Performance Impact of Data Compression on Virtual Private Network Transactions (Extended Version)

John P. McGregor and Ruby B. Lee
Department of Electrical Engineering
Princeton University
Princeton, NJ 08544
{mcgregor, rblee}@ee.princeton.edu

Abstract

Virtual private networks (VPNs) allow two or more parties to communicate securely over a public network. Using cryptographic algorithms and protocols, VPNs provide security services such as confidentiality, host authentication and data integrity. The computation required to provide adequate security, however, can significantly degrade performance. In this paper, we characterize the extent to which data compression can alleviate this performance degradation. More specifically, we study the performance obtained when combining the IP Payload Compression Protocol (IPComp) with the IP Security Protocol (IPsec).

We evaluate performance using 3 system models; each of these models consists of some or all of the computation and transmission operations required to support VPN transactions. Using speedup equations that describe the performance impact of compression in the system models, we derive inequalities that specify the conditions required for data compression to improve performance. We also gather and analyze empirical performance results by simulating packet transmission over several network types and by timing the execution of IPComp and IPsec procedures on a 367 MHz HP PA-8500 processor. The results indicate that the performance depends on the compressibility of the payload data, on the throughput of the cryptographic and compression algorithms, and on the network speed. We find that compression usually improves performance when using 10 Mbps or slower networks, but compression only improves performance in systems with 100 Mbps or 1 Gbps networks when encryption is being used.

1.0 Introduction

As Internet usage grows exponentially and computing devices become increasingly interconnected, network security issues become increasingly important. Many Internet applications such web browsers and distributed databases require private communication over public networks. Using a virtual private network (VPN), multiple hosts can communicate securely over a public network. The details of VPN protocols vary, but most consist of two major steps: a handshake and bulk data encryption/authentication. The VPN is established during the handshake step. This step involves protocol and algorithm negotiation, authentication of hosts, and secret key exchanges between the hosts. The hosts can then communicate privately by encrypting and authenticating all of the data that travels over the public network.

The IP Security Protocol (IPsec) can be used to implement virtual private networks in a vendor-independent, application-invisible manner. IPsec provides a variety of security services at the IP layer for both IPv4 and IPv6 [12]. VPN bulk data encryption and authentication is supported in IPsec using the Encapsulating Security Payload and Authentication Header protocols. The Encapsulating Security Payload (ESP) provides for confidentiality of the IP packet payload, and both ESP and the Authentication Header (AH) ensure the authenticity as well as the integrity of the IP packet payload [10], [11]. The encryption and authentication provided by ESP and AH, however, require significant computational time and therefore can degrade performance when compared to unsecured transmissions.

In past work, researchers have improved the performance of secure network transactions using a variety of techniques. By adding new instructions to conventional instruction set architectures, multiple-instruction operations in implementations of cryptographic algorithms can be replaced with a single RISC or CISC instruction [17]. For example, fast bitwise permutation instructions can significantly improve the performance of DES, and support for arithmetic in Galois fields can accelerate the throughput of elliptic curve cryptosystems [17], [24]. In addition, the computation associated with many cryptographic protocols is highly parallelizable. When performing encryption, a multiprocessor system can achieve nearly linear speedup by assigning individual packets or connections to single

processing elements [16]. Cryptographic algorithms can also be explicitly designed to contain abundant instruction level parallelism; this leads to fast execution on wide superscalar processors and on multiprocessor systems.

In this paper, we investigate the performance benefit of compressing IP packet payloads when using the IP Security Protocol to implement a virtual private network. The IP Payload Compression Protocol (IPComp) employs data compression algorithms to reduce the size of packet payloads at the IP layer [23]. This size reduction decreases the time required to transmit IP packets by decreasing the amount of information that is physically transferred over the network. In addition, IPComp decreases the execution times of the encryption and authentication algorithms by reducing the amount of information to be encrypted and authenticated. However, the data compression algorithms used by IPComp consume a significant number of clock cycles. As a result, compressing the IP packet payload may increase the total time needed to complete a particular transaction and therefore degrade performance. We restrict our investigation to the performance impact of compression on bulk data encryption/authentication. Data compression would not improve the performance of the handshake step due to the nature of the algorithms and protocols involved.

We use three network-processor models to measure performance. These models represent systems such as individual servers as well as server-network-client infrastructures. The performance measure in each of these models consists of some combination of the time required to encrypt, decrypt, hash compute, hash verify, compress, decompress, and transmit data over a network. Using speedup equations that define performance in each of the network-processor models, we derive inequalities that describe the conditions required for compression to improve performance. If the throughputs of the encryption and authentication algorithms as well as the bandwidth of the network can be treated as constants, these inequalities only depend on the throughput of the compression and decompression algorithms and on the compression ratio.

Through experimentation, we analyze the performance of several combinations of network connection speeds, data benchmarks, packet payload sizes, and algorithms used by ESP, AH and IPComp. We obtain empirical performance results by executing the algorithms and simulating the network types on a 367 MHz HP PA-8500 processor. As expected, the results indicate that the performance depends on the compressibility of the payload data, on the throughput of the encryption, authentication and compression algorithms, and on the network speed. In general, the performance benefit of IPComp decreases as network bandwidth increases. We find that compression usually improves performance when using 10 Mbps or slower networks, but compression only improves performance in systems with 100 Mbps or 1 Gbps networks when encryption is being used.

The remainder of the paper is organized as follows. In Section 2, we discuss the network-processor models that we use to measure performance. Section 3 discusses the details of IPsec, and Section 4 explores the performance cost of using IPsec. In Section 5, we describe IPComp and investigate the compressibility and throughput achieved by different compression algorithms. We present speedup equations for calculating performance in the different models and derive inequalities that predict when compression will improve performance in Section 6. In Section 7, we present empirical performance results of combining IPComp and IPsec when using a 367 MHz HP PA-8500 processor. We summarize and discuss directions for future work in Section 8.

2.0 Performance Models

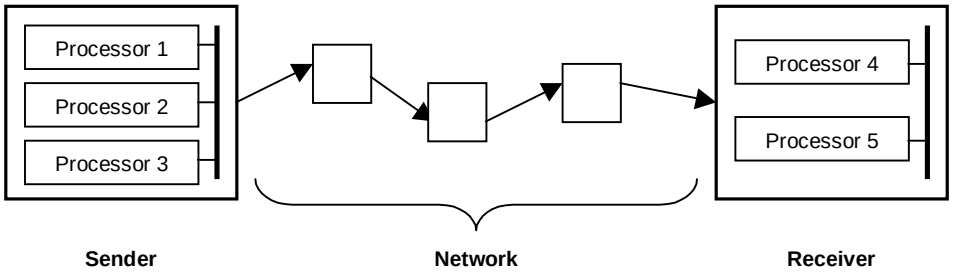
When combining IPComp with IPsec, the performance measure depends on the characteristics of the system. For example, the time needed for an individual to retrieve banking account information over the Internet consists of three major components. These components include the computational time required by the bank's servers to prepare the IP packets, the time needed to physically transmit the IP packets, and the computational time required by the individual's computer to interpret the packets. Some users, however, would not be concerned with all three of these computation and transmission components. In several situations, the user or business is only concerned with the time needed to prepare IP packets or the time needed to interpret IP packets.

The *prepare packet* operation consists of the procedures such as encryption and hash computation required to prepare the packet for secure transmission. The *interpret packet* operation consists of the procedures such as decryption and hash verification required to extract the contents of the packet upon receipt. As we will explain in Section 3, hash computation and verification are used to perform data authentication. We use the terms hash compute and hash verify interchangeably with authenticate in this paper. We describe encryption and decryption in Section 3, and we describe compression and decompression in Section 5.

Consider a network-processor system that consists of a sender, a network, and a receiver, as shown in Figure 1. Such systems can consist of multiple processors on both the receiver and sender ends, and the packets may pass through several gateways and network types before reaching their final destinations. One approach to modeling such a system would be to treat the computation and transmission segments as a pipeline. In a pipelined model, the

performance would depend on the most time-consuming segment. In this paper, however, we model the performance in terms of total latency. More specifically, we are concerned with the total amount of time required by all computation and network resources to prepare, transmit, and interpret packets. We investigate the extent to which IPComp increases or decreases the total amount of time required to complete these tasks.

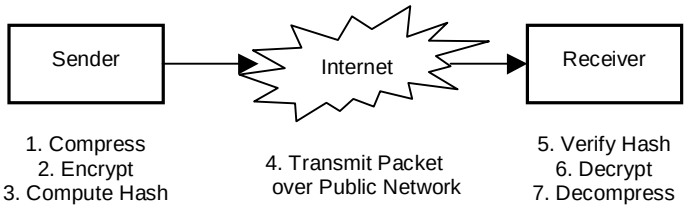
Figure 1: Network-Processor System



We evaluate performance using three network-processor models: the Local Send Model, the Local Receive Model, and the Complete Model. The Local Send Model consists of the computation needed to prepare the packet for transmission. The performance measure in this model is based upon the time needed to serially compress, encrypt, and compute the keyed hash value for the IP packet. The Local Receive Model includes the computation needed to interpret the IP packet after transmission. The performance measure in this model is based upon the time required to serially verify the keyed hash value for, decrypt, and decompress the IP packet.

Lastly, the Complete Model includes the computation needed to prepare the IP packet, the physical transmission of the packet over the network, and the computation needed to interpret the IP packet. The performance measure in this model is based upon the time needed to serially compress, encrypt, hash compute, transmit, hash verify, decrypt, and decompress. We model our network as a link of constant bandwidth. The bandwidth B equals N/T, where T is the average time required for N bytes to travel from the sender to the receiver. Figure 2 illustrates the structure of the Complete Performance Model.

Figure 2: Complete Model



3.0 IP Security Protocol

The IP Security Protocol (IPsec) provides a variety of security services at the IP layer in both IPv4 and IPv6 [12]. IPsec confidentiality and authentication services are implemented using the Encapsulating Security Payload and the Authentication Header. The Encapsulating Security Payload (ESP) provides for confidentiality of the IP packet payload using symmetric key encryption algorithms [11]. Both the Authentication Header (AH) and ESP insure the authenticity as well as the integrity of the IP packet payload using symmetric key encryption algorithms or secret-keyed one-way hash functions [10], [11]. In addition, AH provides protection against IP address spoofing. IPsec allows ESP and AH to be applied to IP packets either alone or in combination with each other [12].

Requests for Comments (RFCs) describing the following protocols include instructions for integration in both IPv4 and IPv6. Only minor differences exist between the IPsec integration procedures for the two IP versions, and these differences do not significantly affect any of the results we present in this paper. We will discuss and evaluate the relevant algorithms, protocols, and procedures only as they are implemented for IPv6, but the conclusions we present in this paper apply to both IPv4 and IPv6 IPsec implementations.

3.1 IP Encapsulating Security Payload

As stated in RFC 2406, "ESP is used to provide confidentiality, data origin authentication, connectionless integrity, an anti-replay service (a form of partial sequence integrity), and limited traffic flow confidentiality." ESP employs symmetric-key encryption algorithms such as RC5 and 3DES to provide confidentiality. ESP uses keyed hash algorithms such as SHA-1 and MD5 to provide for authentication, data integrity, and anti-replay services.

ESP transforms IPv6 packets as illustrated in Figure 3. The original IPv6 packet depicted in Figure 3 consists of 4 components. The IP header is the essential first component of every IPv6 packet. Following the IP header, the packet contains a variable number of Extension headers. These headers include (but are not limited to) the Routing header, the Fragmentation header, and the ESP header. The Upper Layer header immediately follows the Extension headers and is used by the protocol immediately above IPv6 (e.g., TCP, UDP, and ICMP). Last, the ULP (Upper Layer Protocol) Payload consists of the data to be sent and received by the ULP. The ESP header is inserted directly preceding the Destination Options header, if it exists. If a Destination Options header does not exist, the ESP header is inserted immediately preceding the ULP header and payload.

Figure 3: ESP in IPv6

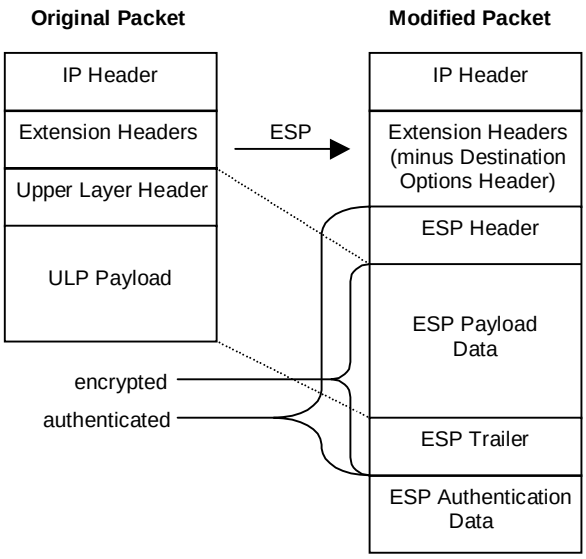
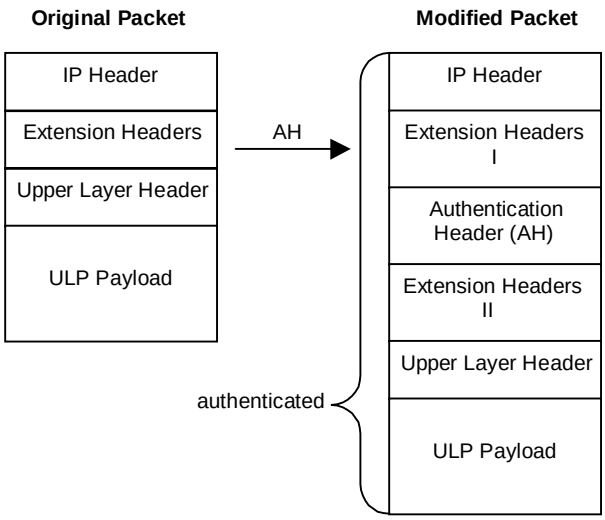


Figure 4: AH in IPv6



The ESP header consists of two 32-bit fields. The first field consists of the Security Parameters Index (SPI), which, in conjunction with the IP destination and the security protocol, specifies the Security Association for the packet. The Security Association (SA) indicates which algorithms and modes of operation are being used to perform the encryption and authentication. We discuss the encryption and authentication algorithms that we use in this study in Section 3.3. It is important to note that the SA can specify the encryption algorithm to be NULL and/or specify the authentication algorithm to be NULL [8]. If the encryption algorithm is NULL, the ciphertext is equivalent to the plaintext, i.e., the encryption function is the identity function. If the authentication algorithm is defined to be NULL, no authentication is performed, and the length of the ESP Authentication Data field is 0. The second field in the ESP header consists of the Sequence Number, which is a monotonically increasing counter value that prevents replay attacks (if authentication is employed).

