

# ОГЛАВЛЕНИЕ

Введение . . . . .	3
Постановка задачи . . . . .	4
1. Общая информация о криптосистеме Эль-Гамала . . . . .	5
1.1 Алгоритм создания открытого и закрытого ключей . . . . .	6
1.2. Шифрование и расшифрование . . . . .	6
1.3. Дешифрование . . . . .	7
1.4. Особенности криптосистемы Эль-Гамала . . . . .	7
2. Алгоритмы решения задачи дискретного логарифмирования . . . . .	9
2.1. В произвольной мультипликативной группе . . . . .	9
2.2. В кольце вычетов по простому модулю . . . . .	9
2.3. Алгоритмы с экспоненциальной сложностью . . . . .	10
2.4. Субэкспоненциальные алгоритмы . . . . .	12
3. Американский стандарт кодирования - ASCII . . . . .	14
4. Анализ DES, ГОСТ 28147-89, Crypto03, El-Gamal . . . . .	16
Список используемых источников . . . . .	20

## ВВЕДЕНИЕ

В настоящее время в вузах Российской Федерации базовые стандарты обучения для ряда специальностей включают в себя разделы, связанные с изучением методов и средств защиты информации. Для успешного освоения данных тем необходимо понимание принципов и знание основных элементов криптографического преобразования информации.

В Интернете можно найти десятки описаний лабораторных работ, посвященных криптографической системе Эль Гамала [1 – 3]. К сожалению, подавляющее большинство из них содержат задания и примеры реализации схемы Эль Гамала без учета особенностей длинной арифметики, не требуя обоснований алгоритмов и использования обучающих программ, не затрагивая вопросы криптоанализа.

Известно несколько компьютерных обучающих программ, позволяющих быстро и достаточно полно ознакомиться с алгоритмами шифрования и расшифрования данных, используемыми в традиционных симметричных и современных асимметричных криптосистемах. К сожалению, эти программы, представленные в сети Интернет, не сопровождаются исходными текстами, ограничиваются краткой справочной информацией и содержат большое число ошибок и недочетов. В связи с этим и было принято решение: разработать алгоритм и реализовать свою электронную обучающую программу для изучения криптосистемы Эль Гамала, а также разработать сценарий лабораторной работы с использованием этой программы. Предлагаемый вариант лабораторной работы призван преодолеть указанные недостатки.

## ПОСТАНОВКА ЗАДАЧИ

1. Провести анализ криптографического алгоритма Эль Гамала.
2. Разработать сценарий выполнения лабораторной работы по изучению алгоритма Эль Гамала.
3. Разработать и реализовать обучающую компьютерную программу "El-Gamal\_Tutor".

# 1. ОБЩАЯ ИНФОРМАЦИЯ О КРИПТОСИСТЕМЕ ЭЛЬ-ГАМАЛЯ

Схема Эль-Гамала (Elgamal) — криптосистема с открытым ключом, основанная на трудности вычисления дискретных логарифмов в конечном поле. Криптосистема включает в себя алгоритм шифрования и алгоритм цифровой подписи. Схема Эль-Гамала лежит в основе бывших стандартов электронной цифровой подписи в США (DSA) и России (ГОСТ Р 34.10-94, ГОСТ Р 34.10-2001). Схема была предложена Тахером Эль-Гамалем в 1985 году. Эль-Гамаль разработал один из вариантов алгоритма Диффи-Хеллмана. Он усовершенствовал систему Диффи-Хеллмана и получил два алгоритма, которые использовались для шифрования и для обеспечения аутентификации. В отличие от RSA алгоритм Эль-Гамала не был запатентован и, поэтому, стал более дешевой альтернативой, так как не требовалась оплата взносов за лицензию. Считается, что алгоритм попадает под действие патента Диффи-Хеллмана.

Криптографические системы с открытым ключом используют так называемые односторонние функции, которые обладают следующим свойством:

- Если известно  $x$ , то  $f(x)$  вычислить относительно просто
- Если известно  $y = f(x)$ , то для вычисления  $x$  нет простого (эффективного) пути.

Под односторонностью понимается не теоретическая однонаправленность, а практическая невозможность вычислить обратное значение, используя современные вычислительные средства, за обозримый интервал времени.

В основу криптографической системы Эль-Гамала положена сложность задачи дискретного логарифмирования в конечном поле. Для шифрования используется операция возведения в степень по модулю большого числа. Для дешифрования за разумное время необходимо уметь вычислять дискретный логарифм в конечном поле по простому модулю, что является вычислительно трудной задачей.

В криптографической системе с открытым ключом каждый участник

располагает как открытым ключом (англ. public key), так и закрытым ключом (англ. private key). В криптографической системе Эль-Гамала открытый ключ состоит из тройки чисел, а закрытый ключ состоит из одного числа. Каждый участник создаёт свой открытый и закрытый ключ самостоятельно. Закрытый ключ каждый из них держит в секрете, а открытые ключи можно сообщать кому угодно или даже публиковать их.

## 1.1. Алгоритм создания открытого и закрытого ключей

Ключи в схеме Эль-Гамала генерируются следующим образом:

1. Генерируется случайное простое число  $p$ .
2. Выбирается целое число  $g$  — первообразный корень  $p$ .
3. Выбирается случайное целое число  $x$ , такое, что  $1 < x < p$ .
4. Вычисляется  $y = g^x \bmod p$ .
5. Открытым ключом является тройка  $(p, g, y)$ , закрытым ключом — число  $x$ .

## 1.2. Шифрование и расшифрование

Предположим, пользователь А хочет послать пользователю Б сообщение. Сообщениями являются целые числа в интервале от 0 до  $p - 1$ . Алгоритм для шифрования:

1. Взять открытый ключ пользователя Б
2. Взять открытый текст  $M$
3. Выбрать сессионный ключ — случайное целое число  $k$  такое, что  $1 < k < p - 1$
4. Зашифровать сообщение с использованием открытого ключа пользователя Б, то есть вычислить числа:  $a = g^k \bmod p$ , и  $b = y^k M \bmod p$ .

