DOI 10.37882/2223-2966.2024.03.38

АНАЛИЗ ОГРАНИЧЕНИЙ ПРИ СИММЕТРИЧНОМ И АССИМЕТРИЧНОМ ШИФРОВАНИИ ДАННЫХ

ANALYSIS OF LIMITATIONS IN SYMMETRIC AND ASYMMETRIC DATA ENCRYPTION

A. Shurygin
A. Brysin
Yu. Zhuravleva
D. Potapova

Summary. The article analyzes the existing encryption algorithms. Using the example of the Magma and Grasshopper encryption algorithms, the possibility of implementing an encryptor based on Russian cryptographic standards is considered. A simplified block diagram of data encryption and decryption has been developed.

Keywords: encryption, decryption, virus, protection.

Шурыгин Андрей Михайлович
МИРЭА — Российский технологический университет
Брысин Андрей Николаевич
Кандидат технических наук, доцент,
МИРЭА — Российский технологический университет
brysin@rambler.ru
Журавлева Юлия Алексеевна
Кандидат технических наук, доцент,
МИРЭА — Российский технологический университет
Потапова Дарья Александровна
Преподаватель,
МИРЭА — Российский технологический университет

Аннотация. В статье проведен анализ существующих алгоритмов шифрования. На примере алгоритмов шифрования Магма и Кузнечик рассмотрена возможность реализации шифровальщика на базе российских криптографических стандартов. Разработана упрощенная блок-схема шифрования и дешифования данных.

Ключевые слова: шифрование, дешифрование, вирус, защита.

Шифровальщики относятся к роду вредоносных программ, которые производят несанкционированное шифрование файлов в системе, используя секретный ключ, известный только злоумышленнику. Согласно существующей классификации ФСТЭК, эта угроза имеет код УБИ.170.

Эксплуатация этой уязвимости является следствием недостаточной антивирусной защиты, «слабой» конфигурации системы или механизмов разграничения доступа [5]. При успешном заражении, основная система сохранит свою работоспособность, но доступность и целостность данных, а также внутренних систем будет нарушена. Если в коде шифровальщика нет ошибок, то при сотрудничестве с операторами вредоносного ПО, данные можно восстановить в исходное состояние (в большинстве случаев после уплаты внушительного выкупа). Помимо измененных файлов, в некоторых случаях может быть нарушена и конфиденциальность данных, если вымогатели перед шифрованием файлов, скачали их на свой сервер для дальнейшего шантажа скомпрометированной организации [6].

Согласно проведенному анализу, данный вид угроз стал актуален начиная примерно с 2020 года. С учетом этого и в условиях отказа от Windows с переходом на Astra Linux, проблема обеспечения безопасности этой Unix-подобной операционной системы становится очень актуальна. Рассмотрим наиболее известные зарубежные алгоритмы шифрования

Разработка алгоритма DES (Data Encryption standard) началась с объявленного в 1973 году конкурса на создание шифра. Изначально компания IBM представила на конкурс шифр Lucifer, но он не удовлетворил требованиям комиссии, после чего был доработан до DES и опубликован в 1975 году. Алгоритм основан на сети Фейстеля с 16 раундами и 56-битным ключом.

Разработанный в 1990 году частной швейцарской компанией Ascom, шифр IDEA (International Data Encryption Algorithm) задумывался в качестве в качестве замены DES.

Шифр CAMELLIA был разработан несколькими крупными японскими компаниями и впервые представлен 10 марта 2000 года. Стал одним из финалистов европейского конкурса по определению безопасных шифров NESSIE.

Алгоритм шифрования, пришедший на смену DES по причине устаревания первого — AES (Advanced Encryption Standard, aka Rijndael). Разработан в 1993 году двумя криптографами (Винсентом Рейманом и Еханом Даменом) из Левенского католического университета (Бельгия);

Поточный шифр RC4 (Rivest cipher 4 или Ron's code) на самом деле требует лицензии для использования. Был разработан сотрудником компании RSA Security в 1987 году и до 1994 года был коммерческой тайной.

Чтобы обойти обязательное лицензирование, использовались названия ARC4 или ARCFOUR.

