Муниципальное бюджетное образовательное учреждение
Лицей №180
Ленинского района г. Нижнего Новгорода

Научное общество учащихся

# Шифр Вернама.

Выполнил: Важдаев Александр Сергеевич,
ученик 10Б класса
Научный руководитель: Сухова М. А.,
учитель информатики
высшей квалификационной категории

Нижний Новгород
2014

**Содержание**

*«Даже когда приземлятся пришельцы из созвездия Андромеды на своих огромных кораблях с компьютерами неслыханной вычислительной мощности, они не смогут прочитать сообщения советских агентов, зашифрованные на одноразовом блокноте (если только они не будут способны также обратить время вспять и вернуть этот блокнот)»*

*Б. Шнайер. «Прикладная криптография»*

## Введение

XXI век — век компьютеров и информации. С появлением Интернета появилась возможность передать сообщение на другой конец Земли за доли секунды. Появилось огромное множество средств, позволяющих передавать информацию. Самые различные IM-протоколы, электронная почта, социальные сети — можно быстро и просто общаться со своим другом из Африки или Австралии. И обычно в общении друзей нет ничего страшного — они вряд ли будут обсуждать план захвата мира. Но вдруг будут?

И тут возникает проблема — надёжность современных средств шифрования.

Вообще, историю криптографии можно считать равной по возрасту истории существования письменности, потому что именно тогда возникла потребность хранить информацию в виде, доступном только для определённых лиц. Естественно, что люди, от которых информация скрывалась, искали способы расшифровать закодированные сообщения. Таких людей сейчас называют криптоаналитиками. Обе стороны находятся в постоянном противоборстве: первые постоянно придумывают новые шифры, вторые ищут (и находят!) способы дешифровки сообщений.

Один из самых популярных сейчас шифров — AES. Он активно расхваливается как самый надёжный шифр, который АНБ использует для шифрования информации, составляющей государственную тайну, вплоть до уровня TOP SECRET. Но ещё в 1991 г. в сенат США был внесён законопроект, который требовал, чтобы любое американское криптографическое оборудование содержало лазейку, известную АНБ. Да что там, даже компания AT&T — один из крупнейших разработчиков криптографического оборудования в США нисколько не скрывает того факта, что АНБ знает лазейки ко всем тем шифрам, которые считаются невероятно надёжными. И никакие пароли длиной в тысячу символов здесь не помогут. Так что же делать? Неужели на тайне связи можно смело ставить крест и смириться с тем, что правительство будет читать нашу переписку?

Вовсе нет. Есть один способ. И о нём — моя работа.

## Шифр Вернама

Существует ли в криптографии абсолютно надёжный шифр, который невозможно взломать даже на самых мощных компьютерах за сотни тысяч лет? Да, существует. Называется он шифр Вернама, также известный как «схема одноразовых блокнотов». Шифр, несмотря на свою надёжность, очень прост, и может использоваться как на компьютере, так и при помощи обычной бумаги с ручкой.

Как он работает? Отправитель и получатель заранее составляют так называемый шифровальный блокнот. Шифровальным блокнотом может быть любая последовательность случайных чисел (байтов). Одинаковые шифровальные блокноты должны быть как у отправителя, так и у получателя. Когда нужно зашифровать сообщение, отправитель делает следующее. Он берёт первый байт своего сообщения и первый байт ключа (шифроблокнота). Для этих байтов он применяет операцию XOR («исключающее ИЛИ», сложение по модулю 2) и получает первый зашифрованный байт. Затем отправитель берёт второй байт сообщения и второй байт ключа, и аналогично получает второй байт шифровки. Операция повторяется для каждого байта сообщения. После шифрования использованный ключ должен быть уничтожен. Затем зашифрованное сообщение пересылается получателю по любому каналу связи. Получатель, получив шифровку, производит те же самые действия для расшифровки, после чего также уничтожает использованный ключ.

Для шифрования на бумаге используется не операция XOR, а сложение букв по модулю кодовой таблицы, содержащей обычно буквы, цифры и знаки препинания. Для расшифровки из шифротекста аналогично вычитаются символы ключа.

Правило использования шифра Вернама, без которого он не будет не то что абсолютно стойким, но вообще не сможет считаться полноценным шифром: *ключ должен использоваться только один раз, после чего уничтожаться.*

Если правило соблюдено, шифр невозможно взломать. Вообще. Немногочисленные случаи расшифровки текстов, закодированных шифром Вернама, связаны исключительно с нарушениями правил использования. Например, если ключ не был истинно случайным, или же, когда один и тот же ключ по каким-то причинам использовался дважды. В этом случае к шифротексту легко применяются методы взлома шифра Виженера, который по сути и есть шифр Вернама, но с зацикленным коротким ключом.

## История шифра

Шифр был изобретён в начале XX века… нет, не великим криптографом, и даже не гениальным учёным, а самым обычным инженером-телеграфистом, сотрудником AT&T Гилбертом Вернамом. Как и всё гениальное, изобретение крайне просто, и было сделано совершенно случайно.

В 10-х годах XX века телеграф стал общедоступным средством связи во всём цивилизованном мире, однако оставался лишённым конфиденциальности.

Аппараты, стоящие на концах линии, могли «слышать» друг друга посредством распространяющегося с огромной скоростью тока, но точно так их мог «слушать» любой, кому удалось подключиться к линии.

Инженер Вернам подошёл к задаче обеспечения конфиденциальности сообщений точно так же, как подходит к ней любой неискушённый программист: «добавив» к каждому символу символ ключа с перфоленты при шифровании и «отняв» его при расшифровке. В качестве «сложения» (оно же «вычитание») им была использована побитная операция XOR.

Это была первая система шифрования, которая позволила сделать шифрование автоматическим. Ведь раньше сообщения вручную шифровались перед отправкой и расшифровывались после получения.

После этого Вернам начинает исследования с целью определить требования к ключу. Первое требование было найдено почти сразу: случайное распределение символов в ключевой последовательности. Пока ключ случаен (то есть непредсказуем, в частности, не является осмысленной фразой), любой шифровке с равной вероятностью соответствует любой открытый текст той же длины. Однако это остаётся справедливым, лишь пока ключ не короче сообщения. Как только мы зашифруем с помощью одного и того же ключа два сообщения, расшифровка становится возможной даже вручную.

В 1918 году коллега Вернама инженер Лаймэн Морхауз предложил использовать для шифрования "вторичный" ключ, генерируемый с помощью двух первичных разной длины также побитным XOR. Два трехметровых кольцевых ключа длиной 1000 и 999 символов дали бы неповторяющуюся последовательность в 999000 символов. К этому времени Секция секретности уже тесно сотрудничала с заинтересовавшимися военными, одним из которых был Джозеф Моборн. Моборну удалось найти способ успешной атаки на шифр Морхауза, что было нетривиальным результатом (много позднее, став главой сигнальной разведки США, Моборн неоднократно подтверждал свою криптоаналитическую квалификацию).

Результат, к которому в конце концов пришли Вернам с Моборном, чисто негативен: то, что называется шифром Вернама, есть, по сути дела, всего лишь отказ от изощрённых механизмов повторения ключа, подобных предложенному Мархаузом. Совместные исследования Вернама и Моборна привели к дополнению принципа случайности ключа принципом бесконечности его длины. На практике это означает превышение длиной ключа суммы длин всех шифруемых с его помощью сообщений.

Вот и всё. Дальше — история «триумфального шествия» шифра Вернама. Постепенно он начал применяться везде, где требовалась сверхсекретная коммуникация. Первыми его взяли на вооружение правительства Германии и СССР. Немцы ввели в практику заранее подготовленные ключевые блокноты с отрывными листами, которые, по использовании, можно было сразу уничтожить. Отсюда — второе название шифра Вернама: «одноразовый ключевой блокнот».

Однако, гениальный алгоритм Вернама так и остался бы незамеченным, если бы не один человек. Этот человек — известнейший криптограф, который и доказал надёжность шифра Вернама. Этот человек — Клод Шеннон.