Алгоритм для расшифрования:

1. принять зашифрованное сообщение  $(a, b)$  от пользователя А
2. Взять свой закрытый ключ  $M$
3. Применить закрытый ключ для расшифрования сообщения:  $M = b(a^x)^{-1} \bmod p$
4. При этом нетрудно проверить, что
$$(a^x)^{-1} \equiv g^{-kx} \pmod{p}, \text{ и поэтому}$$
$$b(a^x)^{-1} \equiv (y^k M)g^{-xk} \equiv (g^{xk} M)g^{-xk} \equiv M \pmod{p}.$$

### 1.3. Дешифрование

Дешифрование - получение открытых данных по зашифрованным в условиях, когда алгоритм расшифрования и его секретные параметры не являются полностью известными и расшифрование не может быть выполнено обычным путем. Алгоритм для дешифрования криптосистемы Эль-Гамала:

1. Перехватить зашифрованное сообщение  $(a, b)$ .
2. Взять открытый ключ  $(p, g, y)$
3. Решить относительно  $x$  уравнение  $y \equiv g^x \pmod{p}$
4. Расшифровать сообщение по формуле  $M = b(a^x)^{-1} \bmod p$

Собственно, самый главный вопрос из этого алгоритма – как по данным  $(p, g, y)$  найти  $x$ . Эта задача называется задачей дискретного логарифмирования [2].

### 1.4. Особенности криптосистемы Эль-Гамала

- Криптосистема асимметричная (двухключевая).
- Блочная, с длиной блока открытого текста меньше или равной длине открытого (публичного) ключа.

- Длина открытого и закрытого ключей, по современным представлениям, 2048 бит или более.
- Используется лишь один метод шифрования – метод аналитических преобразований.
- Базируется на вычислительно трудной задаче дискретного логарифмирования.
- Предоставляет возможность реализации электронной подписи.

## 2. АЛГОРИТМЫ РЕШЕНИЯ ЗАДАЧИ ДИСКРЕТНОГО ЛОГАРИФМИРОВАНИЯ

### 2.1. В произвольной мультипликативной группе

Разрешимости и решению задачи дискретного логарифмирования в произвольной конечной абелевой группе посвящена статья J. Buchmann, M. J. Jacobson и E. Teske [8]. В алгоритме используется таблица, состоящая из  $O(\sqrt{|g|})$  пар элементов, и выполняется  $O(\sqrt{|g|})$  умножений. Данный алгоритм медленный и не пригоден для практического использования. Для конкретных групп существуют свои, более эффективные, алгоритмы.

### 2.2. В кольце вычетов по простому модулю

Рассмотрим сравнение

$$a^x \equiv b \pmod{p} \quad (1)$$

где  $p$  — простое,  $b$  не делится на  $p$ . Если  $a$  является образующим элементом группы  $\mathbb{Z}/p\mathbb{Z}$ , то сравнение (1) имеет решение при любых  $b$ . Такие числа  $a$  называются ещё первообразными корнями, и их количество равно  $\phi(p) = p - 1$ , где  $\phi$  — функция Эйлера. Решение сравнения (1) можно находить по формуле:

$$x \equiv \sum_{i=1}^{p-2} (1 - a^i)^{-1} b^i \pmod{p} \quad (2)$$

Однако, сложность вычисления по этой формуле хуже, чем сложность полного перебора.

Следующий алгоритм [3] имеет сложность  $O(\sqrt{p} \cdot \log p)$ . Алгоритм

1. Присвоить  $H := \lfloor \sqrt{p} \rfloor + 1$
2. Вычислить  $c = a^H \pmod{p}$
3. Составить таблицу значений  $c^u \pmod{p}$  для  $1 \leq u \leq H$  и отсортировать её.



4. Составить таблицу значений  $b \cdot a^v \bmod p$  для  $0 \leq v \leq H$  и отсортировать её.
5. Найти общие элементы в таблицах. Для них  $c^u \equiv b \cdot a^v \pmod{p}$  откуда  $a^{H \cdot u - v} \equiv b \pmod{p}$
6. Выдать  $H \cdot u - v$ .

Существует также множество других алгоритмов для решения задачи дискретного логарифмирования в поле вычетов [3]. Их принято разделять на экспоненциальные и субэкспоненциальные. Полиномиального алгоритма для решения этой задачи пока не найдено.

### 2.3. Алгоритмы с экспоненциальной сложностью

Алгоритм Гельфонда-Шенкса (алгоритм больших и малых шагов, baby-step giant-step) был предложен независимо советским математиком Александром Гельфондом в 1962 году и Дэниэлем Шенксом в 1972 году. Относится к методам встречи посередине. Идея алгоритма состоит в выборе оптимального соотношения времени и памяти, а именно в усовершенствованном поиске показателя степени.

Пусть задана циклическая группа  $G$  порядка  $n$ , генератор группы  $\alpha$  и некоторый элемент группы  $\beta$ . Задача сводится к нахождению целого числа  $x$ , для которого выполняется  $\alpha^x = \beta \bmod m$ .

Алгоритм Гельфонда — Шенкса основан на представлении  $x$  в виде  $x = i \cdot t - j$ , где  $t = \lfloor \sqrt{n} \rfloor + 1$ , и переборе  $1 \leq i \leq t$  и  $0 \leq j \leq t$ . Ограничение на  $i$  и  $j$  следует из того, что порядок группы не превосходит  $t$ , а значит указанные диапазоны достаточны для получения всех возможных из полуинтервала  $[0; t)$ . Такое представление равносильно равенству

$$\alpha^{im} = \beta \alpha^j \tag{3}$$

Алгоритм предварительно вычисляет  $\alpha^{im}$  для разных значений  $i$  и сохраняет их в структуре данных, позволяющей эффективный поиск, а затем перебирает всевозможные значения  $j$  и проверяет, если  $\beta \alpha^j$  соответствует какому-то значению  $i$ . Таким образом находятся индексы

$i$  и  $j$ , которые удовлетворяют соотношению (3) и позволяют вычислить значение  $x = i \cdot t - j$ .

Алгоритму Гельфонда — Шенкса требуется  $O(n)$  памяти. Возможно выбрать меньшее  $t$  на первом шаге алгоритма, но это увеличивает время работы программы до  $O(n/t)$ .



