Механизм коррекции результатов дешифровки квантово-устойчивого шифра на основе нейросети

Чайко Владимир Иванович, студент

Кузбасский гуманитарно-педагогический институт Кемеровского государственного университета (г. Новокузнецк)

В данной статье автор раскрывает сложности реализации квантово-устойчивого шифра на основе нейронной сети при реализации на ЭВМ и показывает, как их можно устранить.

Ключевые слова: *квантово-устойчивый шифр, нейронная сеть, тип данных, сложность вычисления.*

Одним из вариантов квантово-устойчивого шифрования является шифр, основанный на применении нейронной сети. Данный шифр является теоретически не взламываемым, как и шифр Вернама, [1] и не требует обмена ключами между двумя сторонами.

На практике реализация данного шифра на ЭВМ сталкивается с рядом ограничений, обусловленных ограничениями нейронов и самой архитектурой ЭВМ. Использование в алгоритме шифрования математических нейронов оказывает ограничения на формат представления шифруемых данных. Формат и диапазон допустимых значений зависит от того, какую активационную функцию используют для реализации математических нейронов. [2] Основные функции активации, и ограничения, накладываемые ими, представлены в таблице 1.

Таблица 1. **Функции активации и ограничения, накладываемые ими**

Функция активации	Формула	Область	
		Определения	Значения
Сигмоида	$y = \frac{1}{1+e^{-x}}$	$-\infty < x < +\infty, R$	$0 < x < 1, \in \mathrm{R}$
Тангенс	$y = \tan x$	$x \in R \setminus \left\{ \frac{\pi}{2} + \pi k : k \in Z \right\}$	$y \in R$
Арктангенс	$y = \tan^{-1} x$	$-\infty < x < +\infty, R$	$-\frac{\pi}{2} < y \leq \frac{\pi}{2}$
Гиперболический арксинус	$y = arsh\, x$	$-\infty < x < +\infty, R$	$-\infty < y < +\infty, R$
Гиперболический арккосинус	$y = arsh\, x$	$1 \leq x < +\infty, R$	$1 \leq y < +\infty, R$

Типы данных «float» и «double» регулируется международным стандартом IEEE 754, согласно которому, под данный тип данных, ЭВМ должна отводить 32 и 64 бита памяти соответственно. Это позволяет хранить дробные числа с 6 и 15 знаками после запятой. [3] Если, в результате вычислений, получилась дробь с большим количеством разрядов, или же бесконечная дробь, ЭВМ «отбрасывает» все знаки, находящиеся после 6 или 15 разряда. [4] Многие функции активации нейронов и тригонометрические функции могут иметь значения, гораздо превышающие 15 разрядов после запятой. Поэтому зачастую реализация подобных функций на ЭВМ выдает результат с некоторой отрицательной погрешностью. [5]

При реализации каких-либо вычислений с числами типа «float» и «double» на ЭВМ точность вычислений снижена по причине сложности их реализации. Сложность заключается в том, что человек использует десятичную систему счисления, а компьютер двоичную. Дробь в десятичной системе счисления невозможно перевести в двоичную. Вместо этого, в память ЭВМ, записывается максимально приближенное к дроби двоичное число. Размер погрешности приближенного числа разнится и зависит от ОС. Такой тип данных нельзя использовать для сравнения на предмет равенства. [4]

Тригонометрические функции, при вычислении на ЭВМ, так-же считаются с погрешностями. Это вызвано тем, что ЭВМ использует упрощенные методы их вычислений, например Ряды Тейлора. Это приводит к тому, что результат отличается от истинного (имеет погрешность). [6]

В построении нейронных сетей наличие данных погрешностей несущественно, а в реализации шифрования критично. «Отбрасывание» знаков после запятой приводит к потере данных, а погрешность в расчетах не позволяет точно декодировать шифр. Примеры этого представлены в таблице 2.

Таблица 2. **Примеры искажения передаваемых данных**

Отправлено	Получено	Δ	Δ%
0,657	0.67444031601614	–0.00055968398385586	–0.082916145756424
0,1	0.099998176456816	–0,00000182354	–0.0018235431837044
0,25	0.2499715139834	–0,00002848601	–0.011394406638487
0,892	0.89071040554366	–0.0012895944563356	–0.14457336954435

Связи с этим, при реализации шифрования данных предлагаемой системой шифрования, программист должен осуществить коррекцию данных после де-

шифровки. Наиболее простым способом коррекции данных является использование псевдоодносторонней функции деления с остатком (1). [7]

$$a \bmod b = q\ (ост.r) \quad (1)$$

Особенность данной псевдоодносторонней функции заключается в том, что, зная делимое и делитель (a и b), всегда можно найти неполное частное (q) и остаток от деления (r). Однако, зная остаток от деления (r) и делитель (b), невозможно сказать, каким было делимое (a), т. к. их бесконечное множество. Пример подбора делимых представлен в таблице 3.

Таблица 3. **Пример подбора делимых**

Делитель	Остаток	Делимое	Расчет
2	0	2	$2 \bmod 2 = 1\ (ост.\ 0)$
2	0	4	$4 \bmod 2 = 2\ (ост.\ 0)$
2	0	6	$6 \bmod 2 = 3\ (ост.\ 0)$
2	0	8	$8 \bmod 2 = 4\ (ост.\ 0)$
2	0	10	$10 \bmod 2 = 5\ (ост.\ 0)$
2	0	12	$12 \bmod 2 = 6\ (ост.\ 0)$
2	0	14	$14 \bmod 2 = 7\ (ост.\ 0)$

Таким образом, передача неполного частного и остатка от деления перехватчику не даст вычислить делимое.