The encrypted data block directly follows the ESP header [11]. This block consists of at most three components: the initialization vector (IV), the ESP payload data, and the ESP trailer. If the encryption algorithm defined by the SA requires an initialization vector, the IV is placed at the front of the encrypted data block. Although the IV is transmitted in plaintext, it is considered to be part of the ciphertext block. The ESP payload data directly follows the IV. It consists of the ciphertext result of encrypting the IPComp and Destination Options headers (if they exist in the original IP packet), the ULP header, and the ULP payload. These headers and the ULP payload are encrypted using the negotiated algorithm. Lastly, the ESP trailer, which consists of 3 fields, is encrypted using the same algorithm and placed directly following the payload data. The first of these 3 fields is a variable length padding of zeroes. The payload data can be padded by as many as 255 bytes in order to satisfy the

block alignment requirements of the encryption or authentication algorithm employed by ESP. The last two fields, the Pad Length and the previously mentioned Next Header field, are each one byte in length.

The ESP Authentication Data field follows the ESP Trailer. This field contains a variable-length Integrity Check Value (ICV) that is computed over the ESP header, the ESP payload data and the ESP trailer. The ICV is calculated using either a secret-keyed hash algorithm or a symmetric key encryption algorithm. The Security Association specifies the algorithm to be used. Byte alignment problems are resolved using the variable-length padding field contained in the ESP Trailer. More details about ESP can be found in [11].

3.2 IP Authentication Header

The IP Authentication Header (AH) provides authentication, protection against replay attacks, and connectionless integrity for IPv6 packets. AH transforms IPv6 packets as illustrated in Figure 4. The original packet is constructed exactly as described in Section 3.1. As shown in Figure 4, the AH should be inserted immediately following Extension Headers I and immediately preceding Extension Headers II [4]. Extension Headers I includes the Hop-by-Hop Options header, the Routing header, and the Fragment header. Extension Headers II includes the ESP header and the Destination Options header. In IPv6, the Authentication Header is also classified as an Extension header [4]. Note that IP packets are not required to contain any Extension headers.

The AH consists of 6 fields. The first three fields contain typical header information such as the total header size. The 32-bit fourth field is the Security Parameters Index (SPI), and the 32-bit fifth field is the Sequence Number. The Sequence Number consists of a monotonically increasing counter value that is used to prevent replay attacks. Lastly, the Authentication Data field contains the Integrity Check Value (ICV) for the packet.

The ICV is the output of the chosen authentication algorithm computed over the entire IP packet [10]. All fields that may be unpredictably modified during packet transit – unpredictably mutable fields – are set to 0 for the ICV computation. The Authentication Data field is set to 0 during the computation of the ICV as well. If the final value of a mutable field is predictable, the field is set to its final value for the ICV computation. Furthermore, the ICV must be a multiple of 64 bits in size, so padding is appended to the end of the ICV if necessary. Additional details about mutable fields and other features of AH can be found in [10].

3.3 Algorithms

In this study, we perform IPsec procedures using 4 different cryptographic algorithms: RC5, Triple DES, MD5, and SHA-1. RC5 and Triple DES (3DES) are symmetric-key encryption algorithms that provide for data confidentiality. MD5 and SHA-1 provide for data origin authentication, data integrity, and anti-replay service.

3.3.1 Encryption Algorithms

Several symmetric key algorithms are defined for use in ESP [19]. In this study, we use two ciphers in CBC mode: RC5-CBC and 3DES-EDE-CBC. CBC, which stands for Cipher Block Chaining, is a mode of operation that provides more security than the ECB (Electronic Code Book) mode of operation, which is easier to implement. CBC mode involves XORing a randomly generated 64-bit initialization vector (IV) with the first plaintext block before encryption. Subsequently, the jth plaintext block is XORed with the (j-1)th ciphertext block before encryption. Additional details about CBC may be found in [22].

DES, which is an acronym for Data Encryption Standard, is a symmetric-key block cipher that uses a 64-bit key to encrypt 64-bit blocks [22]. Eight of the 64 key bits are parity bits, so the effective key length is 56 bits. Furthermore, DES is a Feistel cipher, so the rounds of the encryption algorithm are equivalent to the rounds of the decryption algorithm. DES has been a worldwide standard for over 20 years [22]. Triple DES (henceforth referred to as 3DES) provides more security than DES by encrypting a single block three times (with DES) using three different 56-bit keys [22]. We implement 3DES in EDE (encrypt-decrypt-encrypt) mode, as prescribed in [19]. To encrypt a block of data, we encrypt with the first key, then decrypt with the second key, and then encrypt with the third key. To decrypt a block of data, we decrypt with the third key, encrypt with the second key, and then decrypt with the first key. We apply CBC mode using the outer chaining technique as described in [22]. 3DES is cryptographically twice as strong as DES, but 3DES takes roughly three times as much computational time as DES to encrypt and decrypt blocks.

RC5 is a symmetric key block cipher that is patented by RSA Data Security, Incorporated [21]. RC5 can operate over different block sizes and key lengths; in this study we use a block size of 64 bits and a key length of 128 bits. We implement RC5 in CBC mode as described in [2]. RC5 is considered to be one of the fastest secure symmetric-key block ciphers, whereas 3DES is one of the slowest. RC5 is also slightly more secure than 3DES: the effective key length of the RC5 key is 128 bits, whereas the effective key length of the 3DES key is 112 bits.

3.3.2 Authentication Algorithms

Two algorithms that are defined for use with AH and with ESP are HMAC-MD5 and HMAC-SHA-1 [10], [11]. In this study, we shall evaluate the performance of both of these algorithms. HMAC is a mechanism that provides message authentication using an iterative cryptographic hash function and a secret symmetric key [13]. We compute the HMAC over the data *P* as follows:

$$H((K \oplus opad), H((K \oplus ipad), P))$$

H is a cryptographic hash function that hashes data by iterating some basic function on data blocks of size *B*. In addition, the comma represents the concatenation function, and *K* is the HMAC secret key. The terms *opad* and *ipad* are fixed byte sequences of length *B*. Details concerning key length and use can be found in [13].

MD5 is a cryptographic hash function that accepts plaintext blocks of size 64 bytes and outputs a 16-byte authentication value [14]. SHA-1, a government standard, is a cryptographic hash function that accepts 64 byte plaintext blocks and outputs a 20-byte authentication value [15]. SHA-1 is considered to be a cryptographically superior hash function, but MD5 is faster [13].

3.4 Implementing ESP and AH

When combining ESP and AH, the AH header directly precedes the ESP header in the modified IP packet. Since AH authenticates more data in an IP packet than the authentication services in ESP, we always use AH to perform authentication. We only use ESP to perform encryption.

Hash computation and hash verification are equivalent operations: the sender and the receiver execute the same hash algorithm over identical keys and identical packets. Therefore, the two operations require the same amount of computation time when executed on the same platform. In addition, when simulating the NULL algorithm of a particular protocol, we treat the packet as if the protocol were not being employed at all. This policy decreases network transmission time by not adding a useless protocol header to the IP packet.

4.0 Performance Impact of IPsec

In this section, we explore the performance costs of employing IPsec procedures. We obtain data concerning the throughput of IPsec procedures by implementing and executing the cryptographic algorithms on a HP Visualize C360 workstation. This workstation consists of a HP 64-bit 367 MHz PA-8500 processor with 1.5 MB of on-chip L1 cache. In addition, the workstation has 128 MB of RAM. We implemented all four of the authentication and encryption algorithms in C. We use the HPUX C compiler (`cc`) to build the modules, and we employ full compiler and linker optimizations (i.e., `+O4`). We obtain the timing results for all of the authentication and encryption procedures using the UNIX `clock()` function. In order to avoid imprecision resulting from the relatively high granularity of the `clock()` output, we execute the IPsec procedures thousands of times. We obtain the final timing result by dividing the total time needed to complete the thousands of iterations by the total number of iterations executed.

4.1 Network Types and Payload Sizes

We simulate several network types and payload sizes. The network connections include 56 kbps (phone line modem), 1.54 Mbps (T1, wireless), 10 Mbps (Ethernet), 100 Mbps (Ethernet), and 1 Gbps (Ethernet). In a lab environment, these network types can achieve 90% of their maximum link speeds [9]. In this study, we relax this condition somewhat; we assume all the network connections sustain 80% of their maximum throughput. Although 80% throughput may not be realistic for many real-world systems, one can approximate the effective bandwidth of most networks with one of the 5 network models used in this study.

We evaluate the performance of the IPsec procedures using 3 ULP payload sizes: 1 kilobyte, 4 kilobytes, and 63 kilobytes. In a variety of network environments, 1 kilobyte and 4 kilobytes are commonly used TCP payload sizes [9]. TCP, which stands for Transmission Control Protocol, is a connection-based protocol that employs a 20-byte payload header [20]. Additional details concerning TCP can be found in [20]. We assume that all ULP payloads are TCP payloads. The maximum size of an IPv6 packet minus the IP header is 64 KB, so the 63 KB payload size represents the largest possible payload size but leaves room for ESP, AH, IPComp, and TCP headers. We set the network MTU to be 1280 bytes, the minimum MTU allowed in IPv6, and we fragment packets accordingly [4]. We assume that all the original IP packets solely consist of a 40-byte IPv6 header, an 8-byte fragmentation header (if necessary), a 20-byte TCP header, and a variable length TCP payload. The use of additional headers or a transport protocol other than TCP will not significantly affect the performance results.

4.2 Performance Results and Analysis

The execution time of the encryption and authentication algorithms is simply a function of the input size (rather than a function of both the input size and the statistical characteristics of the input data). Since hash computation and hash verification are equivalent operations, hash computation and hash verification will consume the same amount of time. Furthermore, DES is a Feistel cipher, so the encryption speed of 3DES is equivalent to the decryption speed of 3DES. Although RC5 is not a Feistel cipher, the speed of the encryption procedure is roughly the same as the speed of the decryption procedure. We summarize the throughput of the encryption and authentication algorithms for the three packet sizes in Table 1. The algorithms are executed in the modes of operation described in Section 3. From Table 1, we see that the throughput of the authentication algorithms is much higher than that of the encryption algorithms. Furthermore, MD5 runs slightly faster than SHA-1 and RC5 runs three times as fast as 3DES.

Table 1: Performance of Encryption and Authentication Algorithms (Mbps)

Algorithm Name	Payload Sizes			
	1K	4K	63K	Average
MD5	229	299	332	287
SHA-1	214	274	302	263
RC5	96	97	96	96
3DES	28	28	28	28

We now calculate the performance degradation caused by different combinations of these algorithms in the Complete Performance Model. As described in Section 2, the Complete Performance Model consists of the computation required to serially compress, encrypt, compute the hash for, verify the hash for, decrypt, and decompress the IP packet. We quantitatively determine the level of performance degradation by calculating the speedup *S* as follows.

$$S = \frac{T_{NET}}{T_E + T_{HC} + T_{SECNET} + T_{HV} + T_D}$$

In this equation, T_{NET}, T_{SECNET}, T_E, T_{HC}, T_{HV}, and T_D represent the original packet transmission time, the encrypted/authenticated packet transmission time, the encryption time, the hash computation time, the hash verification time, and the decryption time, respectively. T_{NET} may not equal T_{SECNET} because of the headers that are added to the encrypted/authenticated packet. Values of *S* close or equal to 1.00 indicate minor performance degradation, whereas values closer to 0.0 indicate enormous performance degradation. We calculate the speedup results using a TCP payload size of 4 KB, and we assume that all the network types sustain 80% of their maximum bandwidth. We execute the algorithms and simulate network transmissions on the PA-RISC workstation described at the beginning of Section 4. The speedup results for the 5 network bandwidths and eight combinations of encryption and authentication algorithms are listed in Table 2.

Table 2: Speedups (Slowdowns) Resulting from Authentication and Encryption

Network Type	Algorithm Combination							
	MD5	SHA-1	RC5	RC5 MD5	RC5 SHA-1	3DES	3DES MD5	3DES SHA-1
56 kbps	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.98
1.54 Mbps	0.99	0.98	0.97	0.96	0.96	0.92	0.90	0.90
10 Mbps	0.94	0.94	0.86	0.82	0.82	0.64	0.62	0.62
100 Mbps	0.65	0.63	0.39	0.32	0.32	0.15	0.14	0.14
1 Gbps	0.16	0.15	0.06	0.04	0.04	0.02	0.02	0.02

From Table 2, we see that encryption and authentication never degrade performance more than 10% when using 56 kbps or 1.54 Mbps network connections. As bandwidth increases, however, the performance impact of authentication and encryption becomes more pronounced. Both encryption and authentication decimate performance when using a 1 Gbps link. The results also show that the encryption algorithms have a greater effect on

the performance than the authentication algorithms, for the encryption routines require much more computation. This performance differential between encryption and authentication becomes more pronounced as bandwidth increases. We now investigate the degree to which data compression can alleviate this performance problem.