Рис. 1: Мартин Хеллман

Другим методом дискретного логарифмирования является алгоритм Сильвера-Полига-Хеллмана. Он работает, если известно разложение числа  $p - 1 = \prod_{i=1}^s q_i^{\alpha_i}$  на простые множители. Сложность оценивается как  $O(\sum_{i=1}^s \alpha_i (\log p + q_i))$ . Если множители, на которые раскладывается  $p - 1$ , достаточно маленькие, то алгоритм чрезвычайно эффективен. Это необходимо учитывать в выборе параметров при разработке криптографических схем, основанных на вычислительной сложности дискретного логарифмирования, иначе схема будет ненадёжной.

Для применения алгоритма Сильвера-Полига-Хеллмана необходимо знать разложение  $p - 1$  на множители. В общем случае задача факторизации — достаточно трудоёмкая, однако если делители числа — небольшие, то это число можно быстро разложить на множители даже методом последовательного деления. Таким образом, в тех случаях, когда эффективен алгоритм Сильвера-Полига-Хеллмана, необходимость факторизации не усложняет задачу.

Ещё одним методом дискретного логарифмирования является  $\rho$ -метод Полларда, который был предложен Джоном Поллардом в 1978 году, основные идеи алгоритма похожи на  $\rho$ -алгоритм Полларда для

факторизации чисел. Условием работы  $\rho$ -метода Полларда является простота порядка группы, порождённой основанием  $a$  дискретного логарифма по модулю  $p$ .

Алгоритм имеет эвристическую оценку сложности  $O(p^{\frac{1}{2}})$ . По сравнению с другими методами дискретного логарифмирования  $\rho$ -метод Полларда является менее затратным как по отношению к вычислительным операциям, так и по отношению к затрачиваемой памяти. Например, при достаточно больших значениях числа  $p$  данный алгоритм является вычислительно менее сложным, чем алгоритм COS и алгоритм Адлемана. С другой стороны, условие работы алгоритма накладывает серьёзные ограничения на его использование.

## 2.4. Субэкспоненциальные алгоритмы

Данные алгоритмы имеют сложность, оцениваемую как  $O(\exp(c(\log p \log p \log p)^d))$  арифметических операций, где  $c$  и  $0 \leq d \leq 1$  — некоторые константы. Эффективность алгоритма во многом зависит от близости  $c$  к 1 и  $d$  — к 0.

Алгоритм Адлемана [9] появился в 1979 году. Это был первый субэкспоненциальный алгоритм дискретного логарифмирования. На практике он всё же недостаточно эффективен. В этом алгоритме  $d = \frac{1}{2}$ .

Алгоритм COS [3] был предложен в 1986 году математиками Копперсмитом (Don Coppersmith), Одлышко (Andrew Odlyzko) и Шреппелем (Richard Schroepel). В этом алгоритме константа  $c = 1$ ,  $d = \frac{1}{2}$ . В 1991 году с помощью этого метода было проведено логарифмирование по модулю  $p \approx 10^{58}$ . В 1997 году Вебер [3] провел дискретное логарифмирование по модулю  $p \approx 10^{85}$  с помощью некоторой версии данного алгоритма. Экспериментально показано, что при  $p \leq 10^{90}$  алгоритм COS лучше решета числового поля.

Дискретное логарифмирование при помощи решета числового поля [3] было применено к дискретному логарифмированию позднее, чем к факторизации чисел. Первые идеи появились в 1990-х годах. Алгоритм, предложенный Д. Гордоном в 1993 году [3], имел эвристическую сложность  $O(\exp(3^{3/2}(\log p \log p \log p)^{\frac{1}{3}}))$ , но оказался достаточно непрактичным.

Позднее было предложено множество различных улучшений данного алгоритма. Было показано, что при  $p \geq 10^{100}$  решето числового поля быстрее, чем COS [3]. Современные рекорды в дискретном логарифмировании получены именно с помощью этого метода.

Наилучшими параметрами в оценке сложности на данный момент является  $c = (92 + 26\sqrt{13})^{1/3}/3 \approx 1,902$ ,  $d = \frac{1}{3}$ . Для чисел специального вида результат можно улучшить. В некоторых случаях можно построить алгоритм, для которого константы будут  $c \approx 1,00475$ ,  $d = \frac{2}{5}$ . За счёт того, что константа  $c$  достаточно близка к 1, подобные алгоритмы могут обогнать алгоритм с  $d = \frac{1}{3}$ .

Другая возможность эффективного решения задачи вычисления дискретного логарифма связана с квантовыми вычислениями. Теоретически доказано, что с их помощью дискретный логарифм можно вычислить за полиномиальное время. В любом случае, если полиномиальный алгоритм вычисления дискретного логарифма будет реализован, это будет означать практическую непригодность криптосистем на его основе [3].

### 3. АМЕРИКАНСКИЙ СТАНДАРТ КОДИРОВАНИЯ - ASCII

ASCII (англ. American Standard Code for Information Interchange) — американская стандартная 7-битная кодировочная таблица для печатных символов и некоторых специальных кодов, использующаяся в компьютерной коммуникации. ASCII представляет собой кодировку для представления десятичных цифр, латинского и национального алфавитов, знаков препинания и управляющих символов.

Таблица была разработана и стандартизована в 1963 году. Множество современных кодировок и стандартов (UTF-8, Win-1251, КОИ-8) являются расширениями стандарта ASCII. В СССР стандарт был утвержден в 1987 году в виде таблицы международной ссылочной версии кода КОИ-7 НО ГОСТ 27463-87 (СТ СЭВ 356-86) «Системы обработки информации. 7-битные кодированные наборы символов» [?].

