АЛГОРИТМ СЖАТИЯ АЛФАВИТНОЙ ИНФОРМАЦИИ С АДАПТАЦИЕЙ ДЛЯ КРИПТОСИСТЕМ

В.Г. ПОТЁМКИН
Н.И. КОРСУНОВ

Белгородский государственный технологический университет

e-mail: *skf-bgtu@yandex.ru*

В статье предлагается эффективный алгоритм сжатия для защиты от перехвата передаваемой информации. Проанализированы алгоритмы, позволяющие сжимать информацию, и выявлены их недостатки, состоящие в том, что в процессе сжатия они неадекватно воспринимают последовательности повторяющихся символов, оперируют с алфавитами недостаточной мощности. Для устранения недостатков вводится новая система правил сжатия, для реализации которой приведен алгоритм. Проведено сравнение предложенного алгоритма с известным.

Ключевые слова: информация, криптостойкость, сжатие, несанкционированный доступ, избыточность, кодирование.

Основным современным подходом в защите информации является сокрытие или изменение ее смысла посредством криптографических методов [1]. Данные методы, изменяя свойства информации, преобразуют ее в шифр. Критерием оценки защищенности информации, содержащейся в шифре, является криптостойкость. Этот критерий представляет собой множество, объединяющее те свойства, от значений которых зависит стойкость шифра к криптоанализу [1]. Среди этих свойств выделим *частотный спектр появления символов*, которое отображает вероятность появления символов в данной информации. Такой выбор связан в первую очередь с тем, что частотный спектр появления символов не зависит от алгоритма шифрования. Это вызвано тем, что практически все известные шифры смещают позиции символов в данных и подменяют их истинное значение. При этом скрывается смысл передаваемой информации, но другие её количественные характеристики остаются неизменными. Смещения позиций символа и значения кода символа при подмене имеют некоторые конкретные значения, вследствие чего на открытые данные по определенному алгоритму происходит наложение некоторой маски скрывающей их суть. При шифровании количество символов на входе алгоритма равно их количеству на выходе. Структура информации в данных не меняется, что приводит к повторению смысловой нагрузки значения символов, алфавита открытого текста и алфавита шифра.

Анализируя достаточно длинный, зашифрованный текст, можно по частотам появления символов произвести восстановление исходного текста. При этом совсем не обязательно анализировать все буквы слов текста. Остальные можно подобрать по смыслу, так как естественные языки обладают большой избыточностью. Для русского языка, например, буква «о» появляется в 45 раз чаще буквы «ф» и в 30 раз чаще буквы «э». Относительная частота появления пробела или знака препинания в русском языке составляет 0,174. Данную проблему легко понять на примере следующей аллегории. Маска, скрывая лик, полностью повторяет его анатомические особенности, но по косвенным признакам (черты, пропорции) можно без труда идентифицировать владельца.

На практике был проведен ряд экспериментов, целью которых было выделение из различных алфавитных данных частотного спектра появления символов. Результат одного из экспериментов проиллюстрирован на рис. 1.

