IPSec: Performance Analysis and Enhancements

Craig A. Shue, Minaxi Gupta
Computer Science Department
Indiana University
{cshue, minaxi}@cs.indiana.edu

Steven A. Myers
Department of Informatics
Indiana University
samyers@indiana.edu

***Abstract*— Internet Protocol Security (IPSec) is a widely deployed mechanism for implementing Virtual Private Networks (VPNs). In previous work, we examined the overheads incurred by an IPSec server in a single client setting. In this paper, we extend that work by examining the scaling of a VPN server in a multiple client environment and by evaluating the effectiveness of connection credential caching. Motivated by the potential benefits of caching, we also propose a cryptographically secure cache resumption protocol for IPSec connections to reduce the connection establishment overheads.**

I. Introduction

IP packets do not have any inherent security. As a result, there is no guarantee that a received IP packet: 1) is from the claimed sender, 2) contains the original data that the sender put in it, or 3) was not sniffed during transit. *IPSec* [1], [2] (short for *IP Sec*urity) provides a method to protect IP datagrams and is commonly used in Virtual Private Networks (VPNs). It defines a method for specifying the traffic to protect, how that traffic is to be protected, and to whom the traffic is sent. For both IPv4 and IPv6, it offers the choice of two protocols: Encapsulating Security Payload (ESP) [3] or Authentication Header (AH) [4]. In order to establish and periodically refresh the necessary cryptographic parameters for both these protocols, IPSec uses another protocol called the Internet Key Exchange (IKE) [5] protocol.

While the performance of IPSec has been well studied, less attention has been devoted to enhancing it. Further, most of the work has focused only on ESP. Consequently, the current enhancements involve the selection of hashing and encryption algorithms that yield better performance for the given system (in our previous study, AES computations were quicker than 3DES in ESP) and using cryptographic hardware support to expedite ESP operations.

In this paper, we focus on enhancing the performance of IPSec by optimizing IKE. As conventional wisdom indicates, and as our previous work confirmed [6], the public key operations in IKE take three orders of magnitude more time than the operations in ESP. This implies that short connections are dominated by the IKE overheads and provides incentive for optimization. Thus, the contributions of this paper are three-fold: 1) we further examine the overheads of an IPSec server when multiple clients connect to it simultaneously, 2) we analyze 30 days of IPSec VPN usage logs to determine the utility of using caching techniques at the VPN server in order to optimize IKE, and 3) we propose a cryptographically secure cache resumption protocol for the VPN server in order to minimize IKE overheads. Our results for VPN server performance in a multiple client setting using Openswan [7], an open source implementation of IPSec, show that the IPSec overheads increase more rapidly than the corresponding overheads for native TCP/IP implementations. This indicates that, as expected, the server becomes computationally bounded with IPSec load, but not with an equivalent volume of unprotected traffic. Our investigation of caching strategies produces encouraging results: caching can cut the number of required IKE exchanges by $50 - 80\%$. Finally, the cache resumption protocol we propose can be implemented by using just six hash operations, making it at least three orders of magnitude faster than performing an IKE exchange.

The rest of the paper is organized as follows. Section II briefly describes the relevant IPSec components. Section III describes the methodology and the results of the multiple client analysis. We describe the caching-based enhancements to IPSec in Section IV. In Section V, we present a protocol to resume connections without requiring IKE. Finally, Sections VI and VII present the related work and some concluding remarks.

II. The IPSec Protocol

IPSec integrates security at the IP layer. In order to provide higher layer services that are IPSec oblivious, such as VPNs, it defines two new protocols, *Encapsulating Security Payload (ESP)* and *Authentication Header (AH)*. Both ESP and AH protocols encapsulate IP packets using ESP and AH headers respectively.

Both ESP and AH protocols can be used in either *tunnel* or in *transport* mode. The transport mode leaves the original IP header untouched and is used to protect only the upper-layer protocols. As a result, it can only be used between two end-hosts that are also cryptographic end points. The tunnel mode protects the entire IP datagram by use of encapsulation and can be used to protect traffic between two end-hosts, or two gateways (e.g. routers, firewalls), or between an end-host and a gateway. Since most VPN deployments use the tunnel mode due to its flexibility, we focus on it in this paper.

The choice between ESP and AH protocols depends on the desired level of protection for the IP datagrams. The AH protocol offers data integrity, anti-replay protection, and data source authentication but does not offer data privacy. The ESP protocol offers data privacy in addition to all the features offered by the AH protocol and is the protocol of choice for VPN deployment. Consequently, this paper focuses on the

ESP protocol used for forming VPNs, even though much of it applies to AH as well.

The selection of a cryptographic mechanism is required before any IP data can be encrypted using the ESP protocol. The available primitives include using a symmetric key between the two cryptographic end points or the public keys of the end points. Since using public key encryption is computationally expensive, IPSec uses symmetric keys. But, before IPSec can use symmetric keys to encrypt data, the symmetric keys must be exchanged. To accomplish this goal, IPSec defines the *Internet Key Exchange (IKE) protocol.*

Sections II-A and II-B describe the IKE and ESP protocols respectively. Due to space constraints, these descriptions are high-level summaries of the protocols. We encourage the reader to consult [2] and [6]for more detailed discussion of the protocols.