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	&#032;	Space	64	40	100	&#064;	@	96	60	140	&#096;	`
1	1	001	Start of Header	33	21	041	&#033;	!	65	41	101	&#065;	A	97	61	141	&#097;	a
2	2	002	Start of Text	34	22	042	&#034;	"	66	42	102	&#066;	B	98	62	142	&#098;	b
3	3	003	End of Text	35	23	043	&#035;	#	67	43	103	&#067;	C	99	63	143	&#099;	c
4	4	004	End of Transmission	36	24	044	&#036;	\$	68	44	104	&#068;	D	100	64	144	&#100;	d
5	5	005	Enquiry	37	25	045	&#037;	%	69	45	105	&#069;	E	101	65	145	&#101;	e
6	6	006	Acknowledgment	38	26	046	&#038;	&	70	46	106	&#070;	F	102	66	146	&#102;	f
7	7	007	Bell	39	27	047	&#039;	'	71	47	107	&#071;	G	103	67	147	&#103;	g
8	8	010	Backspace	40	28	050	&#040;	(	72	48	110	&#072;	H	104	68	150	&#104;	h
9	9	011	Horizontal Tab	41	29	051	&#041;	)	73	49	111	&#073;	I	105	69	151	&#105;	i
10	A	012	Line feed	42	2A	052	&#042;	*	74	4A	112	&#074;	J	106	6A	152	&#106;	j
11	B	013	Vertical Tab	43	2B	053	&#043;	+	75	4B	113	&#075;	K	107	6B	153	&#107;	k
12	C	014	Form feed	44	2C	054	&#044;	,	76	4C	114	&#076;	L	108	6C	154	&#108;	l
13	D	015	Carriage return	45	2D	055	&#045;	-	77	4D	115	&#077;	M	109	6D	155	&#109;	m
14	E	016	Shift Out	46	2E	056	&#046;	.	78	4E	116	&#078;	N	110	6E	156	&#110;	n
15	F	017	Shift In	47	2F	057	&#047;	/	79	4F	117	&#079;	O	111	6F	157	&#111;	o
16	10	020	Data Link Escape	48	30	060	&#048;	0	80	50	120	&#080;	P	112	70	160	&#112;	p
17	11	021	Device Control 1	49	31	061	&#049;	1	81	51	121	&#081;	Q	113	71	161	&#113;	q
18	12	022	Device Control 2	50	32	062	&#050;	2	82	52	122	&#082;	R	114	72	162	&#114;	r
19	13	023	Device Control 3	51	33	063	&#051;	3	83	53	123	&#083;	S	115	73	163	&#115;	s
20	14	024	Device Control 4	52	34	064	&#052;	4	84	54	124	&#084;	T	116	74	164	&#116;	t
21	15	025	Negative Ack.	53	35	065	&#053;	5	85	55	125	&#085;	U	117	75	165	&#117;	u
22	16	026	Synchronous idle	54	36	066	&#054;	6	86	56	126	&#086;	V	118	76	166	&#118;	v
23	17	027	End of Trans. Block	55	37	067	&#055;	7	87	57	127	&#087;	W	119	77	167	&#119;	w
24	18	030	Cancel	56	38	070	&#056;	8	88	58	130	&#088;	X	120	78	170	&#120;	x
25	19	031	End of Medium	57	39	071	&#057;	9	89	59	131	&#089;	Y	121	79	171	&#121;	y
26	1A	032	Substitute	58	3A	072	&#058;	:	90	5A	132	&#090;	Z	122	7A	172	&#122;	z
27	1B	033	Escape	59	3B	073	&#059;	;	91	5B	133	&#091;	[	123	7B	173	&#123;	{
28	1C	034	File Separator	60	3C	074	&#060;	<	92	5C	134	&#092;	\	124	7C	174	&#124;	
29	1D	035	Group Separator	61	3D	075	&#061;	=	93	5D	135	&#093;	]	125	7D	175	&#125;	}
30	1E	036	Record Separator	62	3E	076	&#062;	>	94	5E	136	&#094;	^	126	7E	176	&#126;	~
31	1F	037	Unit Separator	63	3F	077	&#063;	?	95	5F	137	&#095;	_	127	7F	177	&#127;	Del

Рис. 2: ASCII коды

В криптографических программах ASCII используется для



преобразования символов текста в цифры, чтобы текст было возможно представить в виде чисел и совершать над ним криптографические преобразования. Например: большим буквам английского алфавита соответствуют значения с 97 по 122.

Поскольку на подавляющем большинстве современных компьютеров минимально адресуемой единицей памяти является байт (размером в 8 бит), там используются 8-битные, а не 7-битные символы. Обычно символ ASCII расширяют до 8 бит, подставляя нулевой бит в качестве старшего. Таким образом, каждый преобразованный в число символ занимает ровно один байт. Уменьшение размера одного символа для криптосистем главным образом означает возможность передать большой шифротекст в одном блоке при неизменной длине ключа.

## 4. АНАЛИЗ DES, ГОСТ 28147-89, CRYPTO03, EL-GAMAL

Перед началом написания программы “El-Gamal\_Tutor” были изучены другие приложения для обучения криптосистемам. Одними из них были: DES, ГОСТ 28147-89, Crypto-03 и El-Gamal.

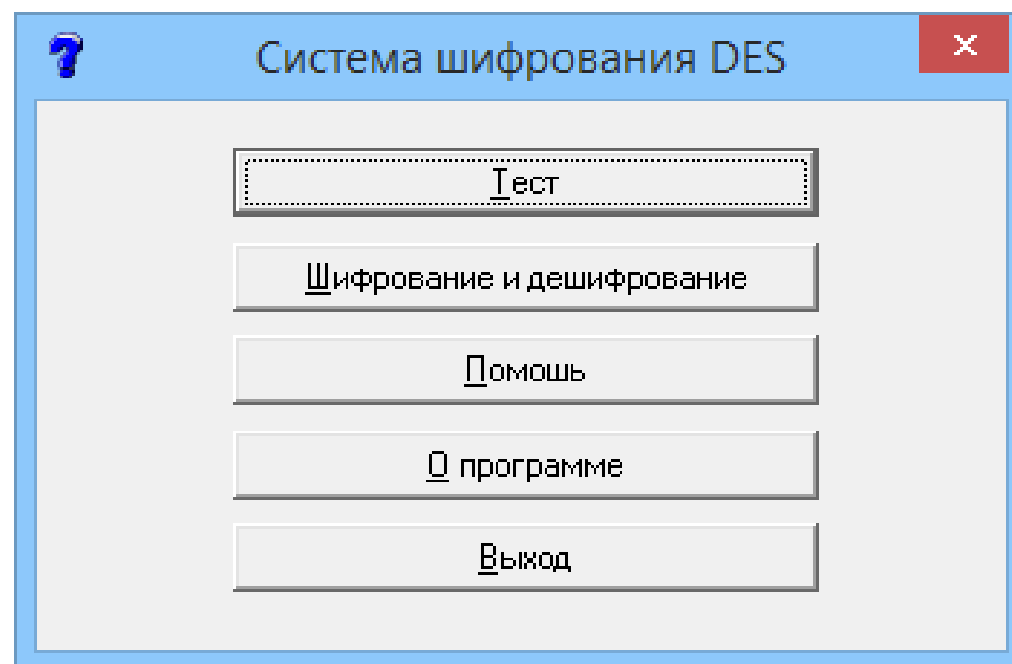


Рис. 3: Главное меню программы "Система шифрования DES"