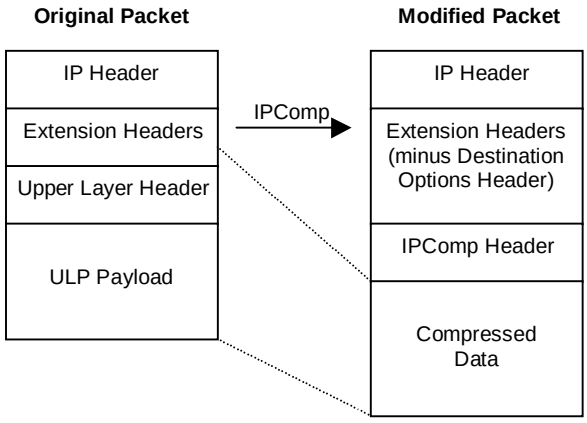
5.0 IP Payload Compression

The IP Payload Compression protocol (IPComp) employs data compression algorithms to reduce the size of IP packets [23]. This size reduction decreases the time required to transmit IP packets by decreasing the number of bits that are physically transferred over the network. Furthermore, IPComp decreases the execution time of encryption and authentication algorithms by reducing the amount of information to be encrypted and authenticated. In this section, we describe IPComp in detail, and then we analyze the compressibility and throughputs achieved by the LZS and DEFLATE compression algorithms.

5.1 Protocol Description

IPComp reduces the size of IP packets in IPv4 and in IPv6 using data compression algorithms [23]. The protocol specification includes several restrictions and guidelines concerning these compression algorithms. To preserve the consistency of the packet payload, the compression algorithm must be lossless. Furthermore, the compression of a packet payload must be completed before any IP security processing is performed. A cryptographically secure encryption algorithm outputs ciphertext that has high entropy; therefore an encrypted payload is incompressible. It follows that the decompression and reassembly of IP packets must also occur after authentication or decryption. In addition, each IP packet must be compressed and decompressed independently of other packets, since IP packets may arrive out of order or may never arrive at all. Lastly, the total size in bytes of the compressed payload and the IPComp header must be smaller than the size of the original payload. If the header and the compressed payload are of equal or of greater size than the original payload, the original payload is sent (without any IPComp header). This non-expansion policy saves clock cycles at the receiver end and guarantees that network traffic will not increase. Additional details may be found in [23].

Figure 5: IPComp in IPv6



IPComp transforms an IPv6 packet as shown in Figure 5. The IPComp header is 4 bytes in size and consists of three fields. The first byte stores the Next Header field, the second byte stores the Flags field (which is always set to 0), and the last two bytes store the Compression Parameter Index (CPI). The CPI indicates which compression algorithm is being used and includes any parameters needed to configure the compression algorithm. These configuration parameters and the chosen compression algorithm are determined during the negotiation of the IPComp Association (IPCA) between the communicating parties. Furthermore, in IPv6, the IPComp header is inserted immediately preceding the Destination Options header in the IP packet [23]. If the Destination Options header does not exist, the IPComp header is inserted immediately preceding the upper-layer header (e.g., the TCP header) in the IP packet. The compression algorithm used by IPComp compresses all the data that follows the IPComp header in the IP packet. The Destination Options header, the upper layer header, and the ULP payload are all compressed.

5.2 Data Compression Algorithms

We evaluate the performance of IPComp using two lossless compression algorithms: DEFLATE and LZS [7], [18]. DEFLATE compresses data using a combination of the LZ77 algorithm and Huffman coding [18]. In this study, we use zlib, a freely distributed implementation of DEFLATE, to obtain our performance results. We use version 1.1.3, and we maintain all the default compression settings. Additional details concerning zlib and DEFLATE can be found in [5] and in [6], respectively.

LZS is a lossless compression algorithm based on LZ77 that employs a sliding window of maximum size 2 kilobytes [7]. LZS is an ANSI standard and is patented by Hi/fn, Inc. In our C implementation of LZS, we store the compression dictionary as a quickly accessible and infrequently updated order-3 hash table. We also optimized our

implementation by employing 64-bit features and by cleverly "shaving" the algorithm. By not using the full compression power of LZS, we can obtain huge gains in throughput but pay only a small penalty in compression ratio. Our implementation requires approximately 50 KB of RAM and outperforms DEFLATE in throughput by an order of magnitude. Additional details concerning LZS may be found in [7] and in [1].

Unlike encryption and authentication algorithms, the performance of the compression algorithms depends on the size as well as the statistical properties of the input data. In other words, the throughput of the compression algorithms depends on the compressibility of the input data. Hence, we analyze the performance of the compression algorithms using an eclectic group of data benchmarks.

Table 3: Benchmark Descriptions

Name	Description
obj2	Compiled code for Apple Macintosh: Knowledge support system
progl	Lisp source code for system software
paper2	A technical paper entitled "Computer (in)security" by Witten
trans	Transcript of a session on a terminal
book1	A book entitled *Far from the Madding Crowd* by Hardy
pic	Picture number 5 from the CCITT Facsimile test files (text + drawings)
gif	Compressed GIF file that contains a campus map
random	Randomly generated data

5.3 Data Benchmarks

In a real-world networking environment, a user may transmit a rich variety of data types and sizes. For example, a user may send or receive a previously compressed 100 KB text file, a 1.3 MB binary executable, or a 12 KB bitmap entirely composed of white pixels. We chose 8 data benchmarks to obtain the performance results; their names and descriptions are listed in Table 3.

Six of these benchmarks – obj2, progl, paper2, trans, book1, and pic – are members of the Calgary corpus. Researchers use the Calgary corpus to evaluate the practical performance of text compression algorithms. More details about the Calgary corpus can be found in [3]. The remaining two benchmarks, gif and random, represent a GIF compressed image file and a randomly generated binary data file, respectively. We expect, therefore, that the gif and random benchmarks will be relatively incompressible. The 6 Calgary corpus benchmarks will exhibit different levels of compressibility, and therefore the performance will vary when using IPComp in conjunction with ESP and AH.

5.4 Compressibility Results

We compute the compressibility results for both DEFLATE and LZS using the 8 benchmarks and the 3 payload sizes defined in Section 4.1. In order to avoid any compression anomalies associated with compressing the first few bytes (e.g., 1 KB or 4 KB) of the benchmarks, we compress several different portions of the benchmarks when using the 1 KB and 4 KB ULP payload sizes. For the 1 KB payloads, we compress the first 40 1-KB blocks of the benchmark, and the compressibility results are based upon the average compressibility of those 40 blocks. Similarly, for the 4 KB payload sizes, we compress the first 10 4-KB blocks of the benchmark. When using the 63 KB payload size, however, we only compress the first 63 KB block of the benchmark. In addition, to avoid timing imprecision resulting from the relatively high granularity of the result of the `clock()` function, we repeat the compression and decompression of the blocks thousands of times.

The eight benchmarks exhibit many different levels of compressibility. Table 4 summarizes the compression ratios achieved by the DEFLATE and LZS algorithms. The size of the compressed data block includes the 4-byte IPComp header. Table 4 shows that the zlib implementation of DEFLATE achieves a higher compression ratio than LZS for each packet size and benchmark combination. Figure 6 illustrates the compression ratios achieved by the two algorithms for 4 KB payloads. DEFLATE usually outperforms LZS in compression ratio by a factor of 1.0 to 2.0, but the ratio of the compression ratios can exceed 5.0 for highly compressible data.

5.5 Throughput Results

The performance of the compression algorithms depends on the benchmark compressibility and on the benchmark data size. Table 5 lists the compression and decompression rates for DEFLATE, and Table 6 lists the compression and decompression rates for LZS. Figures 14 through 19 in the Appendix compare the compression

Table 4: Compression Ratios

Benchmark Name	DEFLATE			LZS		
	1K	4K	63K	1K	4K	63K
obj2	1.76	2.08	2.36	1.52	1.67	1.77
progl	2.31	3.16	4.23	1.76	2.16	2.33
paper2	1.74	2.08	2.70	1.28	1.50	1.61
trans	1.93	2.55	5.02	1.52	1.82	2.01
book1	1.68	1.94	2.34	1.22	1.40	1.46
pic	26.10	67.48	21.41	10.04	12.70	8.01
gif	0.99	1.00	1.00	0.89	0.89	0.89
random	0.99	1.00	1.00	0.89	0.89	0.89

Figure 6: Comparison of Compression Ratios for 4 KB Payloads

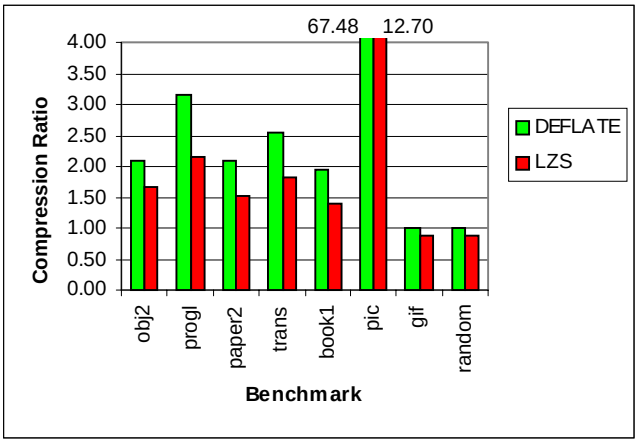


Table 5: DEFLATE Performance (Mbps)

Benchmark Name	Compression			Decompression		
	1K	4K	63K	1K	4K	63K
obj2	14	21	23	66	121	365
progl	18	31	31	77	146	492
paper2	17	25	23	67	123	365
trans	17	29	43	70	143	492
book1	17	23	20	68	112	340
pic	34	72	76	166	299	703
gif	14	28	31	0	0	0
random	14	29	31	0	0	0

Table 6: LZS Performance (Mbps)

Benchmark Name	Compression			Decompression		
	1K	4K	63K	1K	4K	63K
obj2	217	210	201	254	252	245
progl	234	245	252	253	263	263
paper2	178	179	182	224	214	211
trans	216	222	232	259	255	259
book1	172	169	169	220	207	203
pic	983	1047	735	498	537	476
gif	186	160	150	0	0	0
random	186	161	151	0	0	0

and decompression rates of the two algorithms graphically. The zero values for gif and random in the decompression columns are a consequence of IPComp's non-expansion policy. If the compression ratio is less than or equal to 1, the ULP header and payload are transmitted in their original forms: no compressed data is transmitted and an IPComp header is not included in the packet. As a result, the receiver does not decompress any data. This policy reduces network transmission time and eliminates unnecessary computation for the receiver.

Tables 5 and 6 show that LZS always greatly outperforms DEFLATE in compression rate. LZS compresses data nearly as fast as SHA-1 authenticates data, and DEFLATE compresses data as slowly as 3DES encrypts data. In addition, LZS significantly outperforms DEFLATE in decompression rate for 1 KB and 4 KB payloads. DEFLATE, however, always achieves a higher decompression rate than LZS for 63 KB payloads.

We now know that DEFLATE always achieves a higher compression ratio than LZS, and we know that LZS usually compresses and decompresses data faster than DEFLATE. Which algorithm should we use to maximize performance? The answer to this question depends on the performance model and network speed. We will address this issue in greater detail later in this paper.

6.0 Calculating the Performance Impact of IPComp on IPsec Transmissions

In this section, we present equations that describe the performance impact of combining ESP and AH with IPComp. First, we introduce and explain the equations we use to calculate speedups for the different performance models. Second, we use these speedup equations to describe the conditions required for compression to improve performance in each model.

6.1 Speedup Calculation

We can express the speedup in each network-processor performance model as a single equation. A speedup greater than or equal to 1 indicates that IPComp reduces the total amount of time needed to conduct a secure transaction. Hence, a speedup greater than or equal to 1 means compression improves performance. A speedup less than 1 indicates that compression increases the total time needed to conduct the transmission, and therefore compression degrades performance. T_E, T_D, T_{HC}, T_{HV}, and T_{SECNET} represent the encryption time, decryption time, hash computation time, hash verification time, and transmission time, respectively. The variables C_E, C_D, C_{HC}, C_{HV},

and C_{SECNET} represent the time needed to encrypt, decrypt, compute the hash for, verify the hash for, and transmit a compressed packet, respectively. If encryption were not being used, for example, then T_E and C_E would both equal 0. Furthermore, the variables T_{COMP} and T_{DECOMP} represent the time required to perform the payload compression and decompression, respectively. In the Local Send Model, we calculate the speedup S using the following equation:

$$S = \frac{T_E + T_{HC}}{T_{COMP} + C_E + C_{HC}}$$

We compute the speedup in the Local Receive Performance Model as follows:

$$S = \frac{T_{HV} + T_D}{C_{HV} + C_D + T_{DECOMP}}$$

Note that it is possible for the speedup in the Local Receive Model to have no value. If neither encryption nor authentication is used and the packet payload is uncompressible, a divide by zero situation will occur. Hence, S is undefined in this case. The Complete Performance Model includes the computation performed by the sender and by the receiver as well as the physical transmission of the data. We therefore calculate the speedup in the Complete Performance Model as follows.

$$S = \frac{T_E + T_{HC} + T_{SECNET} + T_{HV} + T_D}{T_{COMP} + C_E + C_{HC} + C_{SECNET} + C_{HV} + C_D + T_{DECOMP}}$$

6.2 Predicting the Performance Impact of Compression

Using the speedup equations defined in Section 6.1, we describe the conditions required for compression to improve the performance in each model. For the Local Send Model, the speedup equation can be rewritten as follows:

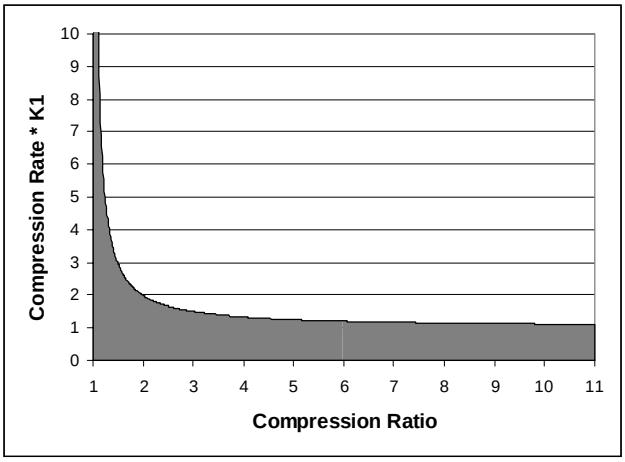
$$S = \frac{T_E + T_{HC}}{T_{COMP} + C_E + C_{HC}} = \frac{\frac{1}{R_E} N_P + \frac{1}{R_{HC}} N_P}{\frac{1}{R_{COMP}} N_P + \frac{1}{R_E}\left(\frac{N_P}{X}\right) + \frac{1}{R_{HC}}\left(\frac{N_P}{X}\right)}$$

R_E, R_{HC}, R_{COMP}, and N_P equal the encryption rate, the hash computation rate, the compression rate, and the size of the original (uncompressed) ULP payload, respectively. X is the compression ratio, which equals the size of the uncompressed payload divided by the size of the compressed payload. In order to make the relationships clear, we choose to neglect the overhead resulting from packet headers. For example, the amount of data to be hashed is not N_P as the previous equation indicates. The AH ICV is calculated over the entire packet, so the amount of data to be hashed is at least N_P plus the size of the IP and AH headers. The total size of all the packet headers is usually less than 100 bytes, so if the payload size is relatively large, e.g., multiple KB, the omission of the header overheads will not significantly affect the results.