Спустя четыре десятка лет после пионерских разработок Вернама им были опубликованы две знаменитые работы, итожащие исследования времён Второй мировой войны: «Математическая теория связи» и «Коммуникационная теория секретных систем». В «Математической теории…» Шеннон ввёл количественную меру информации как величины, обратной энтропии. В «Коммуникационной теории…» ему удалось также ввести количественную меру секретности.

Совершенной секретностью, по Шеннону, обладает система, в которой апостериорные (то есть после перехвата шифровки) вероятности сообщений равны априорным их вероятностям (то есть до перехвата шифровки). При этом противника Шеннон определяет, как не ограниченного временем и ресурсами.

Шеннон доказал теорему (в «Теории связи…» она имеет номер 6), согласно которой необходимым и достаточным условием совершенной секретности является равенство полной вероятности всех ключей, продуцирующих шифровку E из открытого сообщения $M_i$, полной вероятности всех ключей, продуцирующих ту же самую шифровку из другого открытого сообщения $M_j$ для любых E, i, j. Далее, Шеннон показал, что шифр Вернама соответствует этому условию.

## Криптостойкость шифра Вернама

Шифр Вернама является не только абсолютно стойким, но ещё и единственным таким шифром. С точки зрения криптографии, невозможно придумать систему безопаснее шифра Вернама.

Итак, докажем абсолютную криптостойкость шифра Вернама.

Пусть дано двоичное сообщение m длины N. Распределение вероятности сообщений может быть любым. Ключ k также является двоичной последовательностью, но с равномерным распределением $P_k(k)=1/2^N$.

Зашифруем сообщение и получим шифротекст c:

$$C = M \oplus K = (m_0 \oplus k_0, m_1 \oplus k_1, \dots, m_{N-1} \oplus k_{N-1})$$

Найдём вероятностное распределение N-блоков шифротекстов:

$$P(c = a) = P(m \oplus k = \text{a}) = \sum_m P(m)P(m \oplus k = a|m) = \sum_m P(m)P(m)\frac{1}{2^N} = \frac{1}{2^N}$$

Результат подтверждает, что сумма двух случайных величин, одна из которых имеет равномерное распределение, является случайной величиной с равномерным распределением. Распределение шифротекстов равномерное.

Запишем совместное распределение открытых текстов и шифротекстов:

$$P(m = a, c = b) = P(m = a)P(c = b|m = a)$$

Найдём условное распределение:

$$P(c = b|m = a) = P(m \oplus k = b|m = a) = P(k = b \oplus a|m = a) = P(k = b \oplus a) = \frac{1}{2^N}$$

Ключ и сообщение являются независимыми случайными величинами. Итого:

$$P(c = b|m = a) = \frac{1}{2^N}$$

Подставляем в формулу совместного распределения:

$$P(m = a, c = b) = P(m = a)\frac{1}{2^N}$$

Что доказывает независимость сообщений и шифротекстов. Это и означает абсолютную криптографическую стойкость.

## Недостатки шифра Вернама

О достоинствах шифра сказано достаточно — абсолютная надёжность и невероятная простота. Теперь поговорим о недостатках. А они есть, ведь нет ничего идеального. Но именно эти недостатки (даже, скорее, не недостатки, а требования к использованию шифра) и создают ту самую абсолютную криптостойкость шифра.

- Для работы шифра необходима истинно случайная последовательность, которая и будет ключом. По определению, последовательность, полученная с помощью какого-либо алгоритма, является всего лишь псевдослучайной, и совершенно не подходит. Поэтому без источника случайности шифр работать не будет.
- Полная избыточность шифра. Длина ключа должна быть равна длине сообщения. Нужно заранее запастись ключом достаточного размера.
- Из предыдущего недостатка вытекает следующий — ограниченность длины ключа. Если ключ закончился, связь между собеседниками прерывается, пока не появится возможность передать новый ключ.
- Невозможно проверить правильность дешифровки. Получатель не может быть точно уверен в том, что он верно расшифровал сообщение. Если же злоумышленник каким-либо образом узнает сообщение, он сможет восстановить ключ и подменить сообщение на другое.
- Необходимо надёжно уничтожать ключ после использования. Особенно остро эта проблема стоит при использовании шифра на компьютере, где есть множество программ для восстановления удалённой информации с носителей.
- Шифр теряет криптостойкость при любом нарушении вышеперечисленных требований к использованию шифра. Именно из-за повторного использования блокнотов американцы смогли расшифровать сообщения советской разведки (проект «Venona»)

Несмотря на все перечисленные выше недостатки, в настоящее время одноразовые блокноты активно используются для шифрования сверхсекретных сообщений. Не имея в своём распоряжении соответствующего блокнота, эти сообщения невозможно прочитать вне зависимости от того, насколько быстро работают суперкомпьютеры, которые используются в ходе криптоаналитической атаки. Даже инопланетные пришельцы со своими сверхсветовыми звездолётами и чудо-компьютерами не смогут их прочесть, если, конечно, не вернутся в прошлое на машине времени и не добудут одноразовые блокноты, использованные для шифрования этих сообщений.

## Устранение недостатков

Конечно, можно просто смириться с недостатками и радоваться хотя бы тому, что шифр абсолютно надёжен, да ещё и не прост, а очень прост. Но не всем нравится жить с недостатками. Тем более, когда можно (и нужно!) бороться с ними. Итак, методы борьбы с недостатками предлагаются следующие:

- Проблема случайности. Можно собирать случайную информацию, используя радиоактивный распад, дробовой шум и космическое излучение. Но это слишком сложно и неоправданно дорого. Гораздо проще вежливо попросить создать случайные числа для шифра того, кому они нужны — то есть, человека. Он является неплохим источником случайности. А главное — просто, дёшево, и легко достать.
- Проблема избыточности. Можно просто сжать ключ архиватором. Хоть и не любой ключ можно эффективно сжать (он ведь случайный).
- Проблема прерывания связи. Новый ключ можно сжать, зашифровать и передать, используя остатки текущего ключа. Таким образом, можно передать больше информации, чем осталось байтов ключа.
- Проблема проверки подлинности. Дописать в конец сообщения контрольную сумму и зашифровать её вместе с сообщением. При дешифровке проверить.
- Проблема уничтожения. Бумажные страницы можно просто сжечь — надёжней некуда, а электронный ключ сотню раз затереть случайными данными.

Как видно, нет ничего невозможного. Но всё равно, есть один недостаток. Именно он часто является главным препятствием в использовании шифра. Это передача ключа. Прежде чем использовать шифр Вернама для связи, нужно как-то обеспечить собеседника ключом. Если возможна личная встреча, то можно передать его лично, если же нет — ничего не получится. Впрочем, в этом случае ничего не выйдет и с любым другим шифром.

Ну и ещё один недостаток, который я и решал (и довольно неплохо решил) на протяжении всего времени работы. Отсутствие удобных (да и вообще хоть каких-нибудь) средств, которые позволили бы воспользоваться этим чудесным шифром простым смертным.

Итак, представляю новое криптографическое средство, основанное на самой надёжной в мире системе шифрования — SANIO WearNum.