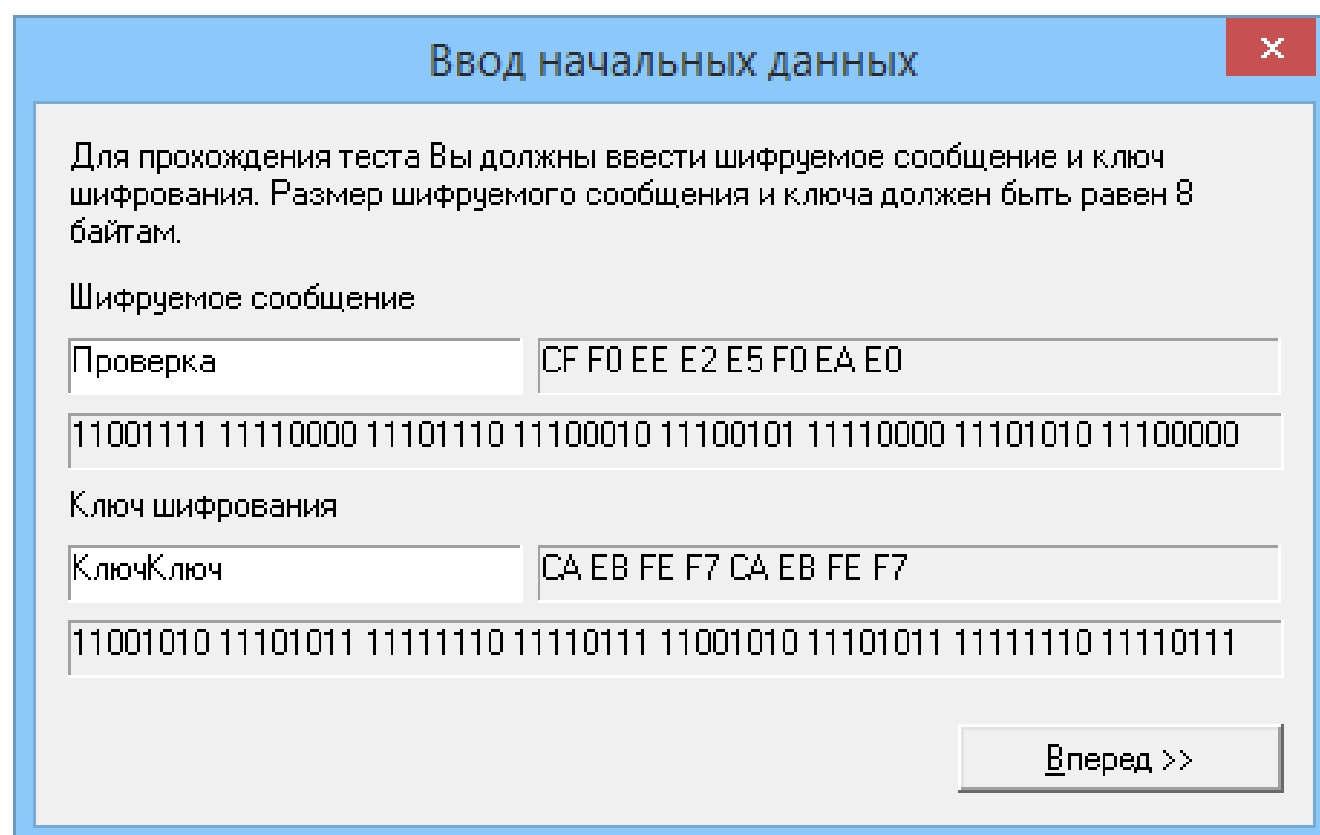


Рис. 4: Ввод начальных данных

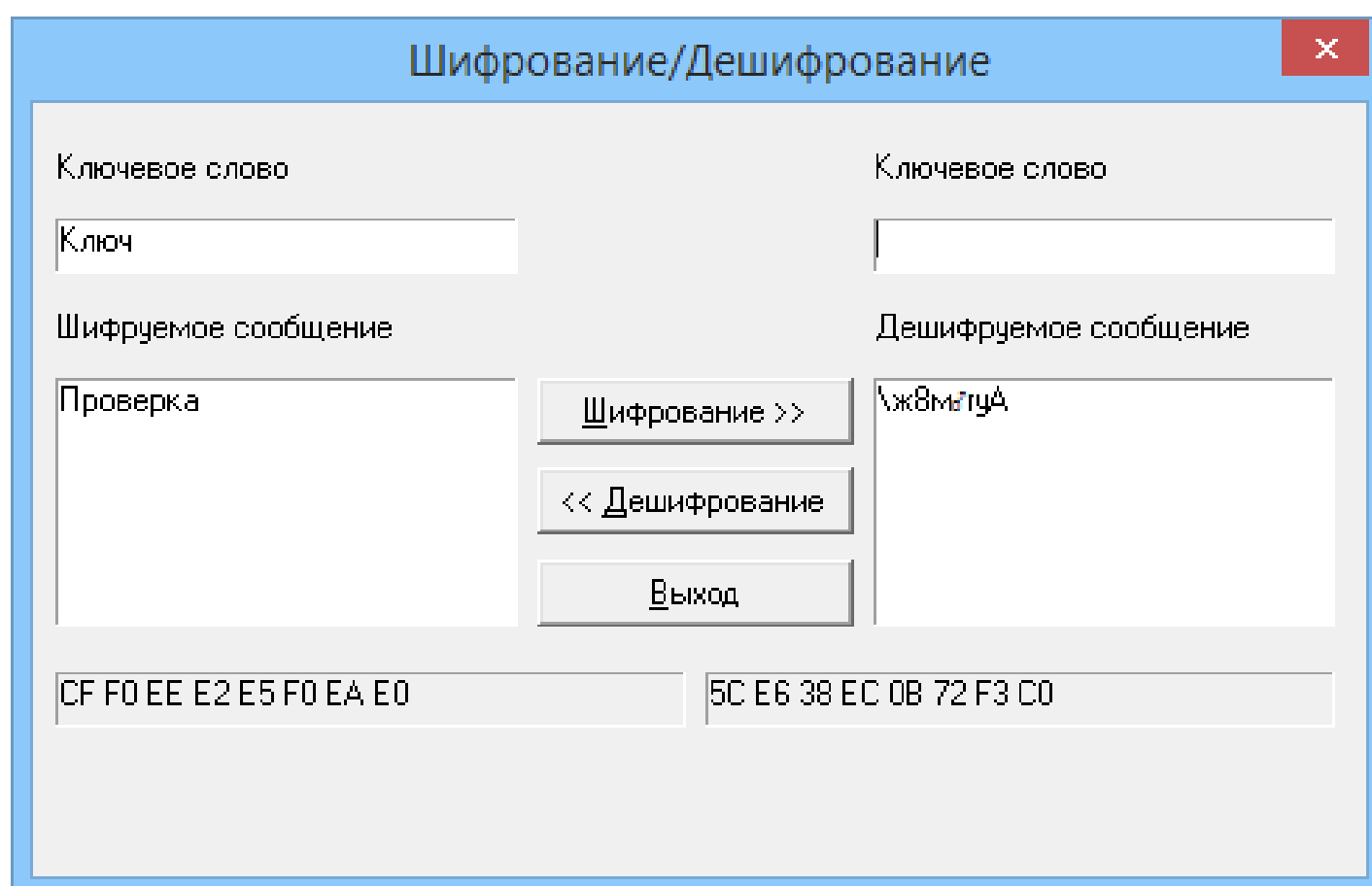


Рис. 5: Шифрование/Расшифрование [4]

В данном случае в программе режим называется неправильно, так как на самом деле вместо дешифрования происходит расшифрование.