Compression improves performance if the speedup achieved in a particular model is greater than or equal to 1. Hence, by setting the right side of the previous equation to be greater than or equal to 1, we obtain a relationship between the compression ratio and the algorithm throughputs such that compression will improve performance. N_P cancels out since we are not considering overhead from packet headers, and we now have the following equation for the Local Send Model:

$$\left(1 - \frac{1}{X}\right)\left(\frac{1}{R_E} + \frac{1}{R_{HC}}\right) \geq \frac{1}{R_{COMP}}$$

Figure 7: Performance-Improving Relationship Between Compression Rate and Ratio in the Local Send Model



If the encryption and signing rates are constant, we have:

$$R_{COMP} \geq \left(\frac{1}{K_1}\right)\left(\frac{X}{X-1}\right), \text{ where } K_1 = \frac{1}{R_E} + \frac{1}{R_{HC}}$$

This inequality shows that compression will improve performance in the Local Send Model if the compression rate is greater than some constant factor times the compression ratio divided by the compression ratio minus 1. Figure 7 illustrates this relationship between the compression rate and the compression ratio. All the points in the shaded region yield a speedup less than or equal to 1, and all the points in the white region yield a performance improvement.

Using similar algebraic manipulation, we obtain the following equation that specifies the conditions required for compression to improve performance in the Local Receive Model:

$$\left(1-\frac{1}{X}\right)\left(\frac{1}{R_D} + \frac{1}{R_{HV}}\right) \geq \frac{1}{R_{DECOMP}}$$

R_D, R_{HV}, and R_{DECOMP} equal the decryption rate, the hash verification rate, and the decompression rate, respectively. If the decryption and authentication rates are constant, we have:

$$R_{DECOMP} \geq \left(\frac{1}{K_2}\right)\left(\frac{X}{X-1}\right), \text{ where } K_2 = \frac{1}{R_D} + \frac{1}{R_{HV}}$$

This inequality specifies a relationship between decompression rate and compression ratio in the Local Receive Model such that the resulting speedup will be greater than or equal to 1. The graph of this relationship has exactly the same shape as the graph for the relationship between the compression rate and the compression ratio in the Local Send Model.

For the Complete Model, we rewrite the speedup equation as follows.

$$S = \frac{\frac{1}{R_E}N_P + \frac{1}{R_{HC}}N_P + \frac{1}{R_{SECNET}}N_P + \frac{1}{R_{HV}}N_P + \frac{1}{R_D}N_P}{\frac{1}{R_{COMP}}N_P + \frac{1}{R_E}\left(\frac{N_P}{X}\right) + \frac{1}{R_{HC}}\left(\frac{N_P}{X}\right) + \frac{1}{R_{SECNET}}\left(\frac{N_P}{X}\right) + \frac{1}{R_{HV}}\left(\frac{N_P}{X}\right) + \frac{1}{R_D}\left(\frac{N_P}{X}\right) + \frac{1}{R_{DECOMP}}N_P}$$

R_{SECNET} is the transmission rate of the network (i.e., effective network bandwidth). N_P cancels out, and when we impose the condition that speedup must be greater than or equal to 1, we have:

$$\left(1-\frac{1}{X}\right)\left(\frac{1}{R_E} + \frac{1}{R_D} + \frac{1}{R_{HC}} + \frac{1}{R_{HV}} + \frac{1}{R_{SECNET}}\right) \geq \frac{1}{R_{COMP}} + \frac{1}{R_{DECOMP}}$$

If the encryption, decryption, hash computation, hash verification, and transmission rates are constant, we obtain the following expression.

$$K_3\left(1-\frac{1}{X}\right) \geq \frac{1}{R_{COMP}} + \frac{1}{R_{DECOMP}}, \text{ where } K_3 = \frac{1}{R_E} + \frac{1}{R_D} + \frac{1}{R_{HC}} + \frac{1}{R_{HV}} + \frac{1}{R_{SECNET}}$$

This inequality describes a relationship between compression rate, decompression rate, and compression ratio in the Complete Model such that the resulting speedup will be greater than or equal to 1.

Using these inequalities, one may determine whether compression will improve the performance of secure packet transactions without conducting extensive simulations. More accurate predictions can be made if the packet header overhead terms are included in the equations. When the header overheads are included, the N_P terms do not cancel out and the equations become somewhat more complicated. In addition, secure network transactions in some systems involve other time-consuming steps such as bus transfer time or disk access time. The speedup equations presented here could be modified to model a particular system more accurately by adding relevant latency terms.

7.0 Experimental Results and Analysis

In this section, we evaluate the experimental performance results when executing the security and compression algorithms on a HP Visualize C360 workstation. This workstation consists of a HP 64-bit 367 MHz PA-8500 processor with 1.5 MB of on-chip L1 cache. In addition, the workstation has 128 MB of DRAM, and it achieves a SPECint95 performance rating of 26.0 and a SPECfp95 performance rating of 28.1. We calculate speedups using all

combinations of compression algorithms, encryption algorithms, authentication algorithms, ULP payload sizes, data benchmarks, network bandwidths, and performance models. We execute and time the encryption and authentication algorithms as described in Section 4, and we execute, time, and measure the compressibility achieved by the compression algorithms as described in Section 5. We simulate the different network types assuming they all maintain 80% of their maximum bandwidths.

We present the speedups achieved in the 3 different models using all the combinations of simulation parameters in Tables 7 through 20 (in the Appendix). Tables 7 and 8 include the speedups for the Local Send Model, Tables 9 and 10 include the speedups for the Local Receive Model, and Tables 11 through 20 contain the speedups for the Complete Model over the 5 network bandwidths. The NULL columns in these tables list the speedups induced by compression when neither encryption nor authentication is used. Speedup values less than 1.00 (i.e., "slowdowns") are listed in **red**, and speedup values greater than or equal to 1.00 are listed in **black**. Note that rounding is used, and therefore some of the **black** values are actually slightly less than 1.00.

Because of the enormous size of the tables, we only present the most valuable performance results graphically. We illustrate results for 4 of the 8 benchmarks and for 2 of the 3 packet sizes. Using the compressibility results discussed in Section 5, we divide the data benchmarks into 4 groups according to level of compressibility. We select 1 benchmark from each of these 4 groups: pic, gif, book1, and trans. The benchmarks pic and gif represent highly compressible payloads and incompressible payloads, respectively, and book1 and trans fall in between. In addition, we only present results for the 1 KB and 63 KB payload sizes. Most of the results for the 4 KB payload size closely resemble the results for the 1 KB payload size.

The choice of compression algorithm depends on the performance model and on the network speed. In the Local Send Model, we find that LZS always yields a larger speedup than DEFLATE. This fact is a consequence of the superior compression throughput of LZS. In the Local Receive Model, however, LZS only yields higher performance than DEFLATE for 1 KB payloads and for 4 KB payloads if slow encryption (3DES) is *not* used. DEFLATE yields a larger speedup than LZS for 63 KB payloads due to its superior decompression rate. In addition, DEFLATE achieves a higher speedup than LZS for 4 KB payloads (if slow encryption is employed) due to the large compression ratios achieved by DEFLATE. In the Complete Model, we discover that compression ratio is more important than compression throughput when using slow network links, whereas the compression and decompression rates are more important than the compression ratio for fast network links. We find that DEFLATE usually yields a higher speedup than LZS for 56 kbps and 1.544 Mbps connections, and LZS usually yields a higher speedup than DEFLATE for 10 Mbps, 100 Mbps, and 1 Gbps links. It is important to note that these conclusions concerning compression algorithms and network types are highly implementation dependent. For example, if our LZS implementation were significantly slower, it would be advisable to use DEFLATE rather than LZS for 10 Mbps links. In general, fast compression algorithms should be used with fast network connections, and algorithms that achieve high compression ratios should be used with slow network connections.

Upon inspecting the speedup results, we discover that certain different algorithm combinations produce similar results. For example, consider the case where AH is used but ESP is not employed. In almost every case, the speedups for MD5 and SHA-1 (when using compression) are both greater than 1 or are both less than 1. In other words, the decision to compress when employing AH but not ESP is independent of the authentication algorithm being used. We expected this result, for the throughputs of these two algorithms differ by only 7.5% when using 63 KB payloads. In addition, for almost every case, we find that the RC5, the RC5/MD5, and the RC5/SHA-1 algorithm combinations all have speedups greater than 1 or all have speedups less than 1. In the few cases where one of the RC5 combinations yields a speedup less than 1 whereas another RC5 combination yields a speedup greater than 1, the smallest speedup is always extremely close to 1 (i.e., > 0.95). Lastly, for every case, we find that the speedups for the 3DES, the 3DES/MD5, and the 3DES/SHA-1 algorithm combinations are either all greater than or equal to 1 or are all less than 1.

This leaves us with 4 algorithm combination classes: combinations that include 3DES (slow encryption), combinations that include RC5 (fast encryption), combinations that include authentication only, and the case where neither authentication nor encryption are employed (henceforth referred to as NULL). The results of the NULL case can be used to determine whether compression improves the performance of ordinary, unsecured communications. We present performance results for the most computationally intensive algorithm combination in each class in order to demonstrate the maximum benefit of payload compression. These algorithm combinations include NULL, SHA-1, RC5/SHA-1, and 3DES/SHA-1, since SHA-1 requires more computation than MD5. In addition, the speedup results we present for these algorithm combinations are based upon the compression algorithm that yields the highest performance for the given performance model, network type, and packet size. Our recommendations for choosing a compression algorithm are listed above.