Алгоритм шифрования SOSEMANUK разработан французскими криптографами в 2004 году. Опубликован в 2008 году после того, как стал одним из финалистов конкурса eStream. Как и SOSEMANUK, Salsa участвовал в конкурсе eStream, в котором стал победителем в области поточных шифров. Был разработан американским криптографом и математиком (Даниэлем Бернштейном) в 2005 году и опубликован в 2007. Модификация Salsa — ChaCha разработана и опубликован в 2008 году им же.

Из соображений обеспечения безопасности ключа дешифрования было бы удобнее использовать асимметричный алгоритм шифрования, поскольку в таком алгоритме для зашифрования и расшифрования используется два разных ключа. В таком случае открытый ключ ничего не даст тому, кто будет пытаться расшифровать файлы самостоятельно. Однако такие алгоритмы существенно медленнее симметричных и пригодны только для шифрования небольших случайных последовательностей символов (например ключей), поскольку в иных случаях шифр поддается криптоанализу. Например, чтобы обеспечить достаточное шифрование открытого текста длиной 128 бит, рекомендуется использовать ключ длиной 1024 или даже 2048 бит. Поскольку размер файлов чаще всего превышает 128 бит, ключ для надежного шифрования таких файлов должен быть в разы больше, что добавляет огромные ресурсозатраты для вычисления шифротекста.

Симметричные алгоритмы шифрования существенно быстрее и производительнее, что критично в случае шифрования большого объема файлов (в том числе, возможно, и файлов большого размера). В отличие от асимметричных алгоритмов, в данном случае для расшифрования и зашифрования используется один и тот же ключ, а значит в случае попадания файла с ключом в руки реверсера, велика вероятность расшифровывания файлов.

Логично будет использовать симметричные алгоритмы, поскольку асимметричные как минимум неприменимы для шифрования чего-либо кроме небольших ключей, составляющих случайный набор данных. Помимо этого, симметричные алгоритмы быстрее и производительнее асимметричных.

Проведем сравнение известных алгоритмов «Магма» и «Кузнечик», описанных в ГОСТ [1], между собой и двумя популярными за рубежом алгоритмами симметричного шифрования — AES (Advanced Encryption Standard) и DES (Data Encryption Standard). Результат сравнения представлен в таблице 1.1:

Таблица 1.1.

Сравнение алгоритмов шифрования

Алгоритм	Длина ключа (бит)	Размер блока (бит)	Число раундов
Кузнечик	256	128	10
Магма	256	64	32
AES	128, 192, 256	128	10–14
DES	56	16	64

На основе проведённого анализа можно утверждать, что при одинаковой длине ключей у «Кузнечика» и «Магмы», у второго алгоритма размер блока существенно меньше, а соответственно и безопасность шифрования будет значительно ниже, чем у первого. Однако даже при большей длине блока, «Кузнечик» значительно превосходит более старый алгоритм «Магма» по скорости шифрования за счет использования более простых и эффективных алгоритмов. Современные алгоритмы, используемые в реализации, дают «Кузнечику» выигрыш и в скорости, и в надежности по сравнению с «Магмой».

При сравнении AES и DES, первичным критерием считаем длину ключа — у DES она равна всего 56 битам (против 256 бит у AES), а это очень плохо сказывается на криптостойкости алгоритма, даже с учетом 64 раундов. Такая длина ключа допускает взлом шифра методом перебора всех возможных вариантов, которых по причине 56-битного ключа не очень много.

По итогам сравнения алгоритмов, наиболее перспективными вариантами считаю «Кузнечик» и AES. Согласно исследованию, проведенному в МГТУ им. Н.Э. Баумана в 2022 году [2], AES и «Кузнечик» почти не различаются по производительности. Учитывая тенденцию блокировки иностранных технологий в РФ, будет более предпочтительно выбрать отечественный «Кузнечик» как основной алгоритм шифрования, используемого в имитации вируса-шифровальщика.

Для проведения анализа работы шифровальщика, была составлен имитатор атаки по упрощенная блок-схема работы этой вредоносной программы, представленная на Рисунках 1 и 2.