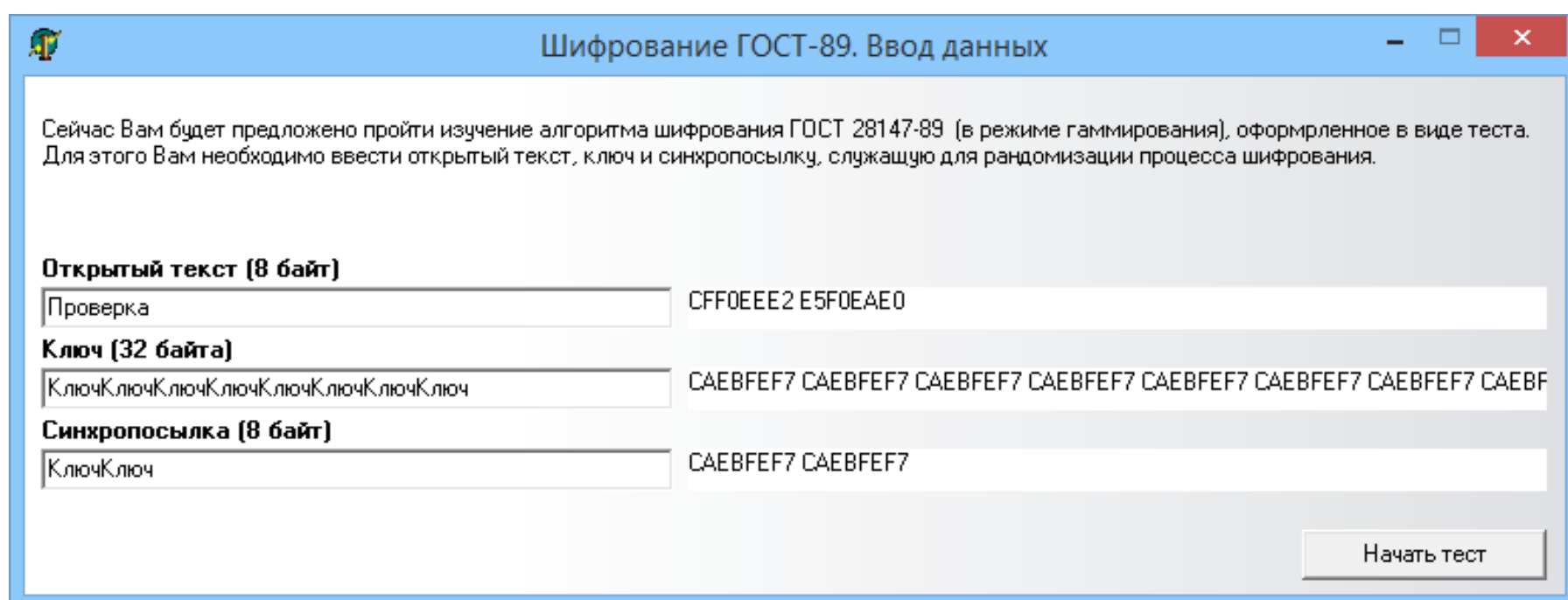


Рис. 6: Ввод данных в обучающей программе ГОСТ 28147-89



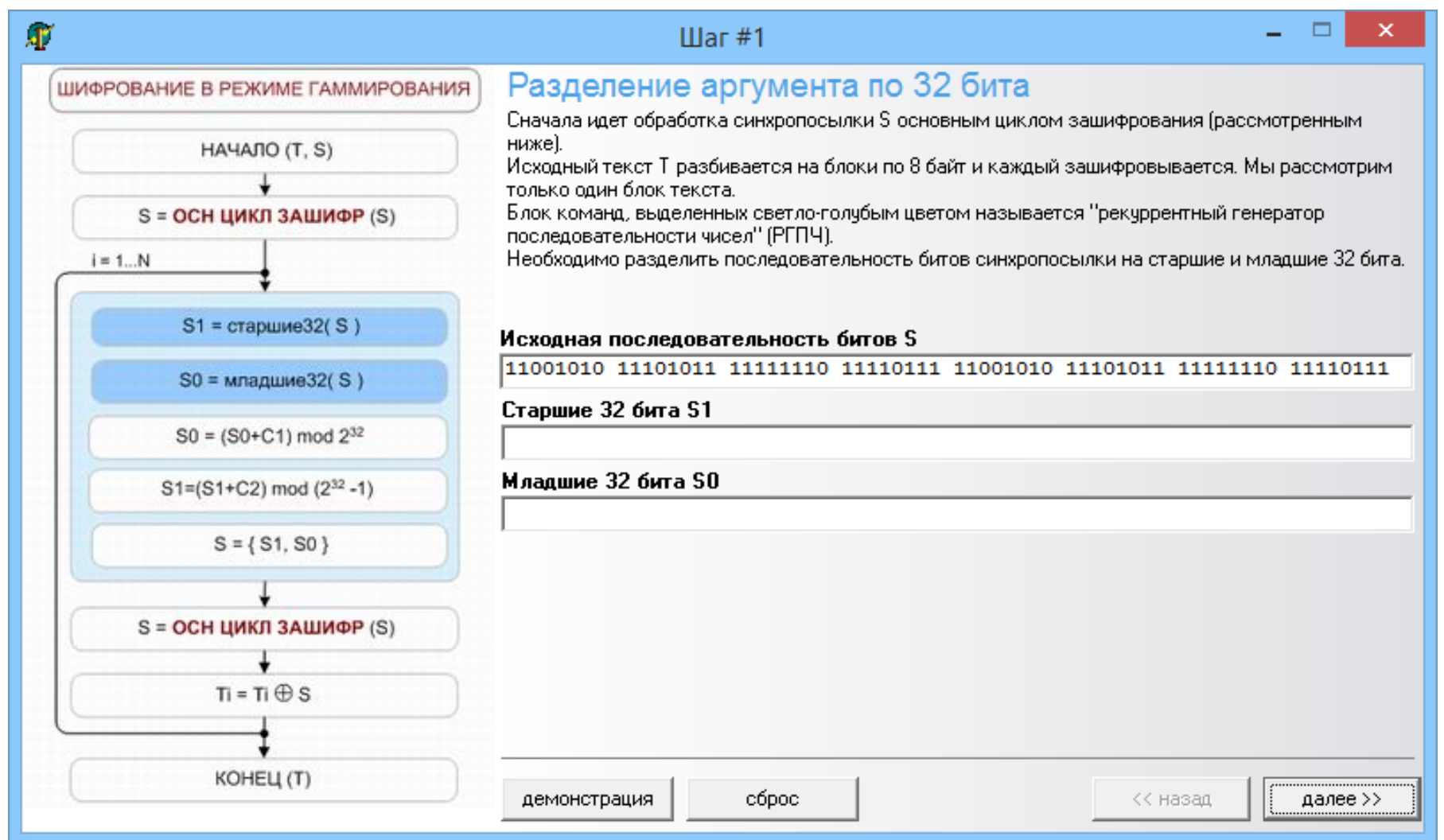


Рис. 7: Первый шаг обучения ГОСТ 28147-89 [6]

Вычисляю:  $X^N \pmod{M}$

X =	23452645
N =	4352345
