Click to edit the notes format

fppt.com

Click to edit the notes format

fppt.com

Click to edit the notes format

fppt.com

Click to edit the notes format

fppt.com

Click to edit the notes format

fppt.com

Click to edit the notes format

fppt.com

Click to edit the notes format

fppt.com

Click to edit the notes format

fppt.com

Click to edit the notes format

fppt.com

Click to edit the notes format

fppt.com

Click to edit the notes format

fppt.com

Click to edit the notes format

fppt.com

Click to edit the notes format

fppt.com

Click to edit the notes format

fppt.com

Click to edit the notes format

fppt.com

Результаты экспериментов: увеличение скорости работы

Click to edit the notes format

Наименование шифра	R	T(R) (секунд)	R_{min}	$T(R_{min})$(секунд)	T_R/T_{Rmin}
CLEFIA	18	464.414	6	161.8	2.87
HIGHT	32	65.086	10	22.973	2.83
SKIPJACK	32	7.199	14	4.567	1.58
LED	32	2037.02	4	264.599	7.70
AES	10	347.881	4	135.562	2.57
KTANTAN64	254	9459.49	30	1172.56	8.07

Результаты работы по библиотеке

Click to edit the notes format

- Создана программная библиотека из более чем 20 шифров (основу составили легковесные шифры библиотеки BLOC);
- проведены проверки корректности реализаций алгоритмов;
- найдены значения минимального числа раундов;
- результаты исследования опубликованы в журналах, рецензируемых РИНЦ и ВАК.