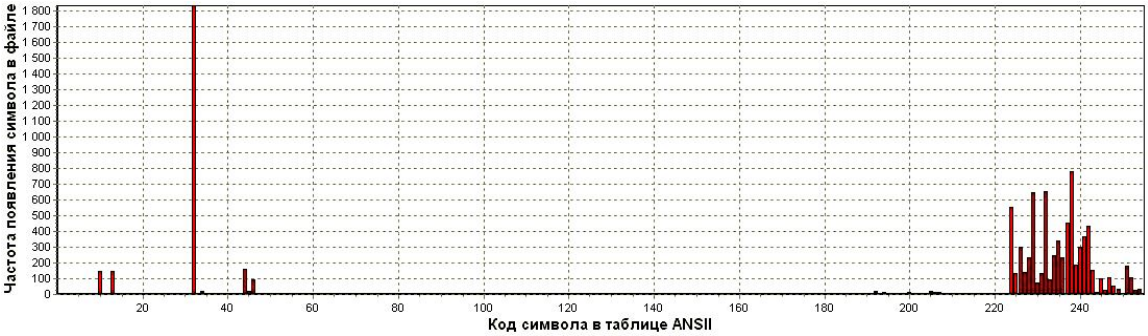

Рис. 1. Частота встречаемости символов в текстовом файле, размером 10 кбайт

Как следует из эксперимента, в спектре отчетливо прослеживаются всплески частоты появления определенных символов. Самый часто встречающийся символ данного текста – знак пробела с кодом 32 в таблице ANSII IBM cp866 [2]. Следовательно в шифре символ с такой же смысловой нагрузкой, но другим значением будет так же часто встречаться. Зная алгоритм шифрования, который, как правило, не является секретным, можно без труда составить структуру текста [1]. Средняя частота встречаемости других символов имеющих всплеск частоты (рис. 1) – это буквы «о», «е», «а», «и», «н», в большинстве текстов лежит в одних и тех же диапазонах.

Из этого следует, что для несанкционированного доступа имеются следующие преимущества:

1) представление о структуре текста (по расположению знаков пробела);

2) виду того, что информация на естественном языке обладает избыточностью, не имеет смысла расшифровывать все слова в предложении, их можно подобрать по смыслу;

3) так как построение слов и букв в них придерживается определенных общеизвестных правил, то зная истинное значение ключевых букв в словах, можно без труда извлечь из шифра скрытую информацию.

Поэтому один из основных подходов в повышении криптостойкости шифра, является избавление его от избыточности, которая характерна для информации, воспринимаемой и воспроизводимой человеком. Для избавления информации от избыточности используется ее сжатие или иначе компрессия. Адаптации алгоритмов сжатия к нуждам криптосистем, требует их классификации и отбора только тех из них, которые не противоречат законам криптографии [1].

Основными техническими характеристиками процессов сжатия являются [3]:

1. *Коэффициент сжатия* – это показатель эффективности алгоритма сжатия. Выражается в отношении численного размера исходного потока информации к размеру потока информации после преобразования.

2. *Скорость сжатия* – время, затрачиваемое на сжатие некоторого объема информации входного потока.

3. *Эффективность сжатия* – величина, показывающая, насколько сильно упакован выходной поток, при помощи применения к нему повторного сжатия, поэтому же или иному алгоритму.

Все способы сжатия можно разделить на две категории: обратимые и необратимые [3]. Под необратимым сжатием подразумевают такое преобразование входного потока данных, при котором в выходном потоке, обобщается смысл информации. При этом происходит потеря той части информации, значимость, которой не оказывает существенного влияния на общий придаваемый ею смысл. Данный способ ориентирован на мультимедийные форматы данных (видео, аудио), и неприменим для сжатия слабоизбыточной информации – текста, исполняемых файлов. Неприемлем он и для шифрования, где потеря информативности недопустима.

В некоторых случаях, когда требуется передать лишь суть события, не вдаваясь в подробности, допустимо применение сжатия с потерей, но при этом возникает необходимость оснастить криптосистему сложным искусственным интеллектом, для проведения анализа сообщения и выделения из него базового смысла. Для данной задачи основой искусственного интеллекта могут служить алгоритмы автореферирования,

наглядным примером работы которых может служить функция «Автореферат» текстового процессора Microsoft Word [4].

Обратимое сжатие приводит к снижению объема выходного потока информации без изменения его информативности, то есть – без потери информационной структуры. Таким образом, получается, что решая основную задачу – повышения криптостойкости шифра, мы косвенно улучшаем еще и его свойства – уменьшая занимаемое пространство при хранении и скорость пересылки при передаче. Именно поэтому в криптоалгоритмах данный класс методов сжатия является наиболее оптимальным.

В настоящее время распространено программное обеспечение, реализующее набор различных по своему подходу к сжатию алгоритмов. Данный класс программного обеспечения получил название *архиваторы*.

Коэффициент сжатия при каскадном использовании нескольких алгоритмов увеличивается, но при этом падает быстродействие по логарифмической зависимости. При подготовке данных к длительному хранению скорость сжатия уже не является решающим критерием качества работы архиватора. По этой причине алгоритмы сжатия в чистом виде кроме как в статическом и динамическом видео или графике не применяются, в виду большой зависимости от характеристик информации и начальных условий.

При интеграции архиватора в криптосистему его быстродействие становится одним из основных при оценке эффективности системы в целом. Практика показывает, что применение архиваторов даже с невысоким коэффициентом сжатия (до 20%) позволяет выровнять частоту появления символов в текстовых данных. Это усложняет частотный криптоанализ, но в процессе подготовки информации к шифрованию можно использовать те алгоритмы, которые для полноценного сжатия не применимы.

