1. **Описания алгоритмов шифрования.**

Шифрование — обратимое преобразование информации в целях сокрытия от [неавторизованных](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) лиц, с предоставлением, в это же время, [авторизованным](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) пользователям доступа к ней.

Главным образом, шифрование служит задачей соблюдения [конфиденциальности](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%84%D0%B8%D0%B4%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C) передаваемой информации. Важной особенностью любого алгоритма шифрования является использование [ключа](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D1%8E%D1%87_(%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F)), который утверждает выбор конкретного преобразования из совокупности возможных для данного алгоритма**.**

1. **Особенности алгоритмов шифрования «симметричный и ассиметричный ключ шифрования»**

**Ключ шифрования**— это секретная информация, используемая криптографическим алгоритмом при зашифровании/расшифровании сообщений, постановке и проверке цифровой подписи, вычислении кодов аутентичности. Количество информации в ключе измеряется в битах: чем больше битов в ключе, тем надежней зашифрована информация.

Ключи шифрования бывают симметричными и ассиметричными.

**ассиметричный шифр -** используется два ключа: один для шифрования, открытый (он передается по открытому каналу), другой для расшифровки (так называемый, закрытый ключ).

**Симметричный шифр** - обычно строится на основании ряда блоков с математическими функциями преобразования, ассиметричный — на математических задачах. Тот же RSA создан на задаче возведения в степень с последующим вычислением модуля. В результате алгоритмы симметричного шифрования модифицировать просто, а ассиметричного — практически невозможно

**Программное использование:**

Области применения у обоих типов шифрования разные, поэтому при выборе нужно соотносить возможности алгоритмов с поставленными задачами. Может быть, использовать гибридные решения.

Следует учитывать и практическую целесообразность. Допустим, если ваша программа предназначена только для персонального пользования, вряд ли будет рациональным использовать ассиметричный подход. Лучший вариант здесь — мощный симметричный алгоритм.