Алгоритм работы шифровальщика начинается с того, что «жертва» запускает исполняемый файл вируса. Шифровальщик в самом начале процесса обращается к серверу оператора и отправляет в запросе уникальный идентификатор «жертвы». При получении такого запроса сервер генерирует ключ для симметричного шифрования, сохраняет его в базу данных в соответствии с идентификатором «жертвы», после чего отвечает на полученный запрос, отправив этот ключ в теле ответа. Таким образом ключ симметричного шифрования будет сохранен только в базе данных оператора, а в памяти компьютера «жертвы» будет существовать лишь на время шифрования. Поскольку симметричный ключ генери-

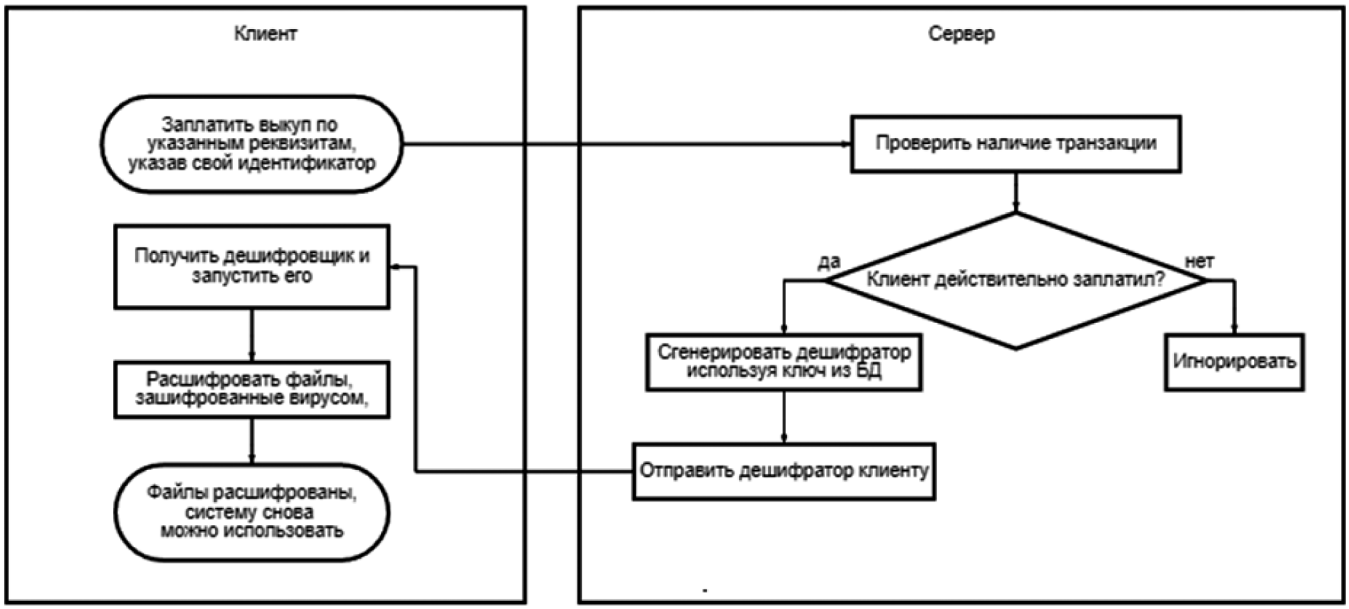

Рис. 1. Упрощенная схема алгоритма шифрования в вирусе

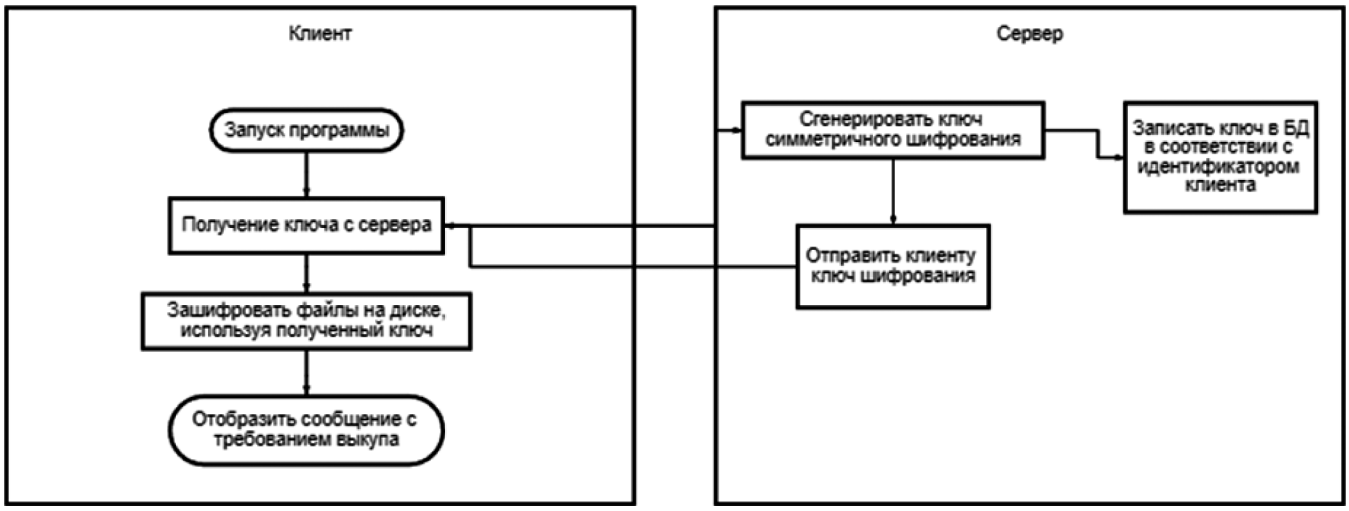

Рис. 2. Упрощенная схема алгоритма дешифрования в вирусе

руется прямо в процессе работы программы, его получение при помощи вскрытия исполняемого файла не будет возможно, что сильно усложнит попытки расшифровать зашифрованную инфраструктуру без участия операторов вируса. В конце процесса шифрования следует продемонстрировать сообщение с требованиями, которые «жертве» нужно выполнить для восстановления зашифрованной инфраструктуры. В общем случае это требование выкупа, пример которого можно увидеть на рисунке 3. В требованиях важно убедить «жертву» в том, что единственный, кто может помочь ему — это оператор вируса, а попытки прибегнуть к помощи сторонних организаций лишь безвозвратно повредят файлы.

Алгоритм дешифрования начинается с обращения «жертвы» к оператору вируса. Когда условия согласованы и выкуп уплачен, оператор генерирует соответствующий дешифратор, используя ключ шифрования, хранящийся в его базе данных. «Жертва», получив дешифратор, запускает его. Дешифратор, используя встроенный в себя ключ, точно также проходит по файловой системе, только вместо шифрования, он расшифровывает файлы, которые затронуты вирусом. После работы дешифратора все файлы будут восстановлены и с ними можно будет работать.

Выводы

Следует признать что в настоящее время альтернативы алгоритму блочного шифрования кузнечик пока не существует. В российской федерации отсутствуют уникальные решения по ассиметричным ключам и в соответствии с ГОСТ Р 34.11-2012 регламентируются только тип хеш-функции (алгоритм Стриборг) Стойкость ассиметричного алгоритма шифрования основывается на проблеме дискретного логарифмирования в группе точек