Многие классические методы сжатия информации [3, 5] неадекватно воспринимают однократно встречающиеся последовательности повторяющихся символов, пропуская их не сжимая. К этому классу последовательностей относятся опечатки, или повторяющиеся пробелы, вместо табуляции и аббревиатуры. При этом, если в документе встречается большое количество вышеописанных структур, то это может привести к тому, что коэффициент сжатия данной информации будет стремиться к нулю.

Наилучшим подходом в решении данной проблемы обладает алгоритм RLE [3]. В основу алгоритма RLE (Run-Length Encoding) положен принцип выявления повторяющихся последовательностей данных и замены их простой структурой, в которой указывается код данных и коэффициент повтора. Программные реализации алгоритма RLE отличаются простотой, высокой скоростью работы, вследствие однопроходного способа сжатия и декомпрессии. Наилучшими объектами для данного алгоритма являются графические файлы, в которых большие одноцветные участки изображения кодируются длинными последовательностями одинаковых байтов [6]. Этот метод также может давать заметный выигрыш на некоторых типах файлов баз данных, имеющих таблицы с фиксированной длиной полей.

Классическая реализация алгоритма состоит в том, что осуществляется поиск наименее часто встречающегося байта, называют его префиксом и делают замены цепочек одинаковых символов на тройки "префикс, счетчик, значение". Если же этот байт встречается в исходном файле один или два раза подряд, то его заменяют на пару "префикс, 1" или "префикс, 2". Остается одна неиспользованная пара "префикс, 0", которую можно использовать как признак конца упакованных данных. К примеру, последовательность AAABBCDEEEE кодируется (#3A) (#2B) (#1C) (#1D) (#4E), # – знак префикса. Сжатая последовательность повторяющихся символов представляется структурой, состоящей из трех элементов: байт-указатель на префикс, счетчик повторов символов, байт, хранящий образец повторяющегося символа.

Одна из наиболее удачных версий данного алгоритма основана на том, что байт, отвечающий за наличия префикса, объединяется с байтом счетчика повторяющихся символов [3]. В этом байте старший бит выполняет функцию флага наличия префикса. Если значение старшего бита равно единице, то данный байт является носителем префикса, а значение оставшихся семь бит – количество повторений символа не более 128. Если значение старшего бита равно нулю, то префикса нет, и значение байта интерпретируется как число в графических данных или цифра в текстовых. Вследствие этого для хранения префикса не нужно выделять дополнительный байт, что позволяет

повысить коэффициент сжатия информации, особенно при большом количестве коротких последовательностей повторяющихся символов. Эффективность (по коэффициенту сжатия) данной версии алгоритма выше, в сравнении с классической версией. Объясняется это тем, что в классическом алгоритме сжатия сжатая последовательность представлена тремя байтами, в данной версии–двумя (байт-указателем префикса, совмещенный со счетчиком и образец повторяющегося символа).

Однако данный алгоритм имеет существенный недостаток: при изъятии одного старшего бита максимальная мощность алфавита сокращается до 128 символов. Закодировать прописные и строчные буквы национального (в данном случае рассматривается русский алфавит) и латинского алфавита, а также цифры, знаки пунктуации и некоторые вспомогательные символы 7 битами (128 возможных значений) невозможно.

Целью исследований приведенных в данной работе является усовершенствование известных однопроходных алгоритмов семейства RLE, что должно привести к повышению эффективности алгоритма при сжатии алфавитной информации.

Модификация алгоритма RLE основана на устранении противоречия, выражающегося в нерациональном использовании старшего бита кодировочного байта, что приводит к неоправданному сокращению мощности используемого алфавита. Для устранения этого противоречия необходимо увеличить мощность алфавита не изменяя размер кодировочной таблицы. При кодировании любого текстового документа необходимо чтобы кодировочная таблица содержала необходимый минимум используемых символов. К ним можно отнести следующие типы: буквы строчные и прописные национального (в данном примере русский алфавит) и латинского алфавитов, цифры, знаки пунктуации и некоторые, по необходимости, вспомогательные символы. Если брать за основу кодировочную таблицу ANSII IBM cp866 [2], то потребуется 164 символа. Однако закодировать такое количество кодов символов семью битами (128 возможных значений) не представляется возможным.

