 == 0 {
        done = true;
    }
}
```

Он применяется, если неизвестно, сколько раз нужно выполнить тело цикла, чтобы получить результат. При каждой итерации цикла проверяется условие, и если оно истинно, то запускается следующая итерация. Иначе цикл **while** завершается.
Если вам нужен бесконечный цикл, то можете сделать условие всегда истинным:

while true {

Однако, для такого случая в Rust имеется ключевое слово loop:

loop {

В Rust анализатор потока управления обрабатывает конструкцию **loop** иначе, чем **while true**, хотя для нас это одно и тоже. На данном этапе изучения Rust нам не важно знать в чем именно различие между этими конструкциями, но если вы хотите сделать бесконечный цикл, то используйте конструкцию loop. Компилятор сможет транслировать ваш код в более эффективный и безопасный машинный код.

● **[RUST] | While (Раннее прерывание цикла)**

Давайте ещё раз посмотрим на цикл **while**:

```
let mut x = 5;
let mut done = false;

while !done {
    x += x - 3;

    println! ("{}", x);

    if x % 5 == 0 {
        done = true;
    }
}
```

В этом примере в условии для выхода из цикла используется изменяемое имя **done** логического типа. В Rust имеются два ключевых слова, которые помогают работать с итерациями цикла: **break** и **continue**.
Мы можем переписать цикл с помощью **break**, чтобы избавиться от переменной **done**:

```
let mut x = 5;

loop {
    x += x - 3;

    println!("{}", x);

    if x % 5 == 0 { break; }
}
```

Теперь мы используем бесконечный цикл **loop** и **break** для выхода из цикла. Использование явного **return** также остановит выполнение цикла. **continue** похож на **break**, но вместо выхода из цикла переходит к следующей итерации. Следующий пример отобразит только нечётные числа:

```
for x in 0..10 {
    if x % 2 == 0 { continue; }

    println!("{}", x);
}
```

> Метки Циклов:
Когда у вас много вложенных циклов, вы можете захотеть указать, к какому именно циклу относится **break** или **continue**. Как и во многих других языках, по умолчанию эти операторы будут относиться к самому внутреннему циклу. Если вы хотите прервать внешний цикл, вы можете использовать метку. Так, этот код будет печатать на экране только когда и **x**, и **y** нечётны:

```
'outer: for x in 0..10 {
    'inner: for y in 0..10 {
        if x % 2 == 0 { continue 'outer; } // продолжает цикл по x
        if y % 2 == 0 { continue 'inner; } // продолжает цикл по y
        println! ("x: {}, y: {}", x, y);
    }
}
```

● **Wiegand**
Wiegand — это стандарт проводной связи, широко используемый в системах контроля доступа для обеспечения связи между считывателем и контроллером. Существует множество вариантов протокола Wiegand, наиболее распространённым из которых является 26-битный формат. 26 бит определяют способ организации фрагментов двоичного кода при их передаче через систему.

● **Wild List (Список вредоносных программ)**
Список WildList был создан в июле 1993 года исследователем антивирусной безопасности Джо Уэллсом и с тех пор ежемесячно публикуется организацией WildList (теперь частью ICSA Labs, которая, в свою очередь, входит в TrueSecure Corporation). Его цель — отслеживать вредоносные программы, распространяющиеся в реальном мире (в разделе часто задаваемых вопросов WildList WildList назван «мировым авторитетом в вопросах, каких вирусов пользователям действительно следует опасаться»). Однако в современном мире, где все взаимосвязано, и в связи с резким ростом числа образцов вредоносных программ в последние годы существует высокий риск распространения новых вредоносных программ до их официального включения в список. В результате различие между вредоносными программами, «in the Wild» и «Zoo» (последний термин используется для вредоносных программ, которые известны своим существованием), WildList стал несколько устаревшим средством измерения реальной угрозы.

● **Wet Inlay (Инлей с клеевым слоем)**
Влажная метка (WET) представляет собой изделие, состоящее из антенны с микрочипом на подложке из клейкой пленки. После вырубки она представляет собой простейшую RFID-метку, которую можно использовать как таковую или подвергать дальнейшей обработке.

● **WEB3**
Web3 — концепция следующего поколения Интернета, основанного на блокчейне, что позволяет добиваться децентрализации. Основное отличие Web3 от Web2.0 — децентрализация, когда пользователи сами владеют и контролируют различные интернет-ресурсы или приложения, а не обращаются к услугам гигантских монополий. Термин был изначально введён со-владельцем Ethereum Гэвином Вудом в 2014 году, и начиная с 2021 года идея становилась всё более популярной среди энтузиастов в сфере криптовалюты, крупных технологических корпораций и фирм венчурного капитала. Впервые концепты того, как должна работать новая версия, были представлены в 2013 году. Критики, в числе которых энтузиаст в сфере криптовалюты и миллиардер Джек Дорси, отмечают, что, несмотря на заявленную децентрализацию и независимость от крупных технологических компаний, пользователи по факту не будут владеть Интернетом, вместо них это будут делать крупные фирмы венчурного капитала.

● **WSQ Wavelet Scalar Quantization**
WSQ - вейвлет-скалярное квантование: стандартный алгоритм сжатия, который используется для обмена отпечатками пальцев в сообществе уголовного правосудия. Он используется для уменьшения размера данных изображений.

● **ZigBee (Протокол связи в радиочастотных системах, включая RFID)**
Спецификация для набора высокоуровневых протоколов связи с использованием маломощных цифровых радиоустройств на основе стандарта IEEE 802.15.4 для беспроводных персональных сетей (WPAN). ZigBee предназначен для радиочастотных приложений, требующих низкой скорости передачи данных, длительного времени автономной работы и безопасного сетевого взаимодействия.

Оператор	Пример	Объяснение	Перегружаемость
!	ident!(...), ident!{...}, ident![...]	Вызов макроса	
!	!expr	Побитовое или логическое отрицание	Not
!=	expr != expr	Сравнение "не равно"	PartialEq
%	expr % expr	Остаток от деления	Rem
%=	var %= expr	Остаток от деления и присваивание	RemAssign
&	&expr, &mut expr	Заимствование	
&	&type, &mut type, &'a type, &'a mut type	Указывает что данный тип заимствуется	
&	expr & expr	Побитовое И	BitAnd
&=	var &= expr	Побитовое И и присваивание	BitAndAssign
&&	expr && expr	Логическое И	
*	expr * expr	Арифметическое умножение	Mul
*=	var *= expr	Арифметическое умножение и присваивание	MulAssign
*	*expr	Разыменование ссылки	Deref
*	*const type, *mut type	Указывает, что данный тип является сырым указателем	
+	trait + trait, 'a + trait	Соединение ограничений типа	
+	expr + expr	Арифметическое сложение	Add
+=	var += expr	Арифметическое сложение и присваивание	AddAssign
,	expr, expr	Разделитель аргументов и элементов	
-	- expr	Арифметическое отрицание	Neg
-	expr - expr	Арифметическое вычитание	Sub
-=	var -= expr	Арифметическое вычитание и присваивание	SubAssign
->	fn(...) -> type, |...| -> type	...	
.	expr.ident	Доступ к элементу	
..	.., expr.., ..expr, expr..expr	Указывает на диапазон чисел, исключая правый	PartialOrd
..=	..=expr, expr..=expr	Указывает на диапазон чисел, включая правый	PartialOrd
..	..expr	Синтаксис обновления структуры	
..	variant(x, ..), struct_type { x, .. }	Привязка «И все остальное»	
...	expr...expr	(Устарело, используйте новый синтаксис ..=) Используется при определении инклюзивного диапазона	

Оператор	Пример	Объяснение	Перегружаемость
/	expr / expr	Арифметическое деление	Div
/=	var /= expr	Арифметическое деление и присваивание	DivAssign
:	pat: type, ident: type	Ограничения типов	
:	ident: expr	Инициализация поля структуры	
:	'a: loop {...}	Метка цикла	
;	expr;	Признак конца инструкции и элемента	
;	[...; len]	Часть синтаксиса массива фиксированного размера	
<<	expr << expr	Битовый сдвиг влево	Shl