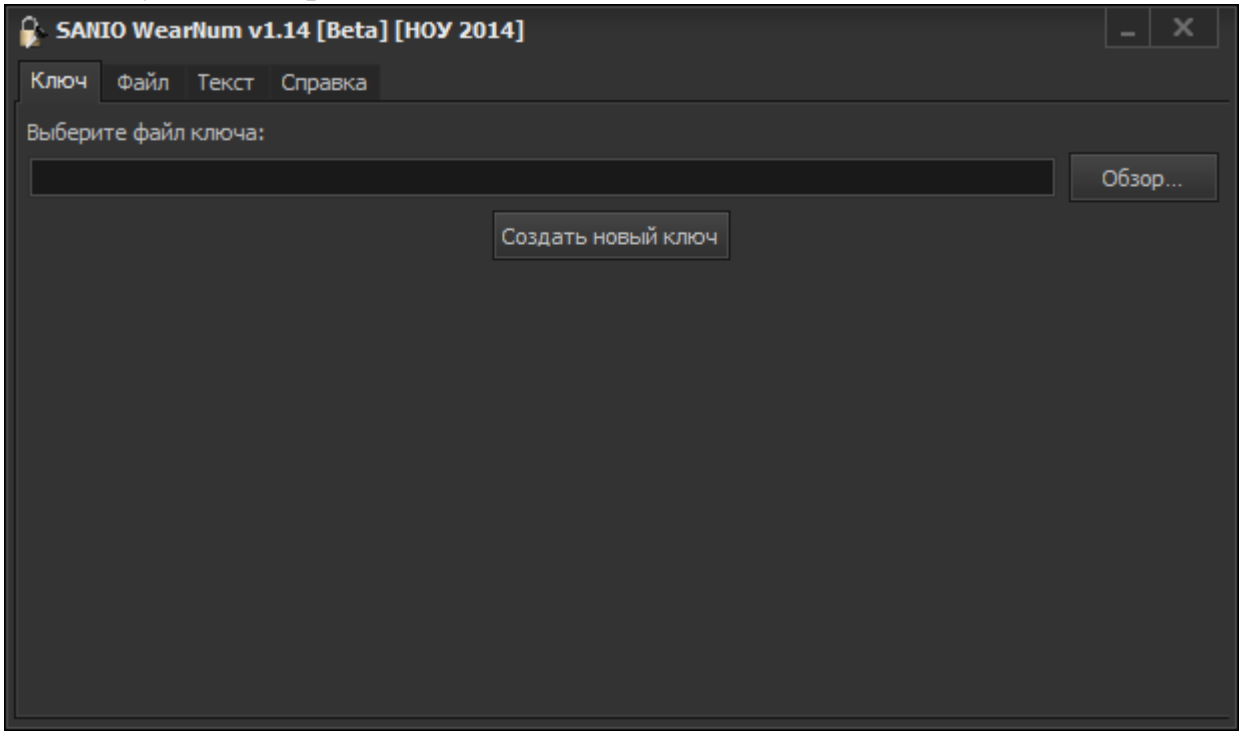
## SANIO WearNum

SANIO WearNum — криптографическое приложение, основанное на алгоритме Вернама с многочисленными улучшениями, такими, как функция проверки целостности ключа и всех шифруемых сообщений и новым, простым и надёжным генератором ключей с использованием человека как источника случайности. Ну и, естественно, с простым, красивым и интуитивно понятным интерфейсом. А также, высокой скоростью работы и не менее высокой надёжностью — хоть это и бета-версия, в ней не было замечено ошибок в работе, зависаний или каких-то других случаев. А если Вы найдёте такие, то всегда можете написать на электронную почту, и в следующей версии ошибка будет устранена. Приложение, естественно, распространяется бесплатно и с открытым исходным кодом, но до завершения закрытого бета-тестирования всем лицам, допущенным до этого самого тестирования, категорически запрещается распространять программу третьим лицам. Пока существует версия только для Windows, но планируется создание версий и для OS X, Linux и Android.

Последняя версия на момент защиты работы — 1.14, остерегайтесь подделок! В них могут быть встроены механизмы сбора информации и передачи их некомпетентным лицам.

Итак, о самом приложении:

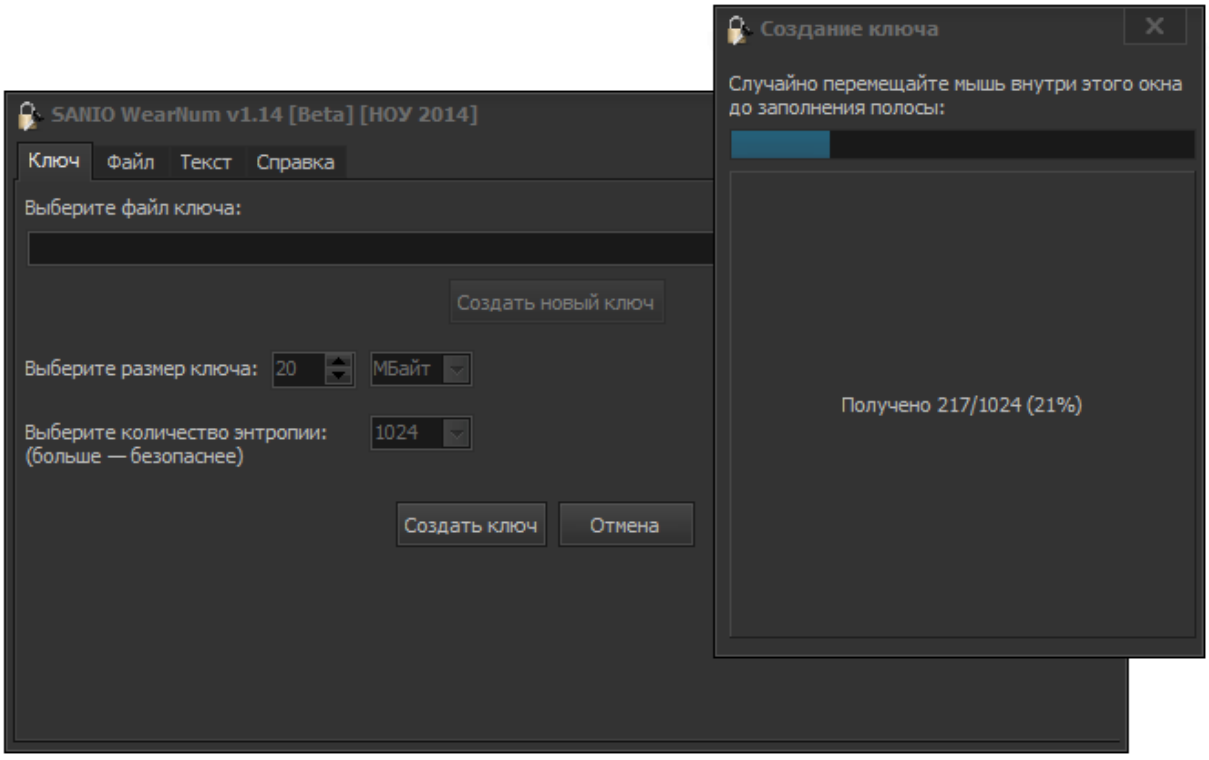
При запуске вы видите главное (и единственное) окно программы. Выглядит оно следующим образом:



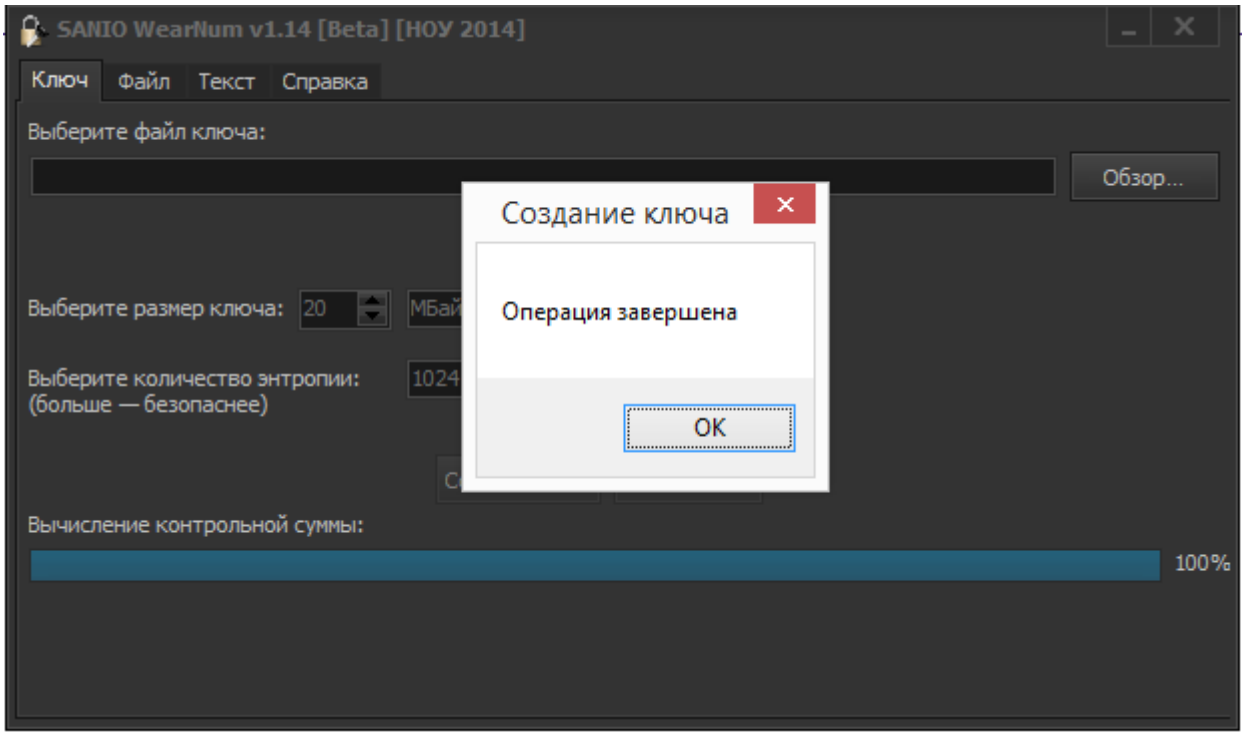
Оно состоит из четырёх вкладок. На первой вы можете создать или открыть ключ. На второй — зашифровать или расшифровать файл. На третьей — зашифровать или расшифровать текст. На четвёртой вкладке содержится информация о создателе и история версий.