Figure 8: Speedups in the Local Receive Model

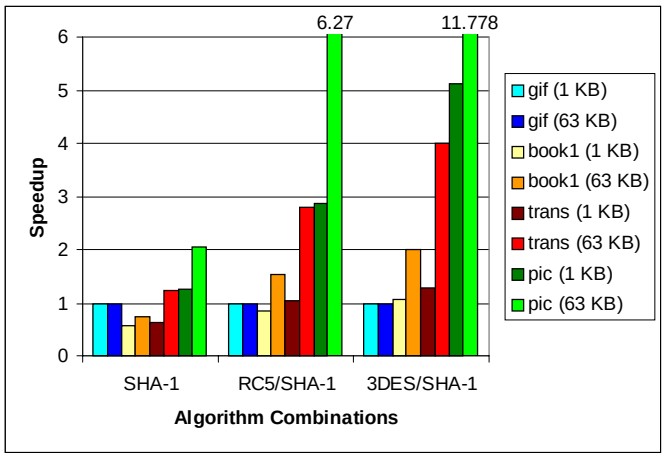


Figure 9: Speedups in the Local Send Model

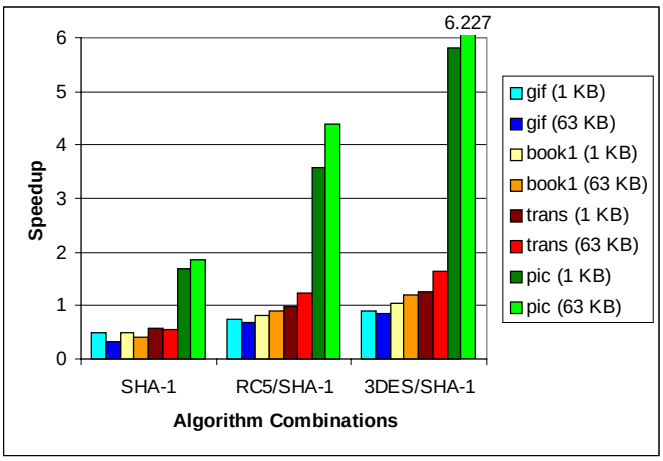


Figure 10: Speedups in the Complete Model When Using Compression Only

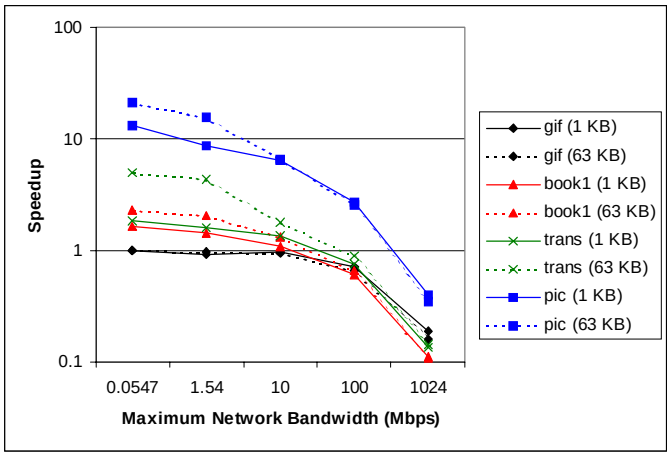


Figure 11: Speedups in the Complete Model When Using SHA-1

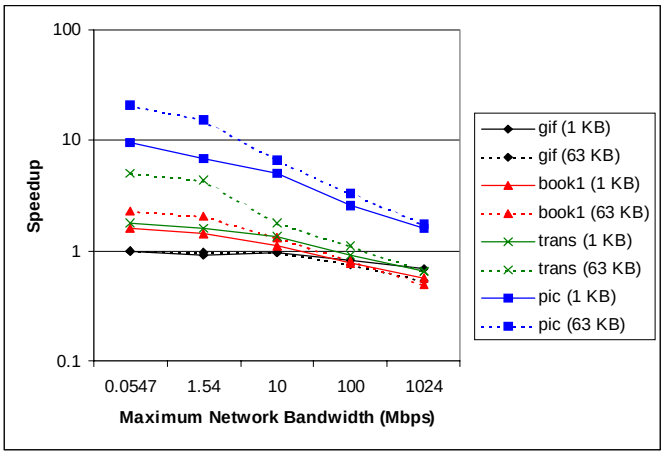


Figure 12: Speedups in the Complete Model When Using RC5/SHA-1

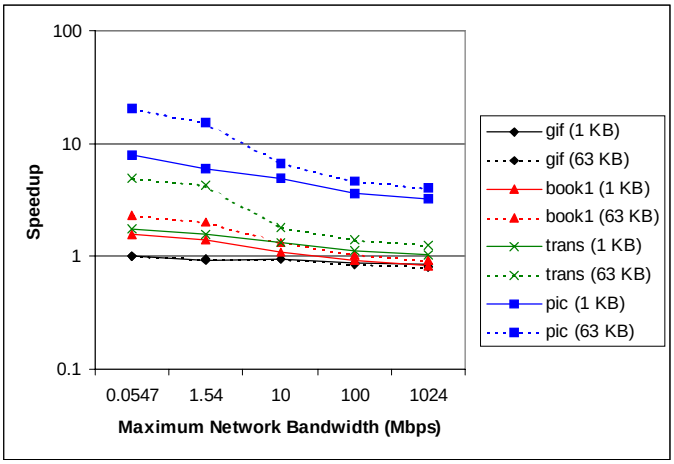
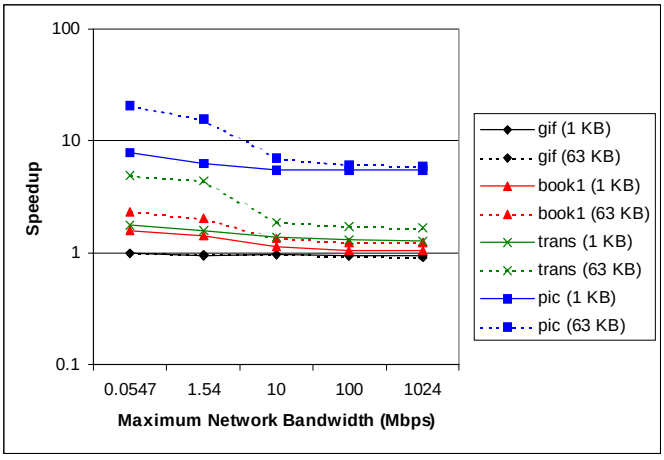


Figure 13: Speedups in the Complete Model When Using 3DES/SHA-1



In Figure 8, we present speedup results for the Local Receive Model. Since this model does not include a network connection, the NULL algorithm combination will always yield a speedup of 0 (or undefined in some cases). Therefore, we do not include the NULL combination in the graph. From Figure 8, we see that compression improves the performance of SHA-1 in the Local Receive Model only if the payload data is highly compressible. In addition, we find that compression improves the performance of RC5/SHA-1 and 3DES/SHA-1 for most benchmarks and packet sizes. In summary, the results suggest that compression should only be used to improve performance in the Local Receive Model if encryption is being used. It is important to note, however, that compression may degrade performance when using computationally light encryption algorithms if the payload is relatively uncompressible.

Figure 9 illustrates the speedup results for the Local Send Model. Since this model also lacks a network connection, the NULL algorithm combination always produces a speedup of 0. Therefore, we do not include the NULL combination in the graph. Figure 9 indicates that compression should be used to improve performance in the Local Send Model if slow encryption (e.g., 3DES) is being used. Compression can also improve the performance of fast encryption and the performance of authentication if the packet payloads are extremely compressible.

Figures 10 through 13 illustrate the speedups obtained for the 4 algorithm combinations in the Complete Model. Figure 10 depicts the speedups achieved when using compression without encryption or authentication. In this case, we see that compression usually improves performance for network bandwidths less than or equal to 10 Mbps. IPComp applied alone, however, almost always degrades performance in the Complete Model when using 100 Mbps and 1 Gbps networks. Figure 11 illustrates the speedups obtained when using SHA-1 in the Complete Model. As we conclude for the NULL algorithm combination, compression should be employed to improve performance when using authentication (without encryption) if the network bandwidth is less than or equal to 10 Mbps. Compression can also be used to improve performance when using standalone authentication for 100 Mbps and 1 Gbps networks if the packet payloads are expected to be highly compressible (i.e., compressible by a factor of 10). Figures 12 and 13 show the speedup results when using RC5/SHA-1 and 3DES/SHA-1 in the Complete Model, respectively. In both graphs, we see that compression improves performance when using network connections less than or equal to 10 Mbps. For 3DES/SHA-1, we see that compression improves performance when using 100 Mbps or 1 Gbps network connections. In addition, for RC5/SHA-1, we see that compression improves performance over 100 Mbps and 1 Gbps links if the data is reasonably compressible (i.e., compressible by a factor of 1.5).

If the security and compression algorithms are all executed on workstations similar to the one described at the beginning of this section, we conclude that compression will improve performance in the Complete Model for network bandwidths less than or equal to 10 Mbps. In addition, compression improves performance in the Complete Model when using network bandwidths as high as 1 Gbps if the payload data is reasonably compressible and encryption is being used. We define reasonably compressible data to mean a compression algorithm can reduce the size of packet payload by at least a factor of 1.5. If processing power increases relative to network speed, the speedups caused by compression will also increase. If processing power decreases relative to network speed, compression will be more likely to cause performance degradation. Similarly, if compression speed and ratio increase relative to encryption/authentication speed, performance speedups due to compression will increase. If compression speed and ratio decrease relative to encryption/authentication speed, compression will be less beneficial for system performance.

8.0 Summary

In this paper, we investigate the performance impact of combining data compression with encryption and authentication to provide for fast, secure transactions in virtual private networks. We define three network-processor models that we use to measure performance. In each of these models, we implement VPN security features using the IP Security Protocol (IPsec). We show that IPsec can seriously degrade system performance, especially when using high-bandwidth networks. Data compression can potentially alleviate this performance problem by reducing the amount of data that is physically transferred over the network and by reducing the amount of information that is processed by the encryption and authentication algorithms. Compression algorithms consume clock cycles, however, and therefore compression can potentially degrade performance. We implement data compression in our system models using the IP Payload Compression Protocol (IPComp).

We describe the performance impact of compression in each of the models with speedup equations. Using these speedup equations, we derive inequalities that can be used to determine when compression will yield a performance improvement in a particular model. The inequalities require knowledge of the security algorithm throughputs, the bandwidth of the network, and the expected throughput and compressibility achieved by the compression algorithm. We also present experimental performance results by executing the security and compression algorithms on 367

MHz PA-8500 processor. For a sender-network-receiver model, we find that compression always improves performance when using network bandwidths of 10 Mbps or less. Compression also improves performance when employing encryption and using network bandwidths of up to 1 Gbps if the payload data is reasonably compressible. Future work includes exploring the impact of compression on secure information processing using more complex models that include communications pipelines, multiple clients and servers, and multiprocessor systems.

Acknowledgements

The authors thank Zhijie Shi for implementing the authentication and encryption algorithms we used in this study. This work was supported by the Hewlett Packard Corporation.

References

[1] American National Standards Institute, Inc., "Data Compression Method – Adaptive Coding with Sliding Window for Information Interchange", ANSI X3.241-1994, August 1994.
[2] Baldwin, R., and R. Rivest, "The RC5, RC5-CBC, RC5-CBC-Pad, and RC5-CTS Algorithms", RFC 2040, October 1996.
[3] Bell, T.C., et al., *Text Compression*, Prentice Hall, Englewood Cliffs, NJ, 1990.
[4] Deering, S., and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
[5] Deutsch, P., "DEFLATE Compressed Data Format Specification version 1.3", RFC 1951, May 1996.
[6] Deutsch, P., "ZLIB Compressed Data Format Specification version 3.3", RFC 1950, May 1996.
[7] Friend, R., and R. Monsour, "IP Payload Compression Using LZS", RFC 2395, December 1998.
[8] Glenn, R., and S. Kent, "The NULL Encryption Algorithm and Its Use With IPsec", RFC 2410, November 1998.
[9] Hsu, Wan-Yen, personal communication, September 1999.
[10] Kent, S., and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.
[11] Kent, S., and R. Atkinson, "IP Encapsulating Security Payload", RFC 2406, November 1998.
[12] Kent, S., and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.
[13] Krawczyk, H., et al., "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.
[14] Madson, C., and R. Glenn, "The Use of HMAC-MD5-96 within ESP and AH", RFC 2403, November 1998.
[15] Madson, C., and R. Glenn, "The Use of HMAC-SHA-1-96 within ESP and AH", RFC 2404, November 1998.
[16] Nahum, E., et al., "Parallelized Network Security Protocols", *Proceedings of the Internet Society Symposium on Network and Distributed System Security (SNDSS `96)*, February 1996.
[17] Nahum, E., et al., "Towards High Performance Cryptographic Software", *Proceedings of the Third IEEE Workshop on the Architecture and Implementation of High Performance Communications Subsystems (HPCS '95)*, August 1995.
[18] Pereira, R., "IP Payload Compression Using DEFLATE", RFC 2394, December 1998.
[19] Pereira, R., and R. Adams, "The ESP CBC-Mode Cipher Algorithms", RFC 2451, November 1998.
[20] Postel, J., "Transmission Control Protocol", RFC 793, September 1981.
[21] Rivest, R., "RC5 Encryption Algorithm", *Dr. Dobb's Journal*, vol. 20, no. 1, January 1995.
[22] Schneier, B., *Applied Cryptography Second Edition*, John Wiley & Sons, New York, NY, 1996.
[23] Shacham, A., et al., "IP Payload Compression Protocol", RFC 2393, December 1998.
[24] Shi, Z., and R. Lee, "Bit Permutation Instructions for Accelerating Software Cryptography", to be published in *Proceedings of the IEEE International Conference on Application-specific Systems, Architectures and Processors*, July 2000.