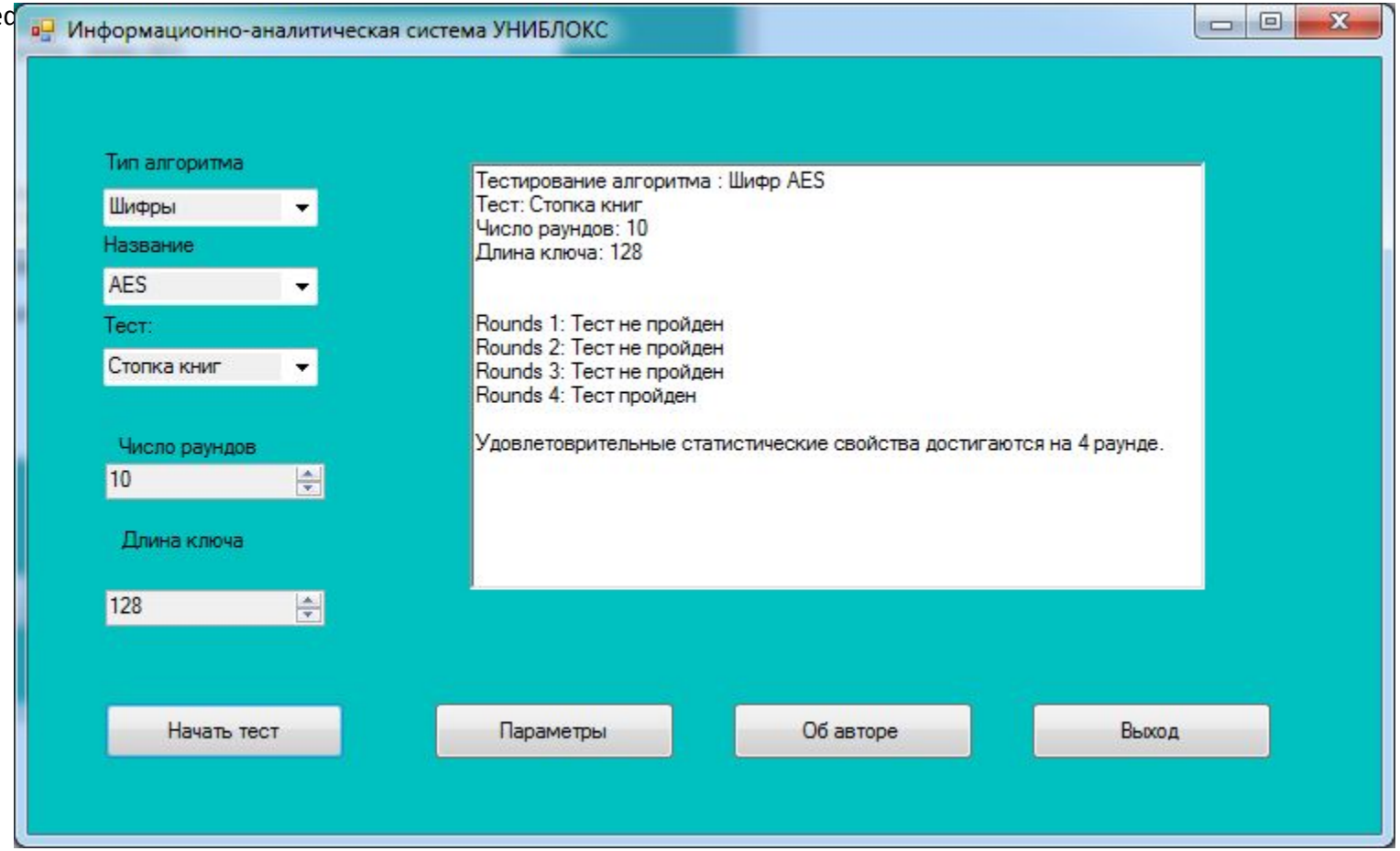
Хеш-функции – текущий этап

Click to edit the notes format

- Собраны алгоритмы хеширования с конкурса SHA-3 и некоторые известные хеш-функции:
  - Keccak
  - CRC
  - Digest
  - MD5
  - SHA-1
  - SHA-256
  - BLAKE
- и другие алгоритмы…

Интерфейс

Click to ed



Преобразование блоков

Click to edit the notes format

```
void _16to32(u16* text16, u32* text32, int n_words32) {
    For (int i = 0; i < n_words32; i++) {
        text32[i] = text16[2 * i];
        Text32[i] <<= 16;
        text32[i] += text16[2 * i + 1];
    }
}
```

Тестирования шифров

Click to edit the notes format

- Статистические свойства шифров анализируются с помощью тестов «стопка книг» и «адаптивный критерий хи-квадрат»

- Тесты предназначены для проверки гипотезы H_0 о том, что буквы некоторого алфавита $A = \{a_1, a_2, \ldots, a_s\}$, где $S>1$ порождаются с равными вероятностями: $H_0 : \{P(a_1) = , P(a_2) = \ldots = P(a_s) = 1/S$

- Подсчитывается величина *«ню»,* по которой подсчитывается критерий $\chi 2$ по формуле:

$$\chi^2 = \sum_{i=1}^{k} \frac{(v_i - NP_i^0)^2}{NP_i^0}$$

Адаптивный критерий хи-квадрат

Click to edit the notes format

- Основная идея: дан алфавит

  $A = \{a_1, a_2, \ldots, a_s\}$, где $S>1$

  $H_0 : \{P(a_1) = , P(a_2) = \ldots = P(a_s) = 1/S$
- Создаётся две выборки – «обучающая» и «контрольная», производится поиск зашифрованных символов «контрольной выборки» в «обучающей».

Адаптивный критерий хи-квадрат

Click to edit the notes format

$$\chi^2 = \sum_{i=1}^{k} \frac{(v_i - \mathrm{NP}_\mathrm{i}^0)^2}{\mathrm{NP}_\mathrm{i}^0},$$

, где

- v_i - количественное значение отклонения
- $NP=2^{12}$ (при длине контрольной выборки 2^{24})
- Нормальным является коэффициент отклонения 3,84.

Тест «Стопка книг»

Click to edit the notes format

Идея метода: При тестировании по данному методу буквы алфавита *A* ($a_1,\ldots,a_i$) *упорядочены* причем этот порядок меняется после анализа каждого выборочного значения x_i следующим образом

При применении описываемого теста множество всех {1,...,S} заранее, до анализа выборка, разбивается на r>1 непересекающихся частей $A_1 = \{1,2,\ldots,k_1\}$, $A_2 = \{k_1+1,\ldots,k_2\}$,..., $A_r = \{k_{r-1}+1,\ldots,k_r\}$.

$$\nu^{t+1}(a) = \begin{cases} 1, & \text{если} \quad x_t = a, \\ \nu^t(a) + 1, & \text{если} \quad \nu^t(a) < \nu^t(x_t), \\ \nu^t(a), & \text{если} \quad \nu^t(a) > \nu^t(x_t). \end{cases}$$

Пример работы с библиотекой

Click to edit the notes format

setCipher(cipherID);

```
void setCipher(Ciphers cipher) {
if (cipher == LBLOCK) {
encrypt = Lblock::encrypt;
resetKey = Lblock::resetKey;
id = Lblock::id;
runTests = Lblock::runTests;
}
else if (cipher == PRESENT) {
. . .
}
```

void resetKey()

void encrypt(u32* text, int rounds)

```
void resetKey() {
for (int i = 0; i < 16; i++) {
Key[i] = rand()&0xFF;
}
KeyExpansion();
}

void encrypt(unsigned int* text,
   int rounds)
{
_32to8(in, text, 4);
Cipher(rounds);
_8to32(out, text, 4);
}
```