Итак, давайте попробуем поработать с программой.

Сначала создадим ключ. Для этого нажимаем на кнопку «Создать новый ключ», задаём размер ключа (допустим, 20 МБ), и размер энтропии — величину, задающую количество собираемой случайной информации. Чем больше это число, тем надёжней будет ключ, но от вас потребуется больше телодвижений. Оставим стандартное значение — 1024. Нажимаем «Создать ключ». Далее вы увидите небольшое окошко. Это окошко собирает случайную информацию. Способ сбора этой информации выбран простой и надёжный — попросить пользователя перемещать мышь в специальной зоне.



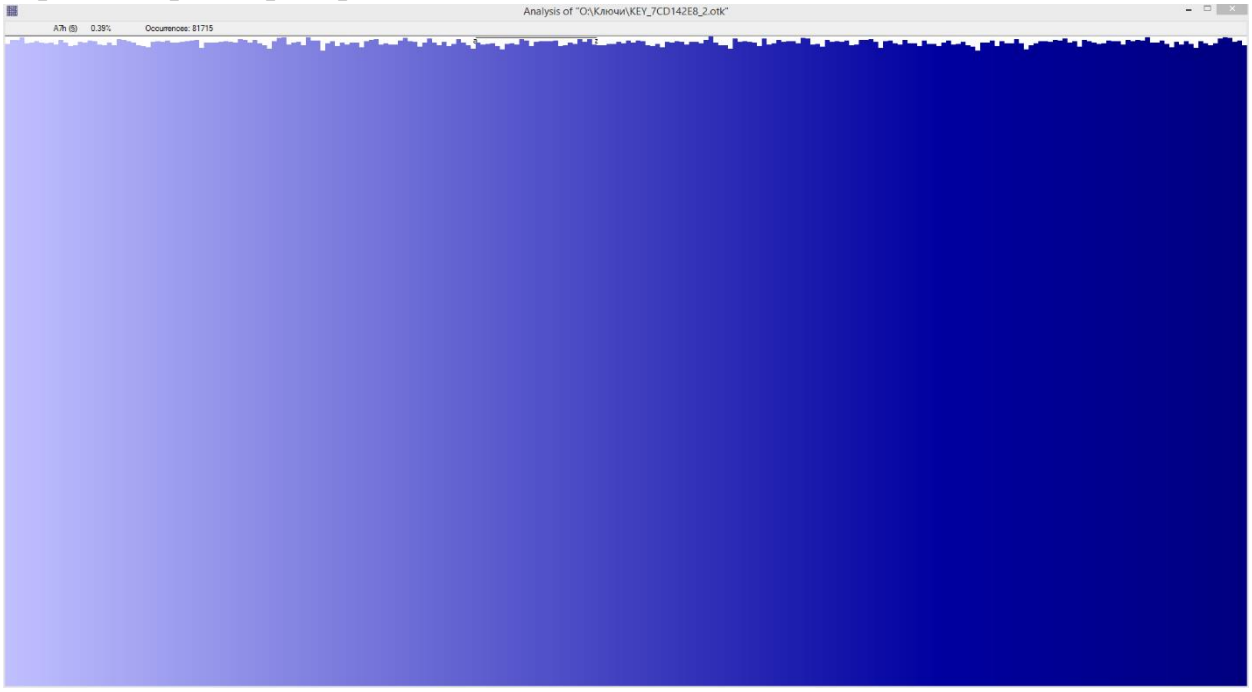

После того, как мы поводим мышью и выберем место сохранения и имя файла, начнётся создание ключа.



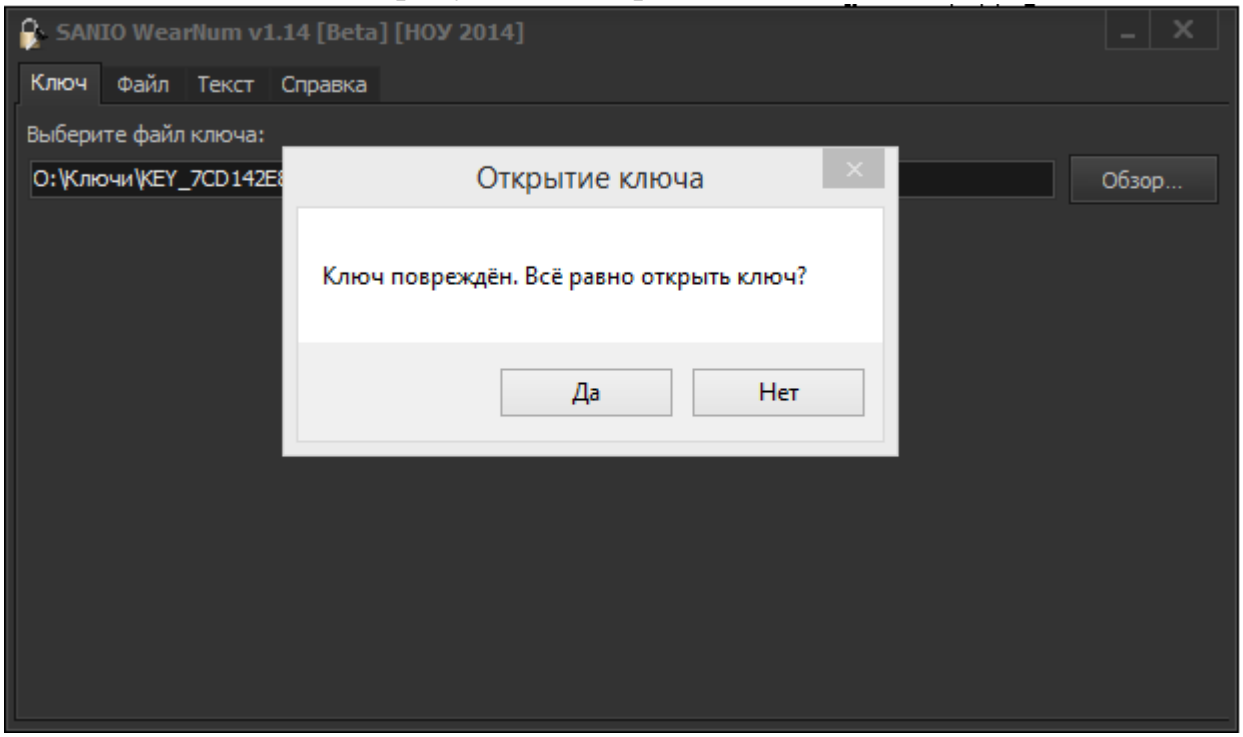

Операция завершена! Отлично. Давайте посмотрим, что из себя представляет новорождённый ключ. А представляет он из себя случайные числа.
Много случайных чисел.

