**Дополнительный материал к лекции по криптографии**

Для того, чтобы образовать систему криптографии в Java Android необходимы основные блоки, которые базируются на принципах JCA.

*Providers*

Существует абстрактный класс интерфейса провайдера для каждого класса криптографического механизма, который представляет конкретный криптографический сервис.

*SecureRandom*

Представляет собой криптографический генератор случайных чисел.

Имеет открытые конструкторы, которые используют для создания экземпляра. Если использовать getInstance (), то нужно передать SHA1PRNG в качестве имени алгоритма. Поскольку SHA1PRNG не является в точности криптографическим стандартом, реализации от разных поставщиков могут вести себя по-разному

Чтобы SecureRandom генерировал случайные байты, передаем байтовый массив методу nextBytes (). Он сгенерирует столько байтов, сколько длина массива и сохранит их в нем.

SecureRandom sr = new SecureRandom();u

byte[] output = new byte[16];

sr.nextBytes(output);

*MessageDiges*

Представляет функциональность хеш-функции.

Экземпляр создается путем передачи имени алгоритма хеширования в метод getInstance(). Входные данные могут быть предоставлены в блоках, используя один из методов update (), а затем вызывая один из методов digest (), чтобы получить вычисленное значение хеша.

MessageDigest md = MessageDigest.getInstance("SHA-256");

byte[] data = getMessage();

byte[] digest = md.digest(data);

*Signature*

Предоставляет общий интерфейс для алгоритмов цифровой подписи, основанных на асимметричном шифровании.

Экземпляры подписи создаются стандартным методом фабрики getInstance (). После получения экземпляра объект Signature инициализируется либо для подписания, путем передачи закрытого ключа в метод initSign (), либо для проверки, путем передачи открытого ключа или сертификата в метод x initVerify () для проверки. Затем данные, которые должны быть подписаны, передаются порциями в один из методов update () или в массе в метод sign (), который возвращает значение подписи. Для проверки подписи подписанные данные передаются одному из методов update (). Подпись передается методу valid (), который возвращает true, если подпись действительна.

*Cipher*

Предоставляет общий интерфейс для операций шифрования и дешифрования.

Объекты Cipher создаются с помощью метода getInstance (), который требует не только простое имя алгоритма, но и указания криптографического преобразование, которое будет выполнять запрошенный шифр.

После того, как экземпляр Cipher получен, его необходимо инициализировать перед шифрованием или дешифрованием данных. Шифр инициализируется передачей целочисленной константы, обозначающей режим работы.

Затем мы можем зашифровать данные, передав фрагменты данных методу update (), который возвращает промежуточные результаты, и получить последний блок, вызвав doFinal (). Окончательный зашифрованный текст получается путем объединения промежуточных результатов с последним блоком. Чтобы расшифровать, мы инициализируем шифр в DECRYPT\_MODE, передавая тот же ключ и IV, используемый для шифрования. Затем мы вызываем update () и вызываем doFinal().