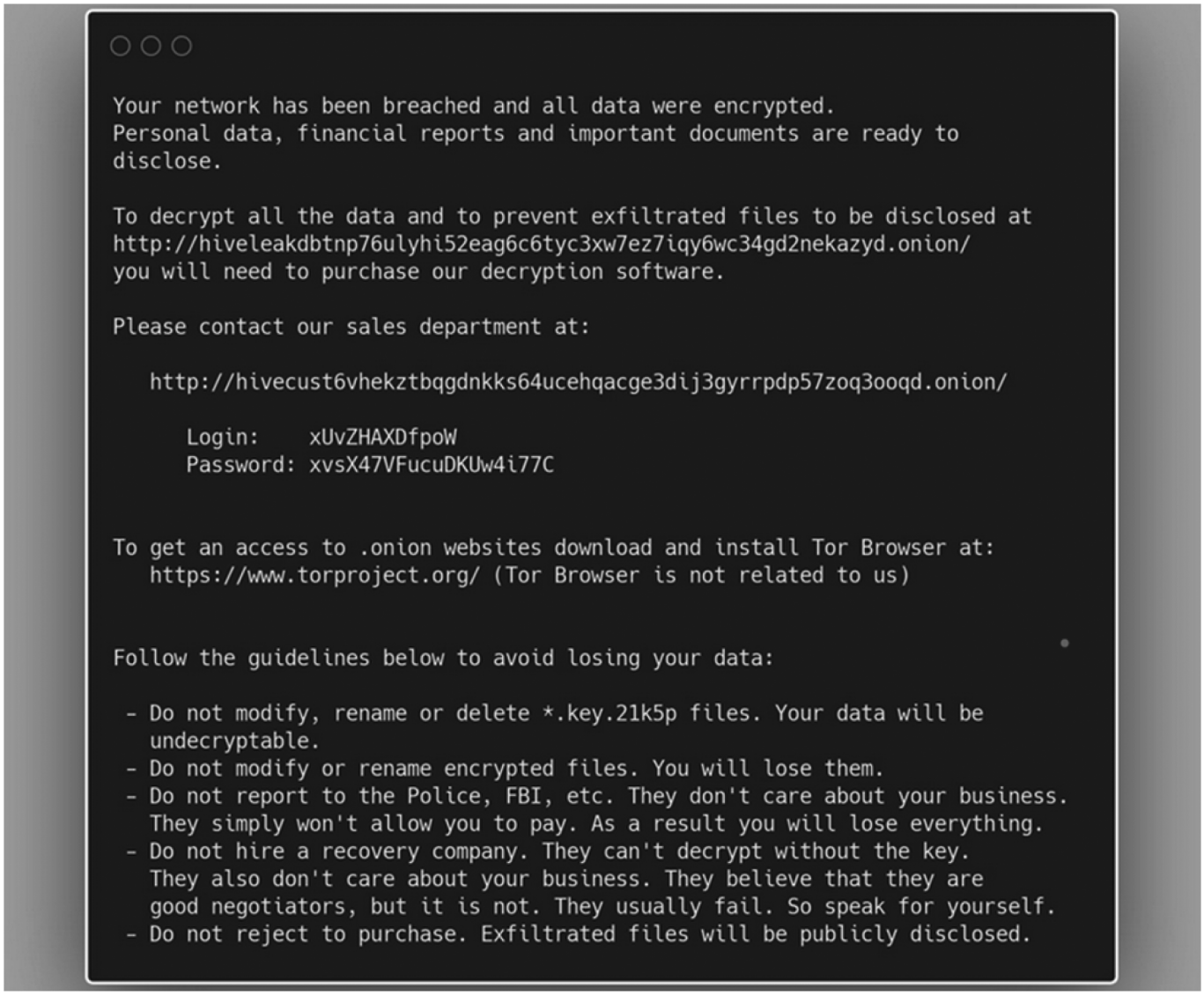

Рис. 3. Пример сообщения, оставленного шифровальщиком Hive

эллиптической кривой. На данный момент нет метода решения данной проблемы. Выходом является объединение удобства асимметричного алгоритма с надежностью и скоростью симметричного. При использовании гибридного шифрования, симметричный алгоритм со случайным ключом будет использован для шифрования любого объема данных, а асимметричный уже будет использован для защищенного хранения ключа.

ЛИТЕРАТУРА

1. ГОСТ Р 34.10-2015. Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи, IDT.
2. Соболев М.А. Сравнительный анализ российского стандарта шифрования по ГОСТ Р 34.12–2015 и американского стандарта шифрования AES. // Политехнический молодежный журнал. 2022 №04 (69) // [Электронный ресурс] // URL: http://ptsj.ru/catalog/icec/insec/785.html (Дата обращения: 22.10.2023).
3. Prashant Tilekar. VMware ESXi Servers: A Major Attack Vector for Ransomware // Онлайн-портал ForeScrout: [Электронный ресурс] // URL: https://www.forescout.com/blog/vmware-esxi-servers-a-major-attack-vector-for-ransomware/ (Дата обращения: 22.10.2023).
4. Lawrence Abrams. LockBit ransomware builder leaked online by «angry developer» // Онлайн-журнал BleepingComputer // [Электронный ресурс] // URL: https://www.bleepingcomputer.com/news/security/lockbit-ransomware-builder-leaked-online-by-angry-developer/ (Дата обращения: 22.10.2023).
5. LockBit ransomware — What You Need to Know // Онлайн-портал Kaspersky // [Электронный ресурс] // URL: https://www.kaspersky.com/resource-center/threats/lockbit-ransomware (Дата обращения: 22.10.2023).
6. Erebus Linux Ransomware: Impact to Servers and Countermeasures // Отчет исследовательской лаборатории TrendMicro // [Электронный ресурс] // URL: https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/erebus-linux-ransomware-impact-to-servers-and-countermeasures (Дата обращения: 22.10.2023).