| Offset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000000 | 35 | C5 | FD | EA | 9A | 03 | 6D | 72 | D9 | 4A | 2D | 97 | 9D | FD | 18 | 46 | 5Eэкљ mrЩJ-— э F |
| 00000010 | 92 | 31 | 31 | A8 | DE | 62 | 63 | BC | 1B | 6E | 3D | CE | 16 | EC | 7F | 4F | ’11ЁЮbcj n=O м O |
| 00000020 | A6 | 53 | 5F | A6 | E8 | 0C | 0F | 22 | 0E | C9 | A7 | 7D | BF | FE | 8E | 55 | ¦S_¦и " Й§}їкЋU |
| 00000030 | 43 | 0E | 37 | 89 | 4F | 02 | 85 | AE | C5 | 53 | 8C | 4E | 2B | A6 | 36 | E9 | C 7‰O …®ЕSЊN+¦6й |
| 00000040 | 20 | E1 | F7 | 28 | 73 | 2F | 98 | 46 | 6A | 32 | 79 | 9B | 4A | 31 | 62 | 34 | бч(s/F j2y›J1b4 |
| 00000050 | 65 | 76 | 64 | 68 | 4D | 80 | 90 | 5C | CE | 21 | CB | 23 | 39 | 70 | 4D | 33 | evdhMЂ \O!Л#9pM3 |
| 00000060 | 6A | 53 | AA | 9D | 5C | 0F | EF | EA | C9 | 56 | F6 | 32 | E3 | 01 | 11 | 65 | jSЄ \ пкЙVц2г e |
| 00000070 | 26 | 40 | 7D | 3C | 73 | 9D | AB | 37 | CF | C0 | 34 | 50 | 91 | C0 | 6E | E0 | &@}<s «7ПА4P‘Ana |
| 00000080 | 2B | 16 | 85 | 2E | CD | B9 | 5D | FF | AA | 72 | 13 | 40 | 92 | 61 | FE | 00 | + ….HЋ]яЄr @’аю |
| 00000090 | 27 | 6D | DD | 9F | B8 | 6E | A4 | C2 | 29 | B6 | FA | 69 | 95 | DD | 52 | E7 | 'mЭџёn¤В)¶ъi•ЭRз |
| 000000A0 | C6 | B4 | D6 | 93 | 0D | 20 | 50 | EB | FC | 06 | D9 | 6C | 8E | 22 | 34 | 63 | ЖґЦ“ Рль ЩlЋ"4c |
| 000000B0 | 47 | 16 | 29 | 0F | 92 | 16 | C9 | 46 | 45 | A8 | 5B | 68 | BB | 17 | C6 | B5 | G ) ’ ЙFЕЁ[h» Жµ |
| 000000C0 | 46 | 84 | 6F | E7 | 44 | 47 | C8 | 8C | 2F | 29 | 31 | A0 | 6A | AE | F1 | 1A | F„озDGИЊ/)1 j®с |
| 000000D0 | 8D | 4D | 1B | EF | A7 | B1 | 78 | AB | 45 | F2 | 18 | 21 | C1 | A4 | 0C | E2 | M п§±x«Ет !Б¤ в |
| 000000E0 | 63 | B3 | 5B | F9 | 44 | 30 | AE | 2B | D5 | 17 | E9 | 42 | 8D | 6B | D9 | C1 | ci[шD0®+X йВ kЩБ |
| 000000F0 | 0A | EC | 9E | F9 | A4 | 22 | 65 | 84 | F1 | 56 | 53 | 7B | 7C | 18 | 4A | 38 | мћш¤"e„cVS{\| J8 |

С равномерным распределением:




Теперь проверим функцию, определяющую целостность ключа. Изменим один байт в ключе и попробуем его открыть:



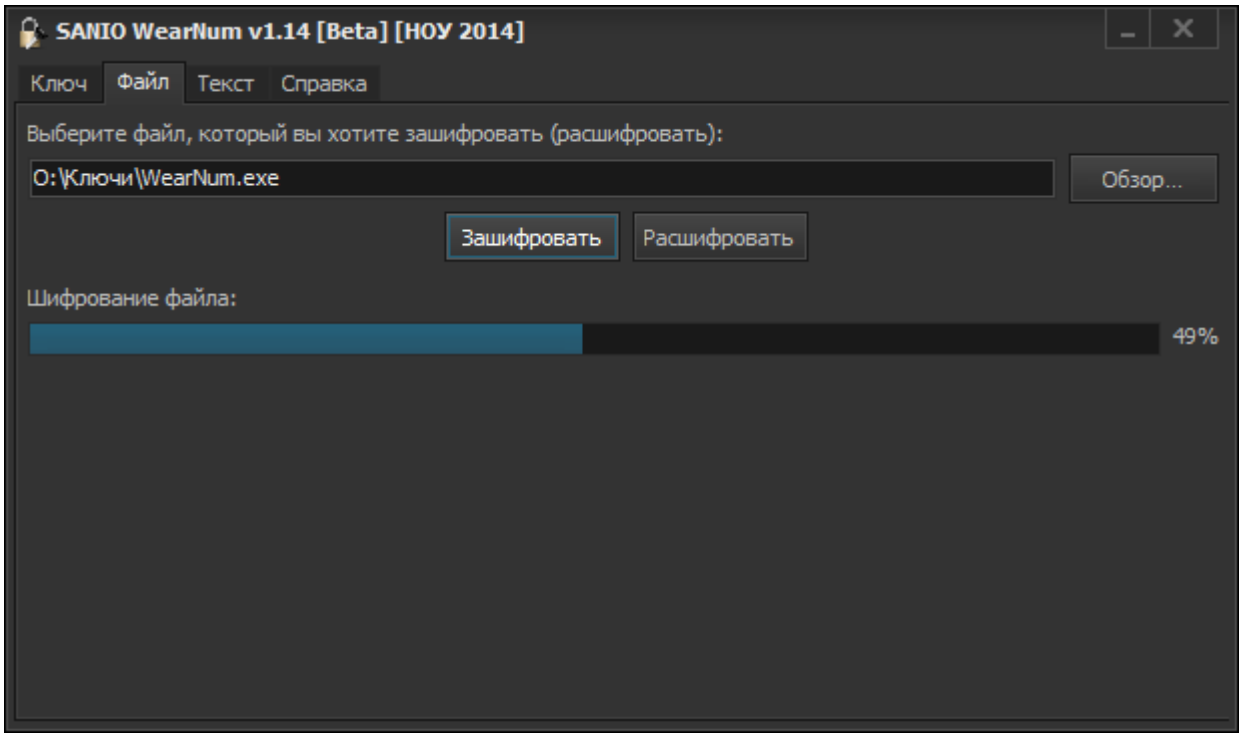

Программа выдаст ошибку, но на случай крайней необходимости всё же позволит использовать повреждённый ключ.

Кстати, о самой функции проверки целостности. В программе используется алгоритм вычисления контрольной суммы CRC32, она находится в последних четырёх байтах ключа, файла или текста и вычисляется при создании ключа и шифровании сообщений.