M =	64325234545

Результат:

6594634925

Очистить окна
Вычислить
Выход

Рис. 9: Вычисление  $X^N \pmod{M}$

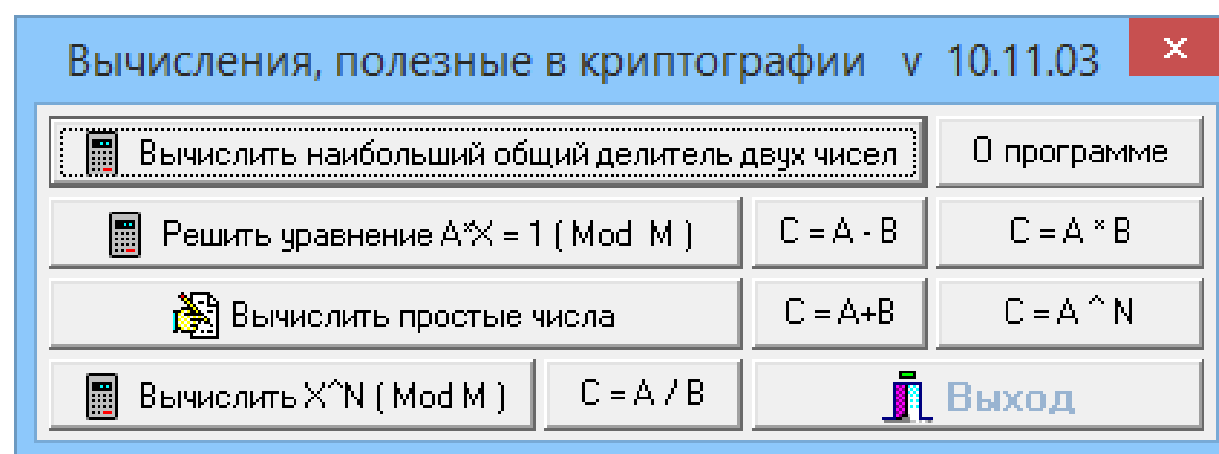


Рис. 8: Основная форма программы вычислений, полезных в криптографии v.10.11.2003 – Crypto03 [4]