Условимся, что передаваемое сообщение (s) — это частное, а остаток от деления (r) должен быть равен 0. Тогда мы можем вычислить для сообщения делитель (d), который можно передать вместе с сообщением. (2)

$$s \bmod d = q\ (ост.0) \quad (2)$$

Данный делитель можно будет использовать для коррекции числа следующим образом: в расшифрованное сообщение (число) нужно прибавлять 1 в младший разряд до тех пор, пока не получится число, которое без остатка разделится на делитель. (3)

$$s \bmod d = n, n \in N \quad (3)$$

Первое число, которое разделится без остатка на делитель и будет корректным сообщением.

Найти такой делитель очень просто: достаточно разделить сообщение (s) на целое случайное число. (4)

$$d = \frac{s}{n}, n \in N \quad (4)$$

Передав такой делитель вместе с зашифрованным сообщением, получатель сможет расшифровать данное сообщение и откорректировать. При этом передача делителя угрозы не несет.

Шифр на основе нейронной сети с коррекцией результата расшифровки представлен на листинге 1.

Листинг 1. Шифр на основе нейронной сети с коррекцией результата.

```
<?php
      function neiro($x) {
            $y=asinh($x);
            return $y;
      }
      function anti_neiro($y) {
            $x=acosh($y);
            return $y;
      }
      function prov_f($x) {
            return $x/rand(1, 10);
      }
      function correct_f($x, $prov) {
          $str=explode(".", strval($x));
      $correct_min="0.";
            for ($i=0; $i<(strlen($str[1])-1); $i++) {
                  $correct_min=$correct_min."0";
            }
            $correct_min=floatval($correct_min."1");
            for ($i=0; $i<1; $i++) {
                  $i--;
                  $u=floatval($x)/floatval($prov);
                  $u=floatval(round($u)-$u);
                  if ($u == 0) {
                  $u=1;
                  $i=2;
                  }
                  else {
```

```
                $x=floatval($x)+floatval($correct_min);
                }
            }
            return $x;
        }
        $key1=0.10; //Первый ключ Алисы
        $key2=0.20; //Второй ключ Алисы
        $key11=0.30; //Первый ключ Боба
        $key22=0.40; //Второй ключ Боба
        $pered=0.123456789; //Передаваемое сообщение
        $prov=prov_f($pered);
        echo $pered." // Сообщение ".$prov." Число
        проверки\n";
        $etap1=neiro(neiro($pered*$key1)*$key11); //
        Шифрование Алисы
        echo $etap1." // Сообщение зашифрованное Алисой \n";
        $etap2=neiro(neiro($etap1*$key2)*$key22); //
        Шифрование Боба
        echo $etap2." // Сообщение зашифрованное Бобом \n";
        $etap3=anti_neiro(anti_neiro($etap2)/$key11)/$key1;
        //Дешифровка Алисы
        echo $etap3." // Алиса сняла свое шифрование \n";
        $etap4=anti_neiro(anti_neiro($etap3)/$key22)/$key2;
        //Дешифровка Боба
        echo $etap4." // Боб снял свое шифрование \n";
        echo "\n";
        $correct=correct_f($pered, $prov);
        echo $correct." // Боб откорректировал свое значение
        \n";
        echo "\n";
?>
```

Результат работы данной программы представлен на листинге 2.

Листинг 2. Результат работы программы.

```
0.123456789 // Сообщение 0.013717421 Число проверки
0.0037036011257876 // Сообщение зашифрованное Алисой
0.0002962880586341 // Сообщение зашифрованное Бобом
0.0098762686211368 // Алиса сняла свое шифрование
```

```
0.12345335776421 // Боб снял свое шифрование

0.123456789 // Боб откорректировал свое значение
```

Данный код был написан на языке программирования PHP 8.4 [8] и проверен на онлайн интерпретаторе OnlineGDB. [9]

Литература:

1. Невзламываемый шифр Вернама // КОД. Код журнал Яндекс Практикума. URL: https://thecode. media/vernam/ (дата обращения: 03.01.2025).
2. Rashid, Tariq. Make Your Own Neural Network / Tariq Rashid. CreateSpace, 2016. — 222с. — Текст: непосредственный.
3. IEEE 754. IEEE Standard for Binary Floating-Point Arithmetic. — NewYork: American National Standard, 1985. — 18 с. — Текст: непосредственный.
4. Числа с плавающей точкой. — Текст: электронный // php.net: [сайт]. — URL: https://www.php.net/manual/ru/language.types.float.php (дата обращения: 15.01.2025).
5. Карцев, М.А. Арифметика цифровых машин / М.А. Карцев. — М.: Наука, 1969. — 576 с. — Текст: непосредственный.
6. Как компьютер считает синусы. — Текст: электронный // КОД: [сайт]. — URL: https://thecode. media/sinus/ (дата обращения: 15.01.2025).
7. MOD function. — Текст: электронный // Microsoft: [сайт]. — URL: https://support.microsoft.com/en-us/office/mod-function-9b6cd169-b6ee-406a-a97b-edf2a9dc24f3 (дата обращения: 15.01.2025).
8. PHP 8.4 // PHP. URL: https://www.php.net/releases/8.4/ru. php (дата обращения: 15.01.2025).
9. OnlineGDB. URL: https://www.onlinegdb.com/online_php_interpreter (дата обращения: 15.01.2025).