Appendix

Figure 14: Compression Rate for 1 KB Packets

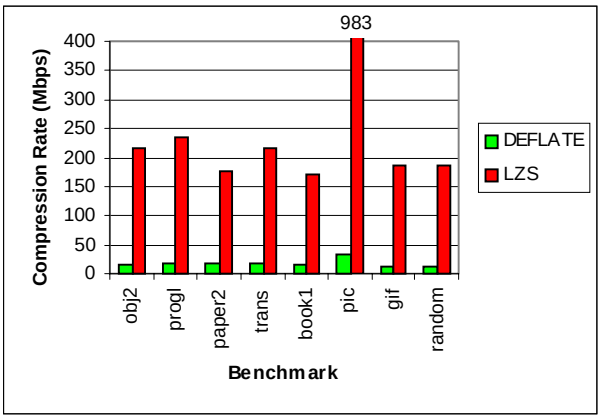


Figure 15: Decompression Rate for 1 KB Packets

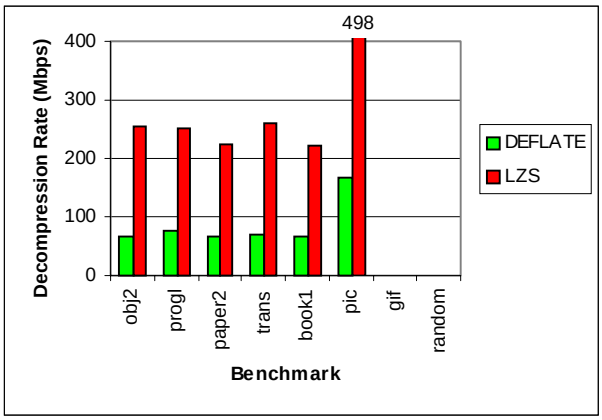


Figure 16: Compression Rate for 4 KB Packets

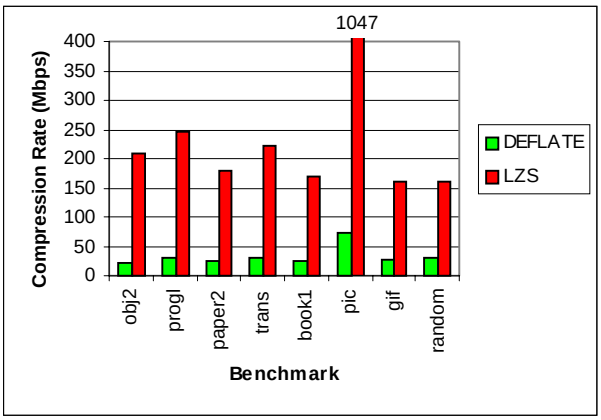


Figure 17: Decompression Rate for 4 KB Packets

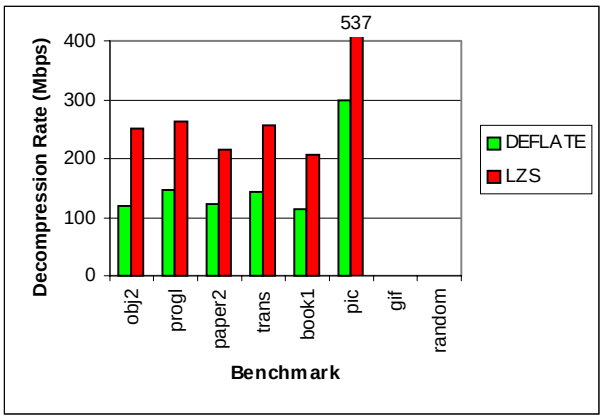


Figure 18: Compression Rate for 63 KB Packets

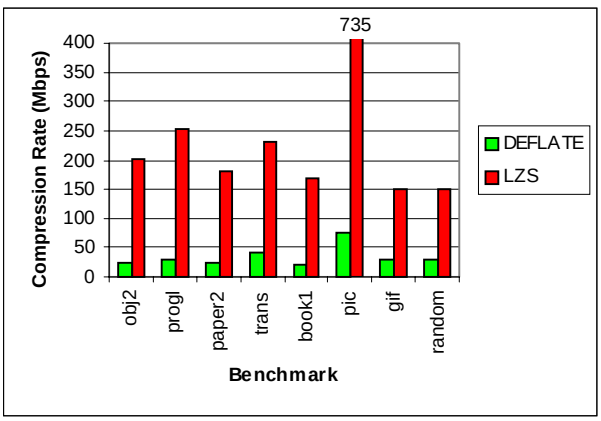


Figure 19: Decompression Rate for 63 KB Packets

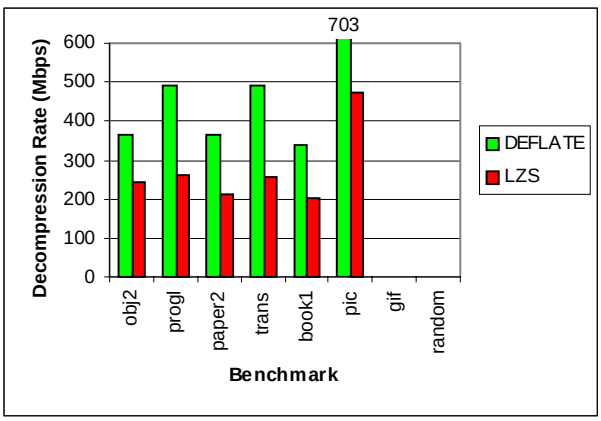


Table 7: Speedups for the Local Send Model Using DEFLATE

Name	Payload Size	Encryption/Authentication Algorithm(s) Used								
		NULL	RC5	3DES	MD5	SHA-1	RC5-MD5	RC5-SHA-1	3DES-MD5	3DES-SHA-1
obj2	1 KB	0.00	0.13	0.39	0.06	0.07	0.19	0.19	0.43	0.43
progl	1 KB	0.00	0.18	0.52	0.08	0.09	0.24	0.25	0.56	0.57
paper2	1 KB	0.00	0.16	0.46	0.08	0.08	0.22	0.22	0.49	0.50
trans	1 KB	0.00	0.17	0.48	0.08	0.08	0.23	0.23	0.52	0.52
book1	1 KB	0.00	0.16	0.45	0.07	0.08	0.22	0.22	0.48	0.49
pic	1 KB	0.00	0.34	1.16	0.15	0.16	0.47	0.48	1.23	1.24
gif	1 KB	0.00	0.12	0.33	0.06	0.06	0.17	0.17	0.36	0.36
random	1 KB	0.00	0.12	0.33	0.06	0.06	0.17	0.17	0.36	0.36
obj2	4 KB	0.00	0.20	0.56	0.07	0.08	0.25	0.26	0.59	0.60
progl	4 KB	0.00	0.29	0.83	0.10	0.11	0.37	0.38	0.88	0.88
paper2	4 KB	0.00	0.23	0.63	0.08	0.09	0.29	0.30	0.66	0.67
trans	4 KB	0.00	0.27	0.75	0.09	0.10	0.34	0.35	0.79	0.79
book1	4 KB	0.00	0.22	0.59	0.08	0.08	0.28	0.28	0.63	0.63
pic	4 KB	0.00	0.74	2.53	0.24	0.26	0.95	0.97	2.69	2.69
gif	4 KB	0.00	0.23	0.51	0.09	0.10	0.28	0.29	0.53	0.53
random	4 KB	0.00	0.23	0.51	0.09	0.10	0.28	0.29	0.54	0.54
obj2	63 KB	0.00	0.22	0.61	0.07	0.07	0.27	0.28	0.65	0.65
progl	63 KB	0.00	0.30	0.90	0.09	0.10	0.38	0.39	0.96	0.96
paper2	63 KB	0.00	0.22	0.64	0.07	0.07	0.28	0.28	0.68	0.68
trans	63 KB	0.00	0.41	1.18	0.13	0.14	0.52	0.53	1.25	1.26
book1	63 KB	0.00	0.19	0.55	0.06	0.07	0.24	0.25	0.58	0.59
pic	63 KB	0.00	0.76	2.45	0.23	0.25	0.97	0.99	2.63	2.65
gif	63 KB	0.00	0.25	0.53	0.09	0.09	0.30	0.30	0.55	0.55
random	63 KB	0.00	0.25	0.53	0.09	0.09	0.30	0.30	0.55	0.55

Table 8: Speedups for the Local Send Model Using LZS

Name	Payload Size	Encryption/Authentication Algorithm(s) Used								
		NULL	RC5	3DES	MD5	SHA-1	RC5-MD5	RC5-SHA-1	3DES-MD5	3DES-SHA-1
obj2	1 KB	0.00	0.91	1.28	0.57	0.57	0.99	0.99	1.27	1.28
progl	1 KB	0.00	1.03	1.45	0.61	0.64	1.11	1.12	1.44	1.44
paper2	1 KB	0.00	0.76	1.06	0.49	0.50	0.84	0.85	1.07	1.07
trans	1 KB	0.00	0.89	1.26	0.57	0.58	0.98	0.99	1.26	1.26
book1	1 KB	0.00	0.73	1.02	0.47	0.49	0.82	0.81	1.03	1.02
pic	1 KB	0.00	4.85	7.42	1.57	1.68	3.53	3.56	5.86	5.81
gif	1 KB	0.00	0.65	0.87	0.46	0.48	0.74	0.74	0.89	0.89
random	1 KB	0.00	0.65	0.87	0.46	0.48	0.74	0.74	0.88	0.89
obj2	4 KB	0.00	0.93	1.37	0.49	0.52	1.03	1.04	1.38	1.38
progl	4 KB	0.00	1.17	1.73	0.57	0.62	1.28	1.30	1.74	1.74
paper2	4 KB	0.00	0.82	1.22	0.42	0.46	0.92	0.92	1.23	1.23
trans	4 KB	0.00	1.01	1.48	0.52	0.55	1.12	1.12	1.49	1.49
book1	4 KB	0.00	0.77	1.14	0.40	0.43	0.86	0.87	1.15	1.16
pic	4 KB	0.00	5.42	9.50	2.07	2.29	5.40	5.41	8.81	8.87
gif	4 KB	0.00	0.62	0.85	0.35	0.37	0.69	0.69	0.86	0.87
random	4 KB	0.00	0.62	0.85	0.35	0.37	0.69	0.70	0.86	0.87
obj2	63 KB	0.00	0.95	1.42	0.45	0.48	1.06	1.07	1.44	1.44
progl	63 KB	0.00	1.23	1.87	0.57	0.61	1.37	1.38	1.89	1.90
paper2	63 KB	0.00	0.87	1.30	0.41	0.44	0.97	0.98	1.31	1.32
trans	63 KB	0.00	1.10	1.62	0.52	0.56	1.22	1.24	1.64	1.65
book1	63 KB	0.00	0.80	1.18	0.38	0.41	0.88	0.90	1.20	1.20
pic	63 KB	0.00	3.86	6.13	1.74	1.84	4.32	4.38	6.21	6.23
gif	63 KB	0.00	0.61	0.84	0.32	0.33	0.67	0.67	0.85	0.85
random	63 KB	0.00	0.61	0.85	0.31	0.33	0.67	0.68	0.86	0.86

Table 9: Speedups for the Local Receive Model Using DEFLATE

Name	Payload Size	Encryption/Authentication Algorithm(s) Used								
		NULL	RC5	3DES	MD5	SHA-1	RC5-MD5	RC5-SHA-1	3DES-MD5	3DES-SHA-1
obj2	1 KB	0.00	0.48	1.01	0.25	0.27	0.61	0.62	1.04	1.04
progl	1 KB	0.00	0.58	1.26	0.29	0.31	0.72	0.73	1.29	1.29
paper2	1 KB	0.00	0.49	1.01	0.26	0.27	0.61	0.61	1.04	1.03
trans	1 KB	0.00	0.52	1.09	0.27	0.29	0.65	0.65	1.12	1.12
book1	1 KB	0.00	0.48	1.00	0.26	0.27	0.60	0.61	1.03	1.03
pic	1 KB	0.00	1.54	4.75	0.60	0.65	1.74	1.77	4.29	4.30
gif	1 KB	no value	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
random	1 KB	no value	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
obj2	4 KB	0.00	0.76	1.41	0.33	0.36	0.89	0.90	1.44	1.44
progl	4 KB	0.00	1.00	1.97	0.41	0.46	1.18	1.19	2.01	2.00
paper2	4 KB	0.00	0.78	1.43	0.34	0.37	0.91	0.92	1.45	1.46
trans	4 KB	0.00	0.93	1.69	0.40	0.43	1.08	1.10	1.73	1.73
book1	4 KB	0.00	0.71	1.32	0.31	0.34	0.83	0.84	1.35	1.35
pic	4 KB	0.00	2.86	9.04	0.89	0.97	3.34	3.41	8.83	8.92
gif	4 KB	no value	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
random	4 KB	no value	1.02	1.00	1.00	1.01	1.01	1.01	1.00	1.00
obj2	63 KB	0.00	1.44	2.01	0.74	0.80	1.58	1.59	2.03	2.03
progl	63 KB	0.00	2.30	3.42	1.10	1.17	2.56	2.60	3.46	3.47
paper2	63 KB	0.00	1.56	2.25	0.77	0.84	1.72	1.74	2.27	2.27
trans	63 KB	0.00	2.45	3.92	1.13	1.24	2.78	2.80	3.98	3.99
book1	63 KB	0.00	1.39	1.96	0.71	0.75	1.54	1.54	1.99	1.99
pic	63 KB	0.00	5.21	11.48	1.89	2.04	6.23	6.27	11.81	11.78
gif	63 KB	no value	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
random	63 KB	no value	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table 10: Speedups for the Local Receive Model Using LZS

Name	Payload Size	Encryption/Authentication Algorithm(s) Used								
		NULL	RC5	3DES	MD5	SHA-1	RC5-MD5	RC5-SHA-1	3DES-MD5	3DES-SHA-1
obj2	1 KB	0.00	0.96	1.31	0.63	0.63	1.04	1.04	1.30	1.30
progl	1 KB	0.00	1.03	1.47	0.64	0.67	1.12	1.13	1.46	1.46
paper2	1 KB	0.00	0.81	1.10	0.56	0.56	0.89	0.90	1.11	1.10
trans	1 KB	0.00	0.96	1.29	0.63	0.64	1.03	1.04	1.29	1.29
book1	1 KB	0.00	0.77	1.05	0.54	0.56	0.86	0.85	1.06	1.06
pic	1 KB	0.00	3.33	6.17	1.18	1.26	2.89	2.89	5.14	5.13
gif	1 KB	no value	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
random	1 KB	no value	1.01	1.00	1.01	1.00	1.01	1.01	1.00	1.00
obj2	4 KB	0.00	1.00	1.41	0.55	0.59	1.10	1.10	1.42	1.42
progl	4 KB	0.00	1.19	1.76	0.60	0.65	1.31	1.32	1.77	1.77
paper2	4 KB	0.00	0.88	1.26	0.48	0.51	0.98	0.98	1.28	1.28
trans	4 KB	0.00	1.06	1.52	0.57	0.61	1.16	1.17	1.53	1.52
book1	4 KB	0.00	0.84	1.19	0.46	0.49	0.92	0.93	1.20	1.20
pic	4 KB	0.00	3.77	7.61	1.33	1.46	4.03	4.04	7.27	7.34
gif	4 KB	no value	1.00	1.00	1.00	1.01	1.00	1.00	1.00	1.00
random	4 KB	no value	1.00	1.00	1.00	1.01	1.00	1.00	1.00	1.00
obj2	63 KB	0.00	1.03	1.47	0.51	0.56	1.14	1.15	1.49	1.49
progl	63 KB	0.00	1.25	1.86	0.60	0.63	1.39	1.41	1.89	1.89
paper2	63 KB	0.00	0.91	1.33	0.46	0.49	1.01	1.02	1.35	1.35
trans	63 KB	0.00	1.13	1.65	0.55	0.61	1.25	1.27	1.68	1.68
book1	63 KB	0.00	0.86	1.22	0.43	0.46	0.95	0.96	1.23	1.23
pic	63 KB	0.00	3.03	5.50	1.20	1.31	3.51	3.56	5.61	5.63
gif	63 KB	no value	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
random	63 KB	no value	1.00	1.00	1.00	1.01	1.00	1.00	1.00	1.00