Предлагается исключить избыточность в таблице, проявляющуюся в одинаковой смысловой нагрузке прописных и строчных букв, которые отличаются лишь в написании под влиянием правил грамматики. Исключив строчные или прописные буквы, а также символы построения псевдографики DOS, и другие неприменяемые спецсимволы, получим упакованную таблицу мощностью 106 кодов символов. При этом еще 22 кода символов останутся в резерве. Для сохранения стилистики документа, присвоим символам различных регистров отличительный признак – единицу в старшем разряде или, что аналогично – смещение кодового значения на 128.

Так как старший бит, необходим только в 58 случаях из 128, то это приводит к неэффективному использованию алгоритма почти для половины символов. Для повышения эффективности алгоритма, оставшиеся коды символов, не меняющие своего регистра, поместим в начало таблицы (таблица 1, где первый столбец – код символа в десятичной системе исчисления, второй – в шестнадцатеричной, третий столбец – графическое представление символа).

При наличии единицы в старшем разряде кода символов с неменяющимся регистром, будем его воспринимать как счетчик повторений символов, а младшие семь бит, как его значение.

Байты сжатой информации необходимо интерпретировать по следующим правилам:

1. Значение байта, старший бит которого равен единице – интерпретируется как:

а) если значение кода символа лежит в диапазоне от 69 до 127 включительно, то это символ принятой кодировочной таблицы 1 с противоположным регистром (в рассматриваемом примере малые прописные буквы);

б) если значение кода символа лежит в диапазоне от 0 до 68 включительно, то это байт следует рассматривать как байт-указатель на наличие последовательности повторяющихся символов – префикс, а значение младших семи бит, как количество повторяющихся символов. Значение байта, следующего за текущим, есть не что иное как код символа, количество повторений которого хранится в байте-указателе.

2. Значение байта, старший бит которого равен нулю – интерпретируется как код символа принятой кодировочной таблицы 1.

Так как минимальная длина воспринимаемой последовательности повторяемых символов равна двум, то и нумерацию количества повторов имеет смысл начать с двух. Для этого потребуется ввести смещение на два значения влево при записи и на два значения вправо при чтении (о – 2 ↔ 2 – о). При этом максимальная длина сжимаемой последовательности увеличиться до семидесяти.

Последнее преобразование не столь принципиально в виду того, что в текстовых данных символы, подряд повторяющиеся более десяти раз редкость, не говоря уже о семидесяти повторах.

Таблица 1

Упакованная кодировочная таблица

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00		32	20		64	40	@	96	60	Б
1	01	`	33	21	!	65	41	[	97	61	В
2	02	{	34	22	"	66	42	\	98	62	Г
3	03	|	35	23	#	67	43	]	99	63	Д
4	04	}	36	24	$	68	44	♥	100	64	Е
5	05	~	37	25	%	69	45	A	101	65	Ё
6	06	^	38	26	&	70	46	B	102	66	Ж
7	07	•	39	27	'	71	47	C	103	67	З
8	08	_	40	28	(	72	48	D	104	68	И
9	09	○	41	29	)	73	49	E	105	69	Й
10	0A	◙	42	2A	*	74	4A	F	106	6A	К
11	0B	♂	43	2B	+	75	4B	G	107	6B	Л
12	0C	♀	44	2C	,	76	4C	H	108	6C	М
13	0D	♪	45	2D	-	77	4D	I	109	6D	Н
14	0E	♫	46	2E	.	78	4E	J	110	6E	О
15	0F	☼	47	2F	/	79	4F	K	111	6F	П
16	10	►	48	30	0	80	50	L	112	70	Р
17	11	◄	49	31	1	81	51	M	113	71	С
18	12	↕	50	32	2	82	52	N	114	72	Т
19	13	‼	51	33	3	83	53	O	115	73	У
20	14	¶	52	34	4	84	54	P	116	74	Ф
21	15	§	53	35	5	85	55	Q	117	75	Х
22	16	▬	54	36	6	86	56	R	118	76	Ц
23	17	↨	55	37	7	87	57	S	119	77	Ч
24	18	↑	56	38	8	88	58	T	120	78	Ш
25	19	↓	57	39	9	89	59	U	121	79	Щ
26	1A	→	58	3A	:	90	5A	V	122	7A	Ъ
27	1B	←	59	3B	;	91	5B	W	123	7B	Ы
28	1C	∟	60	3C	<	92	5C	X	124	7C	Ь
29	1D	↔	61	3D	=	93	5D	Y	125	7D	Э
30	1E	▲	62	3E	>	94	5E	Z	126	7E	Ю
31	1F	▼	63	3F	?	95	5F	A	127	7F	Я

Блок-схема алгоритма, построенная на основе приведенных правил, приведена на рис. 2.