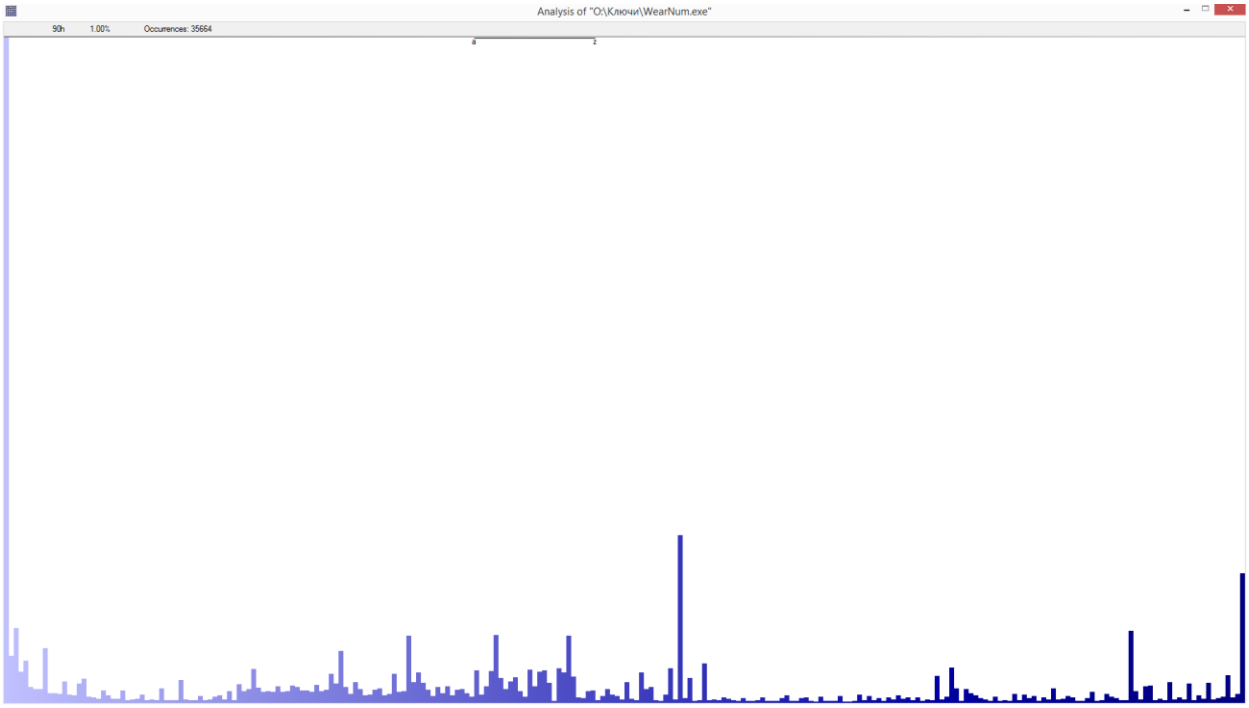
Код функции на языке C++:

```
unsigned int crc32(AnsiString fileAddr)
{
      big fileSize=getFileSize(fileAddr);
      FILE* file; file=fopen(fileAddr.c_str(),"rb");
      unsigned int ret=0, crcTable[256]={0};
      for(int i=0;i<256;i++)
      {
            ret=i;
            for(int j=0;j<8;j++)
            ret=ret&1?(ret>>1)^0xedb88320:ret>>1;
            crcTable[i]=ret;
      }
      ret=0xffffffff;
      while(fileSize--)
      {
            ret=crcTable[(ret^getc(file))&0xff]^(ret>>8);
      }
      fclose(file);
      return ret;
}
```

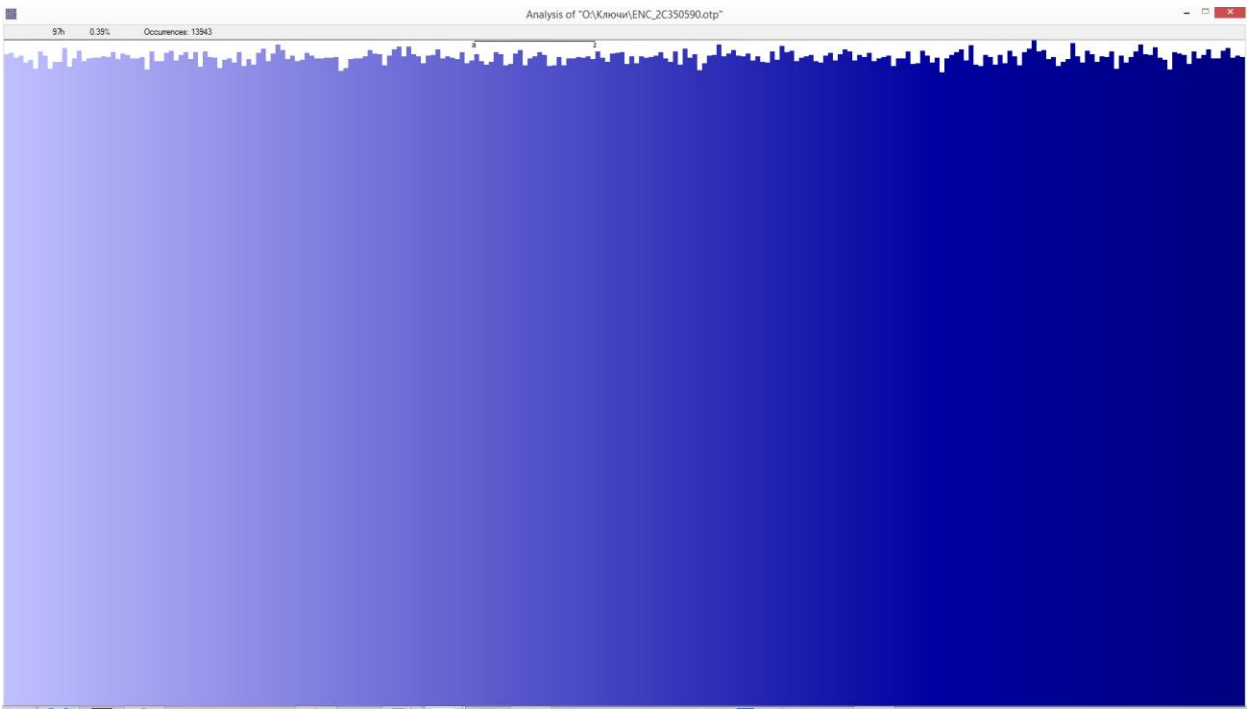
Хорошо, ключ загружен. Давайте теперь попробуем зашифровать что-нибудь. Давайте зашифруем этой программой, например, саму себя. Переходим на вкладку «Файл», выбираем файл и нажимаем «Зашифровать». Выбираем место сохранения файлов и ждём завершения операции:



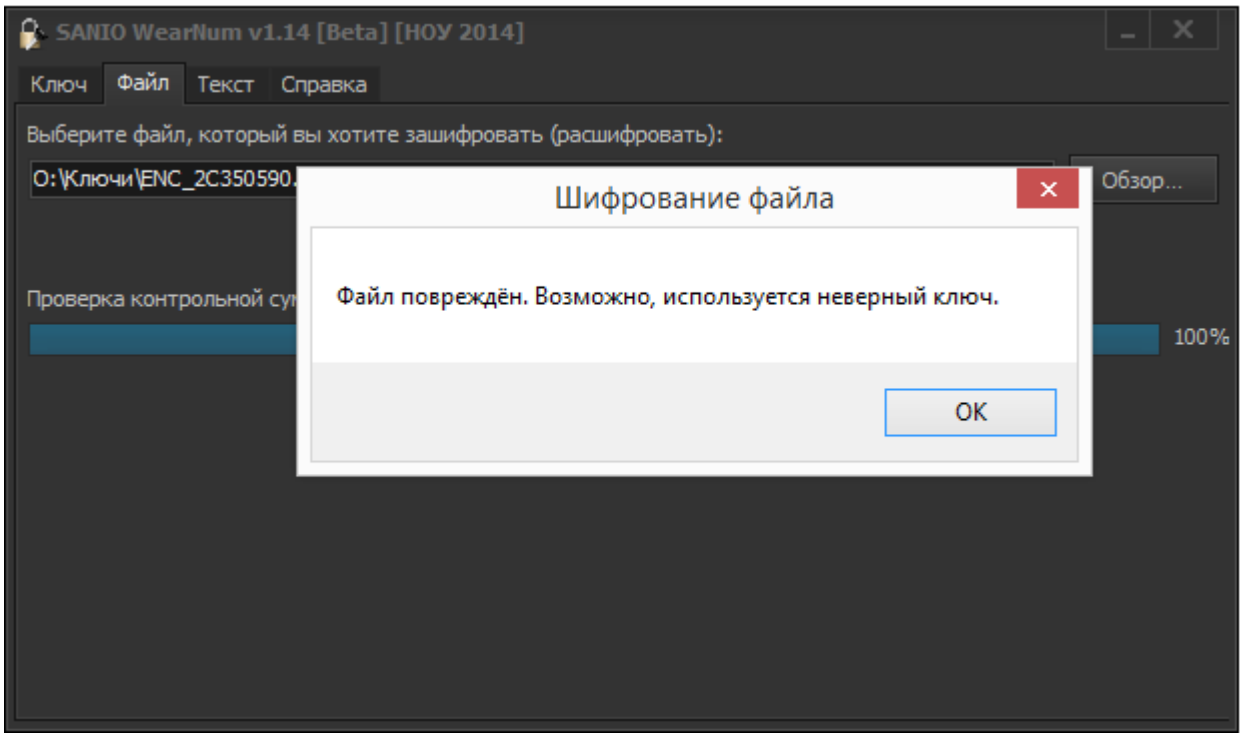
В результате шифрования файла с распределением, типичным для исполняемого файла:




Мы получаем файл с равномерным распределением, в очередной раз подтверждая факт, гласящий, что сумма двух величин, одна из которых является случайной с равномерным распределением, также является случайной с равномерным распределением:



Теперь давайте ещё раз проверим функцию, отвечающую за проверку целостности, на этот раз на зашифрованном файле. Изменим, как обычно, один байт и попробуем расшифровать:




Программа пристально следит за невнимательным пользователем, который может случайно выбрать не тот ключ, или не тот файл, а без этой функции ключ, который мог быть необходим для дешифровки других сообщений, был бы безвозвратно утрачен, ведь программа надёжно уничтожает все использованные байты, чтобы никакие программы не смогли его достать.