Table 11: Speedups for the Complete Model (56 kbps Network) Using DEFLATE

Name	Payload Size	Encryption/Authentication Algorithm(s) Used								
		NULL	RC5	3DES	MD5	SHA-1	RC5-MD5	RC5-SHA-1	3DES-MD5	3DES-SHA-1
obj2	1 KB	1.70	1.67	1.67	1.67	1.66	1.64	1.64	1.64	1.64
progl	1 KB	2.19	2.14	2.14	2.12	2.11	2.08	2.07	2.08	2.07
paper2	1 KB	1.68	1.65	1.65	1.65	1.65	1.63	1.62	1.63	1.62
trans	1 KB	1.85	1.83	1.83	1.81	1.80	1.79	1.78	1.79	1.78
book1	1 KB	1.63	1.61	1.61	1.61	1.60	1.59	1.58	1.59	1.58
pic	1 KB	13.28	10.28	10.29	10.13	9.51	8.36	7.95	8.37	7.96
gif	1 KB	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
random	1 KB	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
obj2	4 KB	2.06	2.05	2.05	2.05	2.04	2.03	2.03	2.03	2.03
progl	4 KB	3.06	3.02	3.02	3.02	3.01	2.98	2.97	2.98	2.97
paper2	4 KB	2.06	2.06	2.06	2.05	2.05	2.04	2.04	2.04	2.04
trans	4 KB	2.51	2.48	2.48	2.48	2.48	2.46	2.45	2.46	2.45
book1	4 KB	1.94	1.92	1.92	1.92	1.92	1.91	1.91	1.91	1.91
pic	4 KB	41.38	34.63	34.66	32.79	30.98	28.45	27.09	28.49	27.13
gif	4 KB	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
random	4 KB	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
obj2	63 KB	2.35	2.35	2.35	2.35	2.35	2.35	2.35	2.35	2.35
progl	63 KB	4.20	4.20	4.20	4.20	4.19	4.19	4.19	4.19	4.19
paper2	63 KB	2.68	2.68	2.68	2.68	2.68	2.68	2.68	2.68	2.68
trans	63 KB	4.98	4.98	4.98	4.98	4.97	4.97	4.97	4.97	4.97
book1	63 KB	2.32	2.32	2.32	2.32	2.32	2.32	2.32	2.32	2.32
pic	63 KB	20.97	20.84	20.85	20.79	20.74	20.67	20.62	20.67	20.63
gif	63 KB	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
random	63 KB	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table 12: Speedups for the Complete Model (56 kbps Network) Using LZS

Name	Payload Size	Encryption/Authentication Algorithm(s) Used								
		NULL	RC5	3DES	MD5	SHA-1	RC5-MD5	RC5-SHA-1	3DES-MD5	3DES-SHA-1
obj2	1 KB	1.49	1.48	1.48	1.48	1.47	1.46	1.46	1.46	1.46
progl	1 KB	1.71	1.68	1.68	1.68	1.67	1.65	1.65	1.65	1.65
paper2	1 KB	1.27	1.25	1.25	1.26	1.26	1.25	1.24	1.25	1.24
trans	1 KB	1.49	1.47	1.47	1.47	1.46	1.45	1.45	1.45	1.45
book1	1 KB	1.21	1.20	1.20	1.20	1.20	1.19	1.19	1.19	1.19
pic	1 KB	7.52	6.50	6.51	6.46	6.21	5.72	5.54	5.73	5.54
gif	1 KB	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
random	1 KB	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
obj2	4 KB	1.65	1.64	1.64	1.64	1.64	1.63	1.63	1.63	1.63
progl	4 KB	2.15	2.14	2.14	2.13	2.13	2.12	2.12	2.12	2.12
paper2	4 KB	1.49	1.49	1.49	1.49	1.49	1.48	1.48	1.48	1.48
trans	4 KB	1.82	1.81	1.81	1.81	1.81	1.80	1.80	1.80	1.80
book1	4 KB	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39
pic	4 KB	11.82	11.20	11.21	11.04	10.84	10.51	10.33	10.52	10.34
gif	4 KB	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
random	4 KB	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
obj2	63 KB	1.76	1.76	1.76	1.76	1.76	1.76	1.76	1.76	1.76
progl	63 KB	2.33	2.33	2.33	2.33	2.33	2.33	2.33	2.33	2.33
paper2	63 KB	1.61	1.61	1.61	1.61	1.61	1.61	1.61	1.61	1.61
trans	63 KB	2.01	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
book1	63 KB	1.46	1.46	1.46	1.46	1.46	1.46	1.46	1.46	1.46
pic	63 KB	7.99	7.97	7.97	7.96	7.96	7.95	7.94	7.95	7.94
gif	63 KB	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
random	63 KB	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table 13: Speedups for the Complete Model (1.54 Mbps) Using DEFLATE

Name	Payload Size	Encryption/Authentication Algorithm(s) Used								
		NULL	RC5	3DES	MD5	SHA-1	RC5-MD5	RC5-SHA-1	3DES-MD5	3DES-SHA-1
obj2	1 KB	1.45	1.44	1.46	1.43	1.43	1.43	1.42	1.44	1.44
progl	1 KB	1.87	1.85	1.87	1.83	1.82	1.81	1.80	1.83	1.82
paper2	1 KB	1.47	1.46	1.47	1.46	1.45	1.44	1.44	1.46	1.45
trans	1 KB	1.61	1.60	1.61	1.58	1.57	1.57	1.57	1.59	1.58
book1	1 KB	1.43	1.43	1.44	1.42	1.41	1.41	1.41	1.42	1.42
pic	1 KB	8.59	7.38	7.67	7.08	6.80	6.29	6.07	6.55	6.32
gif	1 KB	0.92	0.92	0.93	0.92	0.92	0.93	0.93	0.93	0.93
random	1 KB	0.92	0.92	0.93	0.92	0.92	0.93	0.93	0.93	0.93
obj2	4 KB	1.82	1.82	1.83	1.81	1.81	1.81	1.81	1.82	1.82
progl	4 KB	2.69	2.67	2.70	2.66	2.66	2.64	2.64	2.67	2.66
paper2	4 KB	1.85	1.85	1.86	1.84	1.84	1.84	1.84	1.85	1.85
trans	4 KB	2.24	2.23	2.24	2.22	2.22	2.21	2.21	2.23	2.22
book1	4 KB	1.74	1.73	1.74	1.73	1.73	1.73	1.72	1.74	1.73
pic	4 KB	22.86	21.01	21.84	19.75	19.09	18.42	17.87	19.20	18.62
gif	4 KB	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
random	4 KB	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
obj2	63 KB	2.09	2.09	2.11	2.09	2.09	2.09	2.09	2.11	2.11
progl	63 KB	3.61	3.62	3.65	3.61	3.61	3.62	3.62	3.65	3.65
paper2	63 KB	2.35	2.35	2.37	2.35	2.35	2.35	2.35	2.37	2.37
trans	63 KB	4.35	4.36	4.39	4.35	4.35	4.36	4.35	4.39	4.39
book1	63 KB	2.03	2.04	2.05	2.03	2.03	2.04	2.04	2.05	2.05
pic	63 KB	15.51	15.56	15.81	15.44	15.41	15.48	15.46	15.73	15.71
gif	63 KB	0.96	0.96	0.97	0.96	0.96	0.97	0.97	0.97	0.97
random	63 KB	0.96	0.96	0.97	0.96	0.96	0.96	0.96	0.97	0.97

Table 14: Speedups for the Complete Model (1.54 Mbps) Using LZS

Name	Payload Size	Encryption/Authentication Algorithm(s) Used								
		NULL	RC5	3DES	MD5	SHA-1	RC5-MD5	RC5-SHA-1	3DES-MD5	3DES-SHA-1
obj2	1 KB	1.47	1.46	1.47	1.45	1.45	1.44	1.44	1.45	1.44
progl	1 KB	1.68	1.66	1.66	1.65	1.64	1.63	1.62	1.63	1.63
paper2	1 KB	1.25	1.24	1.24	1.24	1.24	1.23	1.23	1.23	1.23
trans	1 KB	1.47	1.45	1.45	1.45	1.44	1.43	1.43	1.43	1.43
book1	1 KB	1.19	1.18	1.19	1.18	1.18	1.18	1.18	1.18	1.18
pic	1 KB	7.33	6.41	6.52	6.21	5.99	5.58	5.41	5.71	5.54
gif	1 KB	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
random	1 KB	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
obj2	4 KB	1.62	1.61	1.62	1.61	1.61	1.61	1.60	1.61	1.61
progl	4 KB	2.11	2.10	2.10	2.09	2.08	2.08	2.08	2.09	2.08
paper2	4 KB	1.47	1.46	1.47	1.46	1.46	1.46	1.46	1.46	1.46
trans	4 KB	1.79	1.78	1.79	1.78	1.78	1.77	1.77	1.78	1.77
book1	4 KB	1.37	1.37	1.37	1.37	1.36	1.36	1.36	1.37	1.37
pic	4 KB	11.39	10.83	10.94	10.57	10.40	10.13	9.96	10.25	10.10
gif	4 KB	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
random	4 KB	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
obj2	63 KB	1.73	1.73	1.73	1.73	1.73	1.73	1.73	1.73	1.73
progl	63 KB	2.28	2.28	2.28	2.28	2.28	2.28	2.28	2.28	2.28
paper2	63 KB	1.58	1.58	1.58	1.58	1.58	1.58	1.58	1.58	1.58
trans	63 KB	1.97	1.97	1.97	1.97	1.97	1.97	1.97	1.97	1.97
book1	63 KB	1.44	1.44	1.44	1.44	1.44	1.44	1.44	1.44	1.44
pic	63 KB	7.74	7.73	7.75	7.72	7.71	7.71	7.70	7.73	7.72
gif	63 KB	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
random	63 KB	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99

Table 15: Speedups for the Complete Model (10 Mbps Network) Using DEFLATE

Name	Payload Size	Encryption/Authentication Algorithm(s) Used								
		NULL	RC5	3DES	MD5	SHA-1	RC5-MD5	RC5-SHA-1	3DES-MD5	3DES-SHA-1
obj2	1 KB	0.79	0.86	0.99	0.82	0.82	0.88	0.88	1.00	1.00
progl	1 KB	1.03	1.11	1.28	1.05	1.05	1.13	1.13	1.28	1.28
paper2	1 KB	0.86	0.93	1.05	0.89	0.89	0.94	0.94	1.06	1.06
trans	1 KB	0.92	0.99	1.13	0.94	0.94	1.01	1.00	1.14	1.13
book1	1 KB	0.85	0.91	1.03	0.87	0.87	0.93	0.93	1.04	1.04
pic	1 KB	2.86	3.09	3.97	2.73	2.71	2.92	2.90	3.70	3.66
gif	1 KB	0.64	0.67	0.74	0.66	0.66	0.69	0.69	0.75	0.75
random	1 KB	0.64	0.67	0.74	0.66	0.66	0.69	0.69	0.75	0.75
obj2	4 KB	1.10	1.17	1.32	1.12	1.12	1.19	1.19	1.33	1.33
progl	4 KB	1.60	1.71	1.94	1.62	1.63	1.72	1.72	1.94	1.94
paper2	4 KB	1.17	1.24	1.39	1.19	1.19	1.26	1.26	1.39	1.40
trans	4 KB	1.40	1.49	1.66	1.42	1.42	1.50	1.50	1.67	1.67
book1	4 KB	1.10	1.17	1.30	1.12	1.12	1.18	1.18	1.31	1.31
pic	4 KB	6.46	7.16	9.26	6.28	6.23	6.92	6.88	8.88	8.81
gif	4 KB	0.79	0.81	0.85	0.80	0.80	0.82	0.82	0.86	0.86
random	4 KB	0.79	0.82	0.86	0.80	0.80	0.82	0.82	0.86	0.86
obj2	63 KB	1.28	1.36	1.53	1.30	1.31	1.38	1.39	1.54	1.55
progl	63 KB	2.01	2.16	2.48	2.06	2.06	2.20	2.21	2.50	2.51
paper2	63 KB	1.37	1.47	1.67	1.40	1.41	1.50	1.50	1.69	1.69
trans	63 KB	2.53	2.71	3.07	2.58	2.59	2.76	2.76	3.10	3.11
book1	63 KB	1.19	1.28	1.45	1.22	1.22	1.30	1.30	1.46	1.46
pic	63 KB	6.26	6.93	8.39	6.44	6.45	7.08	7.10	8.50	8.51
gif	63 KB	0.80	0.83	0.86	0.81	0.81	0.83	0.83	0.87	0.87
random	63 KB	0.80	0.82	0.86	0.81	0.81	0.83	0.83	0.87	0.87