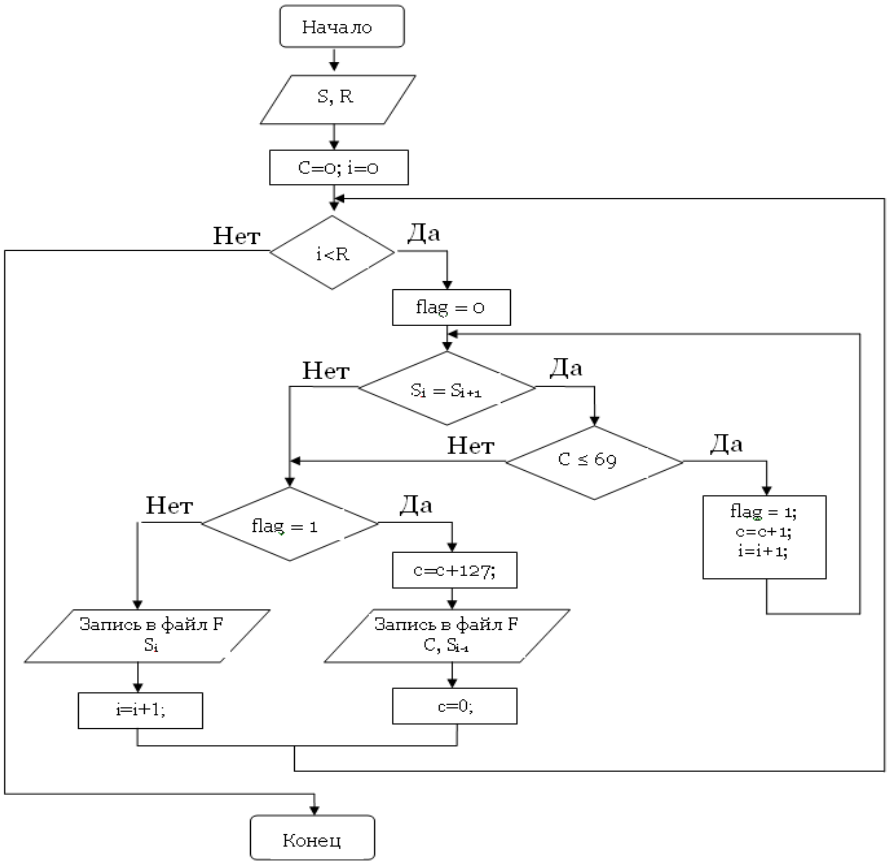

Рис. 2. Блок-схема алгоритма сжатия, где S – текстовый файл длины R, S_i – i-й элимент файла S, c – счетчик повторяющихся символов, flag – флаг наличия последовательности повторяющейся символов, F –файл, сжатой информации

В приведенном алгоритме в файле S длины R считываются последовательно символы S_i, сравниваются два рядом стоящих и при их совпадении устанавливается flag в единицу. Осуществляется подсчет повторяющихся символов, по окончанию повтора производится запись в файл F префикса со значением счетчика повторений и повторяющегося символа. Если flag принимает значение нуль, то это означает, что символ повторов не имеет и его следует записывать в файл F без префикса.

При декомпрессии символы обрабатываются в следующем порядке:

1) если код символа принадлежит диапазону от 196 до 255, то прописная буква со значением, смещенным на 127;

2) если код символа принадлежит диапазону от 127 до 195, то это байт-указателя префикса повторяющегося символа, позиция в файле которого будет следующая относительно байта префикса, количество повторов хранится в младших семи битах.