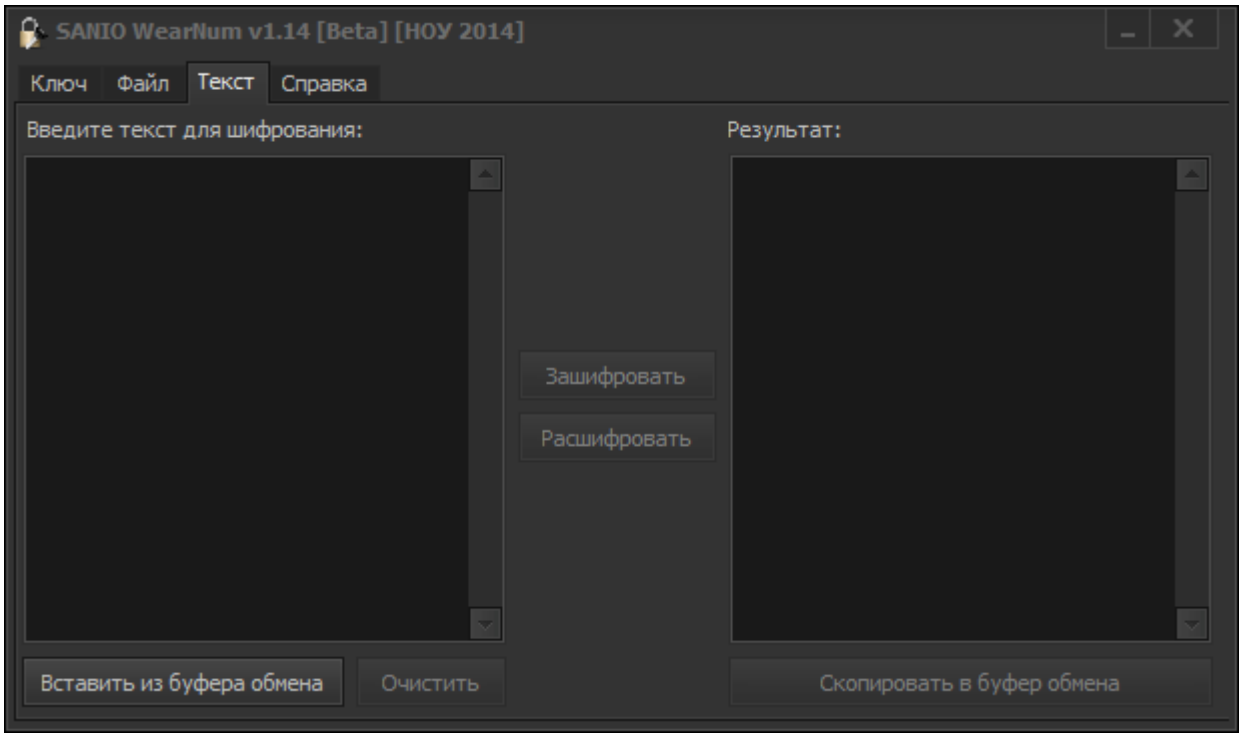
Кстати, вот код функции, шифрующей файлы:

```
int fileEncrypt(AnsiString inAddr,AnsiString outAddr,AnsiString
tabAddr)
{
     big fileSize=getFileSize(inAddr);
     big tableSize=getFileSize(tabAddr);
     if(fileSize>tableSize-4) throw 1;
     unsigned int chkSum=0;
     if(!fileDecFlag) chkSum=crc32(inAddr,0);
     FILE* inFile; inFile=fopen(inAddr.c_str(),"rb");
     FILE* outFile; outFile=fopen(outAddr.c_str(),"wb");
     FILE* tabFile; tabFile=fopen(tabAddr.c_str(),"rb");
     if(fileDecFlag) fileSize-=4;
     for(big i=0;i<fileSize;i+=8)
     {
          fseek(tabFile,-i-5,SEEK_END); big in=0,tab=0,out=0;
          if(i-fileSize>=8)
          {
               fread(&in,8,1,inFile);
```

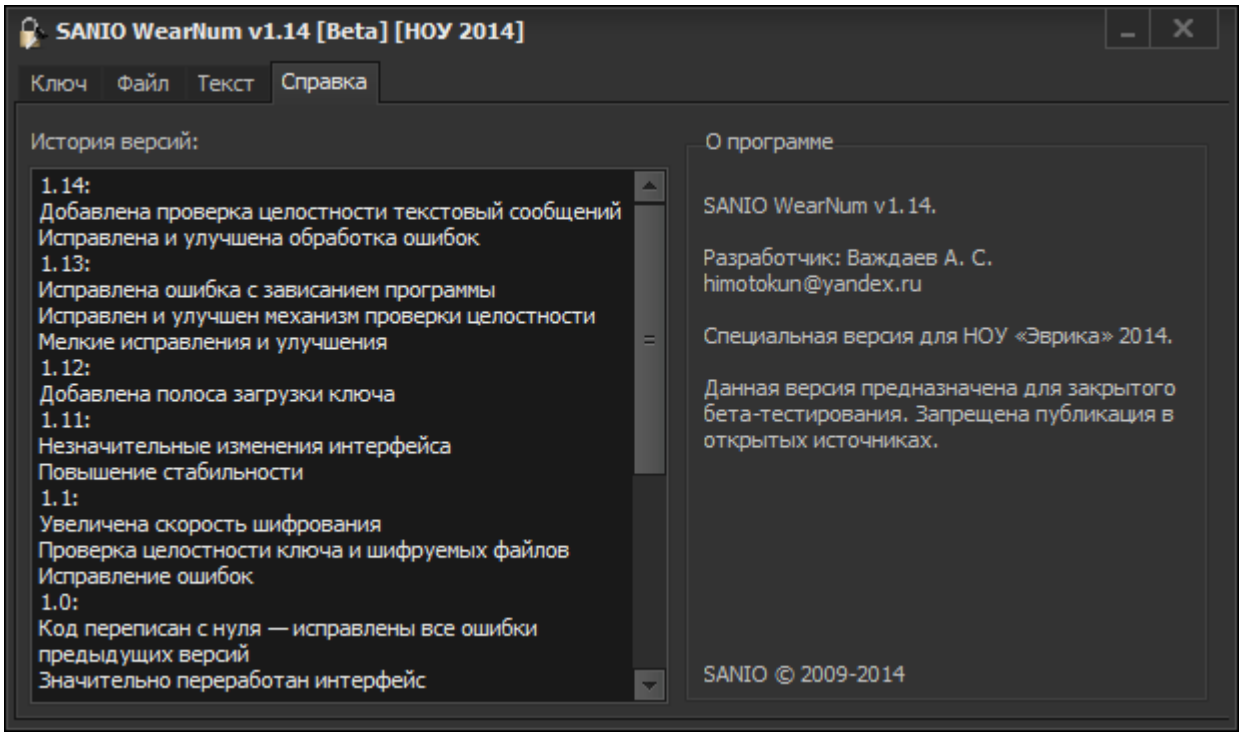
```
                fread(&tab,8,1,tabFile);
                out=in^tab;
                fwrite(&out,8,1,outFile);
            }
            else for(;i<fileSize;i++)
            {
                fseek(tabFile,-i-5,SEEK_END);
                fputc(fgetc(inFile)^fgetc(tabFile),outFile);
           }
      }
      int ret=0;
      if(!fileDecFlag)
      {
            unsigned int chkTab=0;
            fseek(tabFile,-fileSize-4,SEEK_END);
            fread(&chkTab,4,1,tabFile);
            chkTab^=chkSum;
            fwrite(&chkTab,4,1,outFile);
      }

else
      {
            fclose(outFile);
            chkSum=crc32(outAddr,0);
            unsigned int chkTab=0,chkGet=0;
            fseek(tabFile,-fileSize-4,SEEK_END);
            fread(&chkTab,4,1,tabFile);
            fread(&chkGet,4,1,inFile);
            chkTab^=chkGet;
            if(chkTab!=chkSum) ret=1;
     }
      fclose(inFile);
      fclose(outFile);
      fclose(tabFile);
      if(ret) remove(outAddr.c_str());
      if(ret) return 1;
      int _handle=open(tabAddr.c_str(),O_WRONLY);
      if(chsize(_handle,tableSize-fileSize-4)) throw 2;
      unsigned int newCrc=crc32(tabAddr,0);
      tabFile=fopen(tabAddr.c_str(),"ab");
      fwrite(&newCrc,4,1,tabFile);
      fclose(tabFile);
     return ret;
}
```

