

Задача № 3

Применение потоковых шифров

Потоковые шифры (stream ciphers) основаны на генераторах псевдослучайных последовательностей. Псевдослучайные последовательности, применяемые в шифрах, часто называются *ключевыми потоками* (keystreams). Ключевой поток должен быть неотличим от случайного процесса, порождающего равновероятные и независимые символы (биты). При этом ключевой поток является функцией секретного ключа K и инициализирующего вектора IV . Предполагается, что ключ не меняется в течение длительного времени, а IV должен быть разным для разных сообщений (не допускается, чтобы при шифровании различных данных использовались одинаковые ключевые потоки). IV не обязан быть секретным. Шифрование данных производится путём сложения по модулю 2 бит данных с битами ключевого потока. Расшифрование производится фактически путём повторного шифрования с тем же ключевым потоком.

Имеются авторские программы потоковых шифров – финалистов конкурса eSTREAM: HC-128, Rabbit, Salsa20 и Sosemanuk. Все они имеют одинаковый программный интерфейс. Основные элементы интерфейса следующие:

ECRYPT_ctx

Структура, описывающая внутренне состояние генератора. В программе необходимо создать экземпляр этой структуры.

void ECRYPT_init (void)

Функция инициализации. Должна вызываться в самом начале.

void ECRYPT_keysetup (ECRYPT_ctx* ctx, const u8* key,
u32 keysize, u32 ivsize)

Установка в структуру ctx секретного ключа key. Типы u8 и u32 – это соответственно байт и 32-битное слово, рассматриваемые как числа без знака. Параметры keysize и ivsize – размеры ключа и инициализирующего вектора в *битах*.

void ECRYPT_ivsetup (ECRYPT_ctx* ctx, const u8* iv)

Установка в структуру ctx инициализирующего вектора iv.

Эти три функции должны вызываться строго последовательно. Функция ECRYPT_ivsetup может вызываться многократно, если секретный ключ не меняется.

void ECRYPT_encrypt_bytes (ECRYPT_ctx* ctx, const u8* plaintext,
u8* ciphertext, u32 msglen)

Функция зашифрования (расшифрования при повторном вызове). ctx – текущее состояние генератора; plaintext – исходный текст (зашифрованный текст при расшифровании); ciphertext – зашифрованный текст (восстановленный текст при расшифровании); msglen – длина текста в *байтах*.

Для каждой программы в конце соответствующего файла *.c дописана функция main в качестве примера использования. Её нужно удалить, если потоковый шифр интегрируется в другое приложение. Что касается примера, то он реализует программу, которая при запуске из командной строки принимает два аргумента: ключ и инициализирующий вектор в виде слов из не более чем 16 символов (проверки на переполнение нет, если

Перспективные технологии защиты информации

аргументы не заданы, то они полагаются равными нулю). Производится шифрование массива, заполненного нулями, и результат выдаётся на `stdout`. Фактически получается псевдослучайная последовательность, которая генерируется алгоритмом.

Задание

1. Для всех четырёх шифров сравнить длительность функций инициализации, и время генерации ключевого потока.
2. Выбрать один понравившийся потоковый шифр и реализовать на его основе приложение для передачи файла в зашифрованном виде по сети.