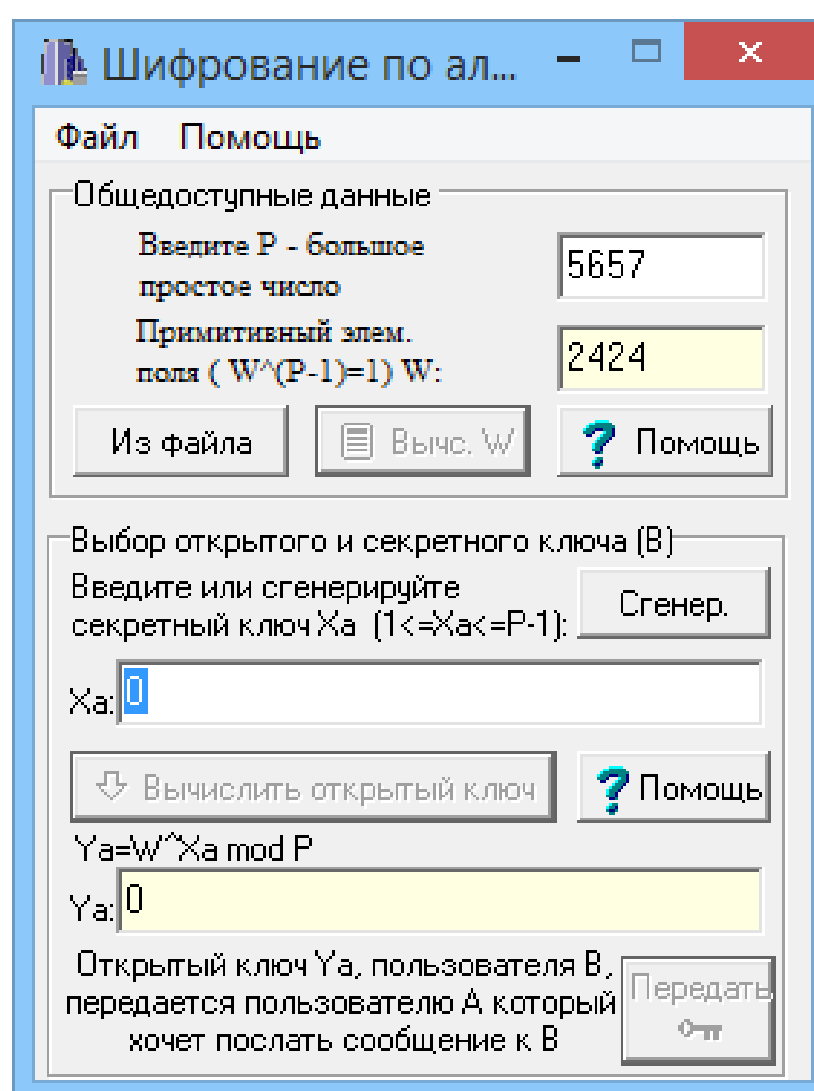


Рис. 10: Основная форма обучающей программы El-Gamal [5]

После рассмотрения всех этих программ, была составлена картина того, как должна выглядеть будущая электронная обучающая программа El-Gamal\_Tutor.

# **Список используемых источников. Шрифт поправлю, Борис Николаевич!**

1. **Гилбарг, Д.** Эллиптические дифференциальные уравнения с частными производными второго порядка / Д. Гилбарг, П. Трудингер. — М. : Наука, 1989. — 464 с.
2. **Ильин, В. А.** О рядах Фурье по фундаментальным системам функций оператора Бельтрами/В. А. Ильин // Дифференц. уравнения. — 1969. — Т. 5, № 11. — С. 1940–1978.
3. **Ильин, В. А.** Некоторые свойства регулярного решения уравнения Гельмгольца в плоской области / В. А. Ильин // Мат. заметки. — 1974. — Т. 15, № 6. — С. 885–890.
4. **Ильин, В. А.** Об одном обобщении формулы среднего значения для регулярного решения уравнения Шредингера / В. А. Ильин, Е. И. Моисеев // ИПМ АН СССР, 1977. — С. 157–166.
5. **Ильин, В. А.** Формула среднего значения для присоединенных функций оператора Лапласа / В. А. Ильин, Е. И. Моисеев // Дифференц. уравнения. — 1981. — Т. 17, № 10. — С. 1908–1910.
6. **Моисеев, Е. И.** Формула среднего для собственных функций эллипти-

ческого самосопряженного оператора второго порядка / Е. И. Моисеев // Докл. АН СССР. — 1971. — Т. 197, № 3. — С. 524–525.

7. **Моисеев, Е. И.** Асимптотическая формула среднего значения для регулярного решения дифференциального уравнения / Е. И. Моисеев // Дифференц. уравнения. — 1980. — Т. 16, № 5. — С. 827–844.
8. **Хелгасон, С.** Дифференциальная геометрия и симметрические пространства / С. Хелгасон. — М. : Мир, 1964. — 534 с.
9. **Иванов, Л. А.** О некоторых свойствах оператора Бельтрами в римановой метрике / Л. А. Иванов, И. П. Половинкин // Докл. РАН. — 1999. — Т. 365, № 3. — С. 306–309.
10. **Йон, Ф.** Плоские волны и сферические средние / Ф. Йон. — М. : Иностр. лит., 1958. — 158 с.
11. **Бицадзе, А. В.** К теории уравнений смешанного типа в многомерных областях / А. В. Бицадзе, А. М. Нахушев // Дифференц. уравнения. — 1974. — Т. 10, № 12. — С. 2184–2191.
12. **Гельфанд, И. М.** Обобщенные функции и действия над ними / И. М. Гельфанд, Г. Е. Шилев. — М. : Физматлит, 1958. — 440 с.
13. **Хермандер, Л.** Анализ линейных дифференциальных операторов с частными производными. Т. 1 / Л. Хермандер. — М. : Мир, 1986. — 464 с.
14. **Мешков, В. З.** К свойствам решений линейных уравнений в частных производных / В. З. Мешков, И. П. Половинкин // Черноземный альманах научных исследований. — Сер. Фундамент. математика. — 2007. — Вып. 1(5). — С. 3–11.
15. **Мешков, В. З.** Разностная формула среднего значения для двумерного линейного гиперболического уравнения третьего порядка / В. З. Мешков, И. П. Половинкин, М. В. Половинкина, Ю. Д. Ермакова, С.

А. Рабеев // ВЕСТНИК ВГУ. СЕРИЯ: ФИЗИКА. МАТЕМАТИКА. — 2015. — № 3

16. **Половинкин, И.П.** К свойствам решений линейных уравнений в частных производных / Половинкин И.П. // Вестник Челябинского государственного университета. Математика. Механика. Информатика. Выпуск 12. — 2010. — № 23(204) — С. 59–66.
17. **Половинкин, И.П.** О получении новых формул среднего значения для линейных дифференциальных уравнений с постоянными коэффициентами / Половинкин И.П., Мешков В.З. // Дифференциальные уравнения. — 2011 — Т. 47, № 12 — С. 1724–1791.
18. **Половинкин, И.П.** Дополнения к свойствам средних значений решений линейных дифференциальных уравнений с постоянными коэффициентами / Половинкин И.П., Мешков В.З. // Дифференциальные уравнения. — 2011 — Т. 47, № 11 — С. 1669 – 1671.