На рис. 3 отображен частотный спектр появления символов в текстовом файле размером 10 Кб, сравнение которого с данными, приведенными на рис. 1, показывает что предложенный алгоритм позволил снизить всплески частоты появления символов в среднем на 20%. Это дает возможность более эффективного применения более мощных алгоритмов сжатия [3, 5].

Описанный усовершенствованный алгоритм является однопроходным, то есть обращение к элементам файла происходит единожды, что является основной причиной его быстродействия. При этом модифицированный алгоритм RLE, с точки зрения защиты от частотного криптоанализа, даже при двукратных повторах символов, улучшает характеристики открытого текста, не увеличивая его размер. Разработанный алгоритм по сравнению с классическим RLE даёт 67 % прироста эффективности по сжатию при кратности повторов более двух и 150 % при кратности равной двум.

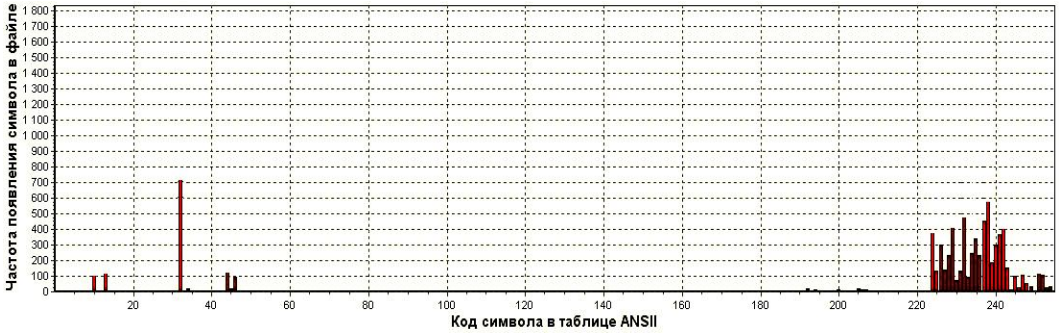

Рис. 3. Частота встречаемости символов в текстовом файле, размером 10 кбайт, подвергнутый сжатию предложенным алгоритмом

Проведенные вычислительные эксперименты показали, что применение предложенного подхода кодирования символов позволило увеличить мощность алфавита шифруемых данных на 52 %, при этом сохранить прежнюю структуру (байт-указатель, символ повтора) и соответственно размер сжатых данных – *архива*.

Таким образом, предложенный алгоритм является оптимальным в отношении эффективность/быстродействие. Его можно обоснованно применять в криптосистемах по уровню самозащиты и стойкости шифра выше среднего и аппаратных модулях шифрования информации «на лету» типа «Криптон» и аналогичных по назначению.

Литература

1. Основы криптографии [Текст] : Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В.; – Гелиос АРВ. – 2002. – с. 480.

2. Лидовский В. В.Загадочное семейство [Текст] / Лидовский В. В.// Компьютерра. – №42. – 2003 г. – с. 2.

3. Основы сжатия информации [Текст] : Фомин А.А. Санкт-Петербургский государственный технический университет, 1998. с. 82.

4. Молявко А. /Официальный учебный курс Microsoft: Microsoft Office Word 2003. Базовый курс [Текст] – М.: Эком. 2005. – 408 с.

5. Blelloch G. Introduction to Data Compression [Текст] / Blelloch G. //Computer Science Department Carnegie Mellon University, – 2001. – p. 156.

6. Ватолин Д.С. /Алгоритмы сжатия изображений [Текст] : – Издательский отдел факультета Вычислительной Математики и Кибернетики МГУ им. М.В.Ломоносова. 1999 г. — 76 с.

ALGORITHM OF COMPRESSION OF THE ALPHABETIC INFORMATION WITH ADAPTATION FOR CRYPTOSYSTEMS

V. G. POTEMKIN
N.I. KORSUNOV

Belgorod State Technological University

e-mail: skf-bgtu@yandex.ru

In this article we propose an efficient compression algorithm to protect against the interception of information transmitted. Analysis algorithms to compress the information, and identified their shortcomings, in that in the process of compression, they do not adequately perceive the sequence of repeated characters, have insufficient capacity to alphabets. To address the shortcomings, a new system of compression, for which the algorithm is given. A comparison of the proposed algorithm with a well-known.

Key words: information, reliability, compression, unauthorized access, redundancy, coding.