<<=	var <<= expr	Битовый сдвиг влево и присваивание	ShlAssign
<	expr < expr	Сравнение "меньше чем"	PartialOrd
<=	expr <= expr	Сравнение "меньше или равно"	PartialOrd
=	var = expr, ident = type	Присваивание/эквивалентность	
==	expr == expr	Сравнение "равно"	PartialEq
=>	pat => expr	Часть синтаксиса конструкции match	
>	expr > expr	Сравнение "больше чем"	PartialOrd
>=	expr >= expr	Сравнение "больше или равно"	PartialOrd
>>	expr >> expr	Битовый сдвиг вправо	Shr
>>=	var >>= expr	Битовый сдвиг вправо и присваивание	ShrAssign
@	ident @ pat	Pattern binding	
^	expr ^ expr	Побитовое исключающее ИЛИ	BitXor
^=	var ^= expr	Побитовое исключающее ИЛИ и присваивание	BitXorAssign
		pat	pat	Альтернативные шаблоны	
		expr	expr	Побитовое ИЛИ	BitOr
|=	var |= expr	Побитовое ИЛИ и присваивание	BitOrAssign
||	expr || expr	Короткое логическое ИЛИ	
?	expr?	Возврат ошибки	

Таблица Б-3. Синтаксис, связанный с путями

Обозначение	Объяснение
ident::ident	Путь к пространству имён
::path	Путь относительно корня крейта (т. е. явный абсолютный путь)
self::path	Путь относительно текущего модуля (т. е. явный относительный путь).
super::path	Путь относительно родительского модуля текущего модуля
type::ident, <type as trait>::ident	Ассоциированные константы, функции и типы
<type>::....	Ассоциированный элемент для типа, который не может быть назван прямо (например <&T>::...., <[T]>::...., etc.)
trait::method(...)	Устранение неоднозначности вызова метода путём именования типажа, который определяет его
type::method(...)	Устранение неоднозначности путём вызова метода через имя типа, для которого он определён
<type as trait>::method(...)	Устранение неоднозначности вызова метода путём именования типажа и типа

Таблица Б-7: Комментарии

Обозначение	Объяснение
//	Однострочный комментарий
//!	Внутренний однострочный комментарий документации
///	Внешний однострочный комментарий документации
/*...*/	Многострочный комментарий
/*!...*/	Внутренний многострочный комментарий документации
/**...*/	Внешний многострочный комментарий документации
_____	

Таблица Б-8: Кортежи

Обозначение	Объяснение
()	Пустой кортеж, он же пустой тип. И литерал и тип.

Обозначение	Объяснение
(expr)	Выражение в скобках
(expr,)	Кортеж с одним элементом выражения
(type,)	Кортеж с одним элементом типа
(expr, ...)	Выражение кортежа
(type, ...)	Тип кортежа
(type, ...)	Выражение вызова функции; также используется для инициализации структур-кортежей и вариантов-кортежей перечисления
expr.0, expr.1, etc.	Взятие элемента по индексу в кортеже

Таблица Б-10: Квадратные скобки

Контекст	Объяснение
[...]	Литерал массива
[expr; len]	Литерал массива, содержащий len копий expr
[type; len]	Массив, содержащий len экземпляров типа type
expr[expr]	Взятие по индексу в коллекции. Возможна перегрузка (Index, IndexMut)
expr[..], expr[a..], expr[..b], expr[a..b]	Взятие среза коллекции по индексу, используется Range, RangeFrom, RangeTo, или RangeFull как "индекс"

Таблица Б-9: Фигурные скобки

Контекст	Объяснение
`{...}`	Выражение блока
`Type {...}`	`struct` литерал

Таблица Б-3. Синтаксис, связанный с путями

Обозначение	Объяснение
ident::ident	Путь к пространству имён
::path	Путь относительно корня крейта (т. е. явный абсолютный путь)
self::path	Путь относительно текущего модуля (т. е. явный относительный путь).

Обозначение	Объяснение
super::path	Путь относительно родительского модуля текущего модуля
type::ident, <type as trait>::ident	Ассоциированные константы, функции и типы
<type>::...	Ассоциированный элемент для типа, который не может быть назван прямо (например <&T>::..., <[T]>::..., etc.)
trait::method(...)	Устранение неоднозначности вызова метода путём именования типажа, который определяет его
type::method(...)	Устранение неоднозначности путём вызова метода через имя типа, для которого он определён
<type as trait>::method(...)	Устранение неоднозначности вызова метода путём именования типажа и типа

Таблица Б-4: Обобщения

Обозначение	Объяснение
path<...>	Определяет параметры для обобщённых параметров в типе (e.g., Vec<u8>)
path::<...>, method::<...>	Определяет параметры для обобщённых параметров, функций, или методов в выражении. Часто называют turbofish (например "42".parse::<i32>())
fn ident<...> ...	Определение обобщённой функции
struct ident<...> ...	Определение обобщённой структуры
enum ident<...> ...	Объявление обобщённого перечисления
impl<...> ...	Определение обобщённой реализации
for<...> type	Высокоуровневое связывание времени жизни
type<ident=type>	Обобщённый тип где один или более ассоциированных типов имеют определённое присваивание (например Iterator<Item=T>)

Обозначения не-операторы

Следующий список содержит все символы, которые не работают как операторы; то есть они не ведут себя как вызов функции или метода. Таблица Б-2 показывает символы, которые появляются сами по себе и допустимы в различных местах.

Таблица Б-2: Автономный синтаксис

Обозначение	Объяснение
'ident	Именованное время жизни или метка цикла

Обозначение	Объяснение
...u8, ...i32, ...f64, ...usize, etc.	Числовой литерал определённого типа
"..."	Строковый литерал
r"...", r#"..."#, r##"..."##, etc.	Необработанный строковый литерал, в котором не обрабатываются escape-символы
b"..."	Строковый литерал байтов; создаёт массив байтов вместо строки
br"...", br#"..."#, br##"..."##, etc.	Необработанный строковый байтовый литерал, комбинация необработанного и байтового литерала
'...'	Символьный литерал
b'...'	ASCII байтовый литерал
|...| expr	Замыкание
!	Всегда пустой тип для расходящихся функций
_	«Игнорируемое» связывание шаблонов; также используется для читабельности целочисленных литералов