Ключ читается с конца, чтобы можно было уменьшать его размер по мере использования. Для достижения большей скорости файл и ключ считываются блоками по восемь байт. Так сокращается число операций по перемещению указателя. Затем в выходной файл записывается XOR двух блоков. Если остаётся нецелый блок, он считывается побайтово. После шифрования в файл и ключ записывается контрольная сумма.

Шифрование текста выполняется точно так же, как и файлов:




И, наконец, последняя вкладка, с информацией:



## Генератор случайных чисел

О безопасности самого шифра сказано уже достаточно. Но самая главная его часть — генератор случайных чисел для ключа. Поскольку надёжность шифра обеспечивается именно случайностью ключа, этот вопрос решался в первую очередь. Как известно, в качестве ключа для шифра Вернама должна использоваться истинно случайная последовательность. Все возможные попытки взломать шифр Вернама сводятся к атаке на ключ. Если он будет создан с помощью какого-либо алгоритма, он будет вовсе не случайным, а только псевдослучайным. И хотя человеку такие числа могут казаться совершенно случайными, компьютер легко найдёт в них закономерность, а значит, сможет воссоздать ключ и расшифровать сообщение.

В WearNum для создания ключа используется технология сбора случайности с человека: он водит мышью внутри окна, координаты считываются, особым образом обрабатываются, объединяются с псевдослучайной последовательностью для большей равномерности распределения (это не вредит случайности, так как сумма любой величины со случайной является случайной) и записываются в выходной файл.

Но прежде чем говорить о качестве генератора ключей, его необходимо проверить — а так ли он хорош в действительности, как это кажется?

Как проверить последовательность чисел на случайность? Для этого существуют специальные тестирующие системы, с лёгкостью отыскивающие закономерности в самых сложных псевдослучайных последовательностях и честно сдающиеся перед истинно случайными. Одна из самых лучших таких систем под названием DIEHARD и была использована для тестирования.

Были проверены несколько ключей, размером от нескольких мегабайт до нескольких гигабайт. Система в результате тестирования по каждому из 15 тестов выдаёт числа, называемые p-value. Уровень случайности зависит от близости этого числа к нулю или единице. Чем ближе — тем хуже.

Для ключей, созданных WearNum, p-value составляло от 0.336187 до 0.653623. Результат просто отличный. Для сравнения, большинство криптографически стойких ГПСЧ выдают результаты 0.1-0.2 или 0.8-0.9. А стандартные функции ГПСЧ вообще всегда выдают единицу.

## Почему стоит использовать WearNum?

И действительно, чем так хороша моя программа, и чем она лучше других? Почему стоит использовать её, а не любую другую аналогичную программу?

Начнём с того, что других таких программ просто нет. Сколько я ни искал, везде предлагается сделать всё самостоятельно или использовать AES. Но ведь не все люди — программисты. А безопасной связи хотят многие. И мне однажды тоже понадобилась такая программа. Но её нигде не было. А обычные блочные шифры мне совсем не подходили. Поскольку я немного умею программировать, я подумал: «Если такой программы нет, а она мне нужна, почему бы мне не написать её самому?»

Через пару дней была готова первая версия. Но она работала медленно и неохотно. Я решил не останавливаться, и улучшить программу и исправить ошибки. Через некоторое время появилась следующая версия. И так далее.

Текущая версия — 1.14, и она далеко не последняя.

Итак, преимущества WearNum:

- Единственная в своём роде программа для работы с шифром Вернама
- Высокая скорость работы
- Стабильность — программа не «вылетит» и не повредит ваши файлы, если вдруг что-то пойдёт не так.
- Удобство интерфейса. «Встречают по одёжке».
- Позволяет обмениваться как текстовыми сообщениями, так и файлами
- Ключ и всё, что вы шифруете, подвергается проверке на целостность. Если злоумышленник захочет подменить текст, у него ничего не получится. Также программа не позволит вам расшифровать послание не тем ключом, которым оно было зашифровано. Человеку свойственно ошибаться. А если случайно выбрать не тот ключ, то всё, что было зашифровано тем ключом, будет потеряно вместе с ним. WearNum этого не допустит.

Если вам нужна действительно безопасная связь, используйте WearNum.

Если вам нужно надёжно спрятать от людей в форме большое количество информации, которая вам потом понадобится, используйте жёсткий диск, закопанный где-нибудь в лесу. WearNum здесь бессилен вам помочь.

Но никогда не используйте AES. В нём есть уязвимость, позволяющая взламывать его. И об этой уязвимости известно АНБ.

## Заключение

Напоследок хочется сказать следующее:

Не стоит злоупотреблять безопасностью. Надёжность шифра должна соответствовать уровню секретности информации. Не стоит использовать шифр Вернама для отправки друзьям фотографий с Нового Года. Если, конечно, это фотографии секретных помещений Пентагона, то их стоит зашифровать. Вряд ли АНБ захочет почитать вашу переписку о том, как вы вчера погуляли. Но вот если вы обсуждаете планы по свержению теневого правительства и построению коммунизма на отдельной взятой планете, то стоит позаботиться о том, чтобы эти сведения не дошли до тех, кого будут свергать. Тут-то и пригодится моё средство.

## Отказ от ответственности

Работа выполнена исключительно в исследовательских целях и ни в коем случае не нарушает закон №149-ФЗ «Об информации, информационных технологиях и защите информации» и указ Президента РФ №334 «О мерах по соблюдению законности в области разработки производства, реализации и эксплуатации шифровальных средств, а также предоставления услуг в области шифрования информации». Все алгоритмы, использованные при разработке приложения, являются общеизвестными и разрешёнными к использованию на территории РФ. Созданное в результате исследования приложение разрешается использовать только в ознакомительных целях на свой страх и риск. Автор не несёт ответственности за возможные последствия, вызванные ненадлежащим использованием приложения.

Работа ни в коем случае не пропагандирует национализм в отношении США и недоверие к государственному аппарату и органам безопасности. Все источники, использованные при выполнении работы являются общедоступными и разрешёнными на территории РФ. Автор не несёт ответственности за использование работы в целях, нарушающих законодательство РФ, а также любые другие последствия, вызванные прочтением, распространением или иным использованием сведений, изложенных в работе.

## Список используемых источников


1. Брюс Шнайер. «Прикладная криптография» (http://files.zipsites.ru/books/cryptoshn.pdf)
2. http://ru.wikipedia.org/wiki/Шифр_Вернама
3. http://samoucka.ru/document34160.html
4. http://forum.nswap.info/index.php?topic=4595.0
5. http://www.computermaster.ru/articles/secur5.html
6. http://old.computerra.ru/1999/305/3111/