Table 16: Speedups for the Complete Model (10 Mbps Network) Using LZS

Name	Payload Size	Encryption/Authentication Algorithm(s) Used								
		NULL	RC5	3DES	MD5	SHA-1	RC5-MD5	RC5-SHA-1	3DES-MD5	3DES-SHA-1
obj2	1 KB	1.36	1.37	1.41	1.35	1.34	1.36	1.35	1.39	1.39
progl	1 KB	1.54	1.55	1.60	1.51	1.51	1.52	1.52	1.57	1.57
paper2	1 KB	1.15	1.16	1.19	1.15	1.15	1.16	1.15	1.18	1.18
trans	1 KB	1.36	1.36	1.39	1.34	1.33	1.34	1.34	1.38	1.38
book1	1 KB	1.10	1.11	1.14	1.10	1.10	1.11	1.10	1.13	1.13
pic	1 KB	6.40	5.99	6.59	5.18	5.05	5.00	4.89	5.63	5.51
gif	1 KB	0.96	0.97	0.97	0.96	0.96	0.97	0.97	0.98	0.98
random	1 KB	0.96	0.97	0.97	0.96	0.96	0.97	0.97	0.98	0.98
obj2	4 KB	1.48	1.50	1.54	1.48	1.48	1.50	1.49	1.54	1.54
progl	4 KB	1.90	1.93	1.98	1.89	1.89	1.92	1.91	1.97	1.97
paper2	4 KB	1.34	1.35	1.39	1.34	1.34	1.35	1.35	1.39	1.39
trans	4 KB	1.63	1.65	1.69	1.63	1.62	1.64	1.64	1.68	1.68
book1	4 KB	1.25	1.27	1.30	1.25	1.25	1.27	1.27	1.30	1.30
pic	4 KB	9.43	9.32	10.05	8.60	8.53	8.64	8.52	9.40	9.34
gif	4 KB	0.95	0.96	0.97	0.96	0.96	0.96	0.96	0.97	0.97
random	4 KB	0.96	0.96	0.97	0.96	0.96	0.96	0.96	0.97	0.97
obj2	63 KB	1.57	1.60	1.64	1.58	1.58	1.60	1.60	1.64	1.64
progl	63 KB	2.05	2.08	2.14	2.06	2.05	2.09	2.09	2.14	2.14
paper2	63 KB	1.43	1.45	1.49	1.44	1.44	1.46	1.46	1.49	1.49
trans	63 KB	1.78	1.81	1.86	1.79	1.79	1.81	1.82	1.86	1.86
book1	63 KB	1.30	1.32	1.36	1.31	1.31	1.33	1.33	1.36	1.36
pic	63 KB	6.59	6.75	7.03	6.62	6.61	6.76	6.76	7.03	7.03
gif	63 KB	0.95	0.96	0.97	0.95	0.95	0.96	0.96	0.97	0.97
random	63 KB	0.95	0.96	0.97	0.95	0.95	0.96	0.96	0.97	0.97

Table 17: Speedups for the Complete Model (100 Mbps Network) Using DEFLATE

Name	Payload Size	Encryption/Authentication Algorithm(s) Used								
		NULL	RC5	3DES	MD5	SHA-1	RC5-MD5	RC5-SHA-1	3DES-MD5	3DES-SHA-1
obj2	1 KB	0.14	0.31	0.63	0.22	0.23	0.38	0.39	0.66	0.67
progl	1 KB	0.18	0.41	0.82	0.29	0.29	0.49	0.50	0.86	0.86
paper2	1 KB	0.16	0.36	0.69	0.26	0.26	0.43	0.43	0.73	0.73
trans	1 KB	0.16	0.38	0.73	0.26	0.27	0.45	0.45	0.77	0.77
book1	1 KB	0.16	0.36	0.68	0.25	0.26	0.43	0.43	0.72	0.72
pic	1 KB	0.35	0.89	2.14	0.56	0.58	1.03	1.04	2.15	2.16
gif	1 KB	0.15	0.31	0.54	0.24	0.24	0.37	0.37	0.57	0.57
random	1 KB	0.15	0.31	0.54	0.24	0.24	0.37	0.37	0.57	0.57
obj2	4 KB	0.21	0.47	0.88	0.30	0.31	0.53	0.54	0.92	0.92
progl	4 KB	0.30	0.67	1.29	0.43	0.45	0.76	0.77	1.34	1.34
paper2	4 KB	0.24	0.52	0.95	0.34	0.35	0.59	0.60	0.99	0.99
trans	4 KB	0.28	0.61	1.14	0.40	0.41	0.70	0.71	1.18	1.18
book1	4 KB	0.22	0.49	0.90	0.32	0.33	0.55	0.56	0.93	0.93
pic	4 KB	0.75	1.88	4.57	1.08	1.11	2.13	2.16	4.68	4.70
gif	4 KB	0.27	0.49	0.71	0.36	0.37	0.54	0.54	0.72	0.73
random	4 KB	0.28	0.49	0.71	0.37	0.37	0.54	0.54	0.73	0.73
obj2	63 KB	0.25	0.55	1.03	0.35	0.36	0.62	0.63	1.07	1.08
progl	63 KB	0.35	0.80	1.58	0.50	0.51	0.91	0.92	1.65	1.66
paper2	63 KB	0.25	0.57	1.10	0.36	0.37	0.65	0.66	1.14	1.15
trans	63 KB	0.46	1.04	2.01	0.65	0.67	1.18	1.19	2.09	2.10
book1	63 KB	0.22	0.50	0.95	0.31	0.32	0.56	0.57	0.99	0.99
pic	63 KB	0.85	2.07	4.60	1.22	1.26	2.39	2.42	4.83	4.86
gif	63 KB	0.29	0.51	0.73	0.37	0.38	0.55	0.56	0.74	0.74
random	63 KB	0.29	0.51	0.73	0.37	0.38	0.55	0.56	0.74	0.74

Table 18: Speedups for the Complete Model (100 Mbps Network) Using LZS

Name	Payload Size	Encryption/Authentication Algorithm(s) Used								
		NULL	RC5	3DES	MD5	SHA-1	RC5-MD5	RC5-SHA-1	3DES-MD5	3DES-SHA-1
obj2	1 KB	0.75	1.09	1.32	0.92	0.91	1.12	1.12	1.31	1.31
progl	1 KB	0.82	1.21	1.49	0.99	1.00	1.24	1.25	1.48	1.48
paper2	1 KB	0.64	0.92	1.11	0.79	0.79	0.96	0.96	1.11	1.11
trans	1 KB	0.75	1.08	1.30	0.92	0.91	1.11	1.12	1.30	1.30
book1	1 KB	0.62	0.88	1.06	0.76	0.77	0.93	0.92	1.06	1.06
pic	1 KB	2.73	4.68	6.70	2.52	2.57	3.69	3.68	5.51	5.46
gif	1 KB	0.71	0.86	0.94	0.81	0.81	0.89	0.89	0.95	0.95
random	1 KB	0.71	0.86	0.94	0.81	0.81	0.89	0.89	0.95	0.95
obj2	4 KB	0.79	1.15	1.42	0.94	0.96	1.20	1.20	1.43	1.43
progl	4 KB	0.94	1.43	1.80	1.12	1.15	1.49	1.49	1.80	1.80
paper2	4 KB	0.69	1.02	1.27	0.83	0.85	1.07	1.08	1.28	1.28
trans	4 KB	0.84	1.24	1.54	1.01	1.02	1.30	1.30	1.55	1.54
book1	4 KB	0.65	0.96	1.19	0.79	0.80	1.01	1.01	1.20	1.20
pic	4 KB	3.33	5.84	8.79	3.70	3.82	5.66	5.64	8.26	8.30
gif	4 KB	0.68	0.84	0.93	0.76	0.77	0.87	0.87	0.94	0.94
random	4 KB	0.68	0.84	0.93	0.76	0.77	0.87	0.87	0.94	0.94
obj2	63 KB	0.79	1.19	1.49	0.96	0.97	1.26	1.26	1.50	1.50
progl	63 KB	0.97	1.51	1.92	1.19	1.21	1.60	1.60	1.94	1.94
paper2	63 KB	0.71	1.08	1.35	0.86	0.88	1.14	1.14	1.37	1.37
trans	63 KB	0.89	1.35	1.68	1.08	1.09	1.42	1.42	1.70	1.70
book1	63 KB	0.66	0.99	1.23	0.80	0.81	1.04	1.05	1.25	1.25
pic	63 KB	2.55	4.38	6.05	3.24	3.29	4.66	4.70	6.12	6.14
gif	63 KB	0.66	0.83	0.93	0.74	0.75	0.86	0.86	0.93	0.93
random	63 KB	0.66	0.83	0.93	0.74	0.75	0.86	0.86	0.93	0.93

Table 19: Speedups for the Complete Model (1Gbps Network) Using DEFLATE

Name	Payload Size	Encryption/Authentication Algorithm(s) Used								
		NULL	RC5	3DES	MD5	SHA-1	RC5-MD5	RC5-SHA-1	3DES-MD5	3DES-SHA-1
obj2	1 KB	0.01	0.22	0.57	0.11	0.12	0.29	0.30	0.61	0.61
progl	1 KB	0.02	0.28	0.75	0.14	0.15	0.38	0.38	0.79	0.80
paper2	1 KB	0.02	0.25	0.64	0.13	0.14	0.33	0.34	0.68	0.68
trans	1 KB	0.02	0.26	0.67	0.13	0.14	0.35	0.35	0.71	0.71
book1	1 KB	0.02	0.25	0.63	0.13	0.14	0.33	0.34	0.66	0.67
pic	1 KB	0.04	0.59	1.89	0.27	0.29	0.76	0.78	1.94	1.95
gif	1 KB	0.02	0.23	0.50	0.13	0.13	0.30	0.30	0.53	0.53
random	1 KB	0.02	0.23	0.50	0.13	0.13	0.30	0.30	0.53	0.53
obj2	4 KB	0.02	0.33	0.81	0.13	0.15	0.41	0.41	0.85	0.85
progl	4 KB	0.03	0.47	1.18	0.19	0.21	0.58	0.59	1.23	1.24
paper2	4 KB	0.03	0.37	0.88	0.15	0.16	0.45	0.46	0.92	0.92
trans	4 KB	0.03	0.43	1.05	0.18	0.19	0.54	0.55	1.09	1.10
book1	4 KB	0.02	0.35	0.82	0.14	0.16	0.43	0.44	0.86	0.87
pic	4 KB	0.07	1.23	4.01	0.44	0.48	1.54	1.57	4.17	4.19
gif	4 KB	0.04	0.38	0.68	0.19	0.20	0.45	0.45	0.70	0.70
random	4 KB	0.04	0.39	0.68	0.19	0.20	0.45	0.46	0.70	0.70
obj2	63 KB	0.03	0.39	0.95	0.15	0.16	0.48	0.48	0.99	1.00
progl	63 KB	0.04	0.56	1.44	0.21	0.22	0.68	0.70	1.51	1.52
paper2	63 KB	0.03	0.40	1.00	0.15	0.16	0.49	0.50	1.05	1.06
trans	63 KB	0.05	0.73	1.84	0.27	0.29	0.89	0.91	1.93	1.94
book1	63 KB	0.02	0.35	0.87	0.13	0.14	0.43	0.43	0.91	0.92
pic	63 KB	0.09	1.39	4.09	0.49	0.53	1.74	1.77	4.34	4.37
gif	63 KB	0.04	0.40	0.70	0.19	0.20	0.46	0.47	0.71	0.72
random	63 KB	0.04	0.40	0.70	0.19	0.20	0.46	0.47	0.71	0.72

Table 20: Speedups for the Complete Model (1Gbps Network) Using LZS

Name	Payload Size	Encryption/Authentication Algorithm(s) Used								
		NULL	RC5	3DES	MD5	SHA-1	RC5-MD5	RC5-SHA-1	3DES-MD5	3DES-SHA-1
obj2	1 KB	0.14	0.95	1.30	0.64	0.64	1.03	1.03	1.29	1.29
progl	1 KB	0.14	1.05	1.46	0.67	0.70	1.13	1.14	1.45	1.45
paper2	1 KB	0.11	0.80	1.08	0.56	0.57	0.88	0.88	1.09	1.09
trans	1 KB	0.14	0.94	1.28	0.64	0.65	1.02	1.02	1.28	1.28
book1	1 KB	0.11	0.77	1.04	0.54	0.56	0.85	0.84	1.05	1.04
pic	1 KB	0.40	4.06	6.73	1.49	1.58	3.24	3.25	5.48	5.45
gif	1 KB	0.19	0.80	0.93	0.66	0.68	0.85	0.85	0.94	0.94
random	1 KB	0.19	0.80	0.93	0.66	0.68	0.85	0.86	0.94	0.94
obj2	4 KB	0.13	0.99	1.39	0.58	0.61	1.08	1.08	1.40	1.40
progl	4 KB	0.15	1.21	1.75	0.66	0.70	1.32	1.33	1.76	1.76
paper2	4 KB	0.12	0.87	1.25	0.51	0.53	0.96	0.97	1.26	1.26
trans	4 KB	0.14	1.06	1.50	0.61	0.64	1.16	1.16	1.51	1.51
book1	4 KB	0.11	0.82	1.17	0.48	0.51	0.90	0.91	1.18	1.18
pic	4 KB	0.44	4.63	8.49	1.87	2.03	4.74	4.75	8.00	8.06
gif	4 KB	0.17	0.78	0.92	0.56	0.58	0.82	0.82	0.93	0.93
random	4 KB	0.17	0.78	0.92	0.57	0.59	0.82	0.83	0.93	0.93
obj2	63 KB	0.13	1.02	1.45	0.55	0.58	1.12	1.13	1.47	1.47
progl	63 KB	0.15	1.27	1.87	0.67	0.70	1.41	1.42	1.90	1.90
paper2	63 KB	0.12	0.91	1.32	0.49	0.52	1.01	1.02	1.34	1.34
trans	63 KB	0.14	1.14	1.64	0.61	0.66	1.26	1.27	1.66	1.66
book1	63 KB	0.11	0.85	1.20	0.46	0.49	0.93	0.94	1.22	1.22
pic	63 KB	0.35	3.52	5.83	1.66	1.76	3.97	4.02	5.92	5.94
gif	63 KB	0.16	0.77	0.92	0.52	0.54	0.81	0.81	0.92	0.92
random	63 KB	0.16	0.76	0.92	0.52	0.54	0.81	0.81	0.92	0.92