A. IKE Protocol

The goal of the IKE protocol[1] is to establish and maintain shared security parameters and authenticated keys between the two IPSec end points. It uses a series of messages contained in UDP datagrams, typically directed to port 500.

The IKE protocol consists of two distinct phases. The first phase establishes a symmetric IKE key between the *initiator* (typically, VPN client) and the *responder* (typically, VPN server). This key is used in the second phase to establish a symmetric IPSec key for use during ESP or AH encapsulation.

The IKE *Security Association (SA)* defines the manner in which two end points communicate; for example, this involves agreeing on the algorithm used to encrypt traffic, the hash algorithm, and the mechanism to authenticate the other end point. IKE defines 3 categories of authentication methods (with 4 individual methods) for phase one: the first method uses pre-shared keys, the next method uses digital signatures (using RSA or other digital signatures algorithms), and the last two methods use public key encryption. In both phases, the Diffie-Hellman protocol is performed in order to exchange the keys.

For better security during longer VPN sessions, IPSec provides a mechanism to periodically refresh both IKE and IPSec keys. Refreshing the IKE key entails running both IKE phases but refreshing the IPSec key only requires running the second phase again.

B. ESP Protocol

We now describe the processing of IP packets when ESP protocol is used in tunnel mode in IPSec VPNs. For processing any outbound packets, the transport layer forwards data to the IP layer which has been enhanced with the IPSec functionality. The IP layer consults a locally maintained Security Policy Database (SPD) that defines the security services afforded to the packet. The output of the SPD dictates whether the IP layer drops the packet, bypasses security, or applies security.

[1]We focus on IKEv1 here due to the availability of an open source implementation of it. No open source implementation for IKEv2 [8] is currently available.

If security is to be applied, the appropriate IPSec Security Association (SA) is consulted by looking up the SA database (SADB) and the entire IP packet is encrypted and placed inside another IP packet. To facilitate the processing of the packet at the other end, an ESP header containing SA mapping information and a sequence number (to prevent replay attacks) is inserted between the new IP header and the original encrypted IP packet. An ESP trailer containing an Integrity Check Value (ICV) is also inserted at the end of the new IP packet before sending it out.

III. Performance Analysis of IPSec VPN Servers

In our previous study [6] using Openswan [7], an open source implementation of IPSec (based on the earlier FreeS/WAN project [9]), we focused on the overheads for individual security operations for IPSec protocols in a single client setting. We utilized two methods to analyze the performance impact of the ESP protocol, the IKE protocol, various encryption algorithms, and various cryptographic key sizes: 1) measuring run-times for individual security operations and 2) replacing various IPSec components with no-ops and recording the speed-up in the run-time of various IPSec phases. We found that 1) the overheads of the IKE protocol at the VPN server are three orders of magnitude higher than those incurred by ESP for processing a single packet, 2) cryptographic operations contribute $23-55\%$ of the overheads for IKE protocol and $34-55\%$ for ESP, 3) digital signature generation and Diffie-Hellman computations are, as expected, the largest contributors of overheads during the IKE process and little overhead can be attributed to the symmetric key encryption and hashing, and 4) symmetric key encryption is the most expensive operation during the ESP process.

In this section, we extend our previous work by analyzing the performance of an IPSec VPN server when multiple clients connect to it simultaneously. The goal was to compare the difference between a native TCP/IP implementation and when IPSec is in use. We measured how the file transfer time changed as the number of concurrent VPN client connections increased from one to six. Specifically, each client in our experiments downloaded a 56.47MB file from the VPN server using the `wget` utility. We choose 128-bit AES with MD5 hashing for these experiments because this configuration exhibited the least overhead in our previous study. Further, any IKE or ESP key refreshments are prevented while conducting the file transfer to avoid causing variations in the results. We conducted 25 trials for each client configuration. We now discuss the details of the experimental environment and the results obtained.

A. Experimental Environment

To conduct our experiments, we used ten x86 Dell Optiplex GX Pentium IV machines (see Table I for their individual specifications). They were connected to each another through 1Gbps and 100Mbps Ethernet switches. The first machine was used as the server and ran Debian Linux [10] with a $2.6.8$ kernel. We disabled the native IPSec support and instead used

	CPU/RAM	NIC	Kernel	IPSec
1	1.66GHz/512MB	100Mbps	2.6.8	KLIPS
2-4	1.6GHz/512MB	100Mbps	2.6.14	Native
5-7	1.8GHz/512MB	1Gbps	2.6.14	Native
8-10	2.8GHz/1GB	1Gbps	2.6.14	Native

TABLE I

SPECIFICATIONS OF EXPERIMENTAL MACHINES.

Kernel Level Internet Protocol Security (KLIPS), the shim provided by Openswan version $2.3.1dr3$ for IPSec support. The rest of the machines were used as clients and had Gentoo Linux [11] with a $2.6.14$ kernel. We retained the native IPSec kernel support for these machines.

B. Experimental Results

We used machine 1 as the VPN server and machines $2-7$ as clients for the IPSec case, since these tests were not bandwidth bound. However, for the native TCP/IP tests, we had to move to machines $5-10$ since bandwidth became the bottleneck in these tests and machines $2-4$ had slower NICs. Figure 1 shows the result of this experiment when the number of simultaneous clients varied from $1-6$. Each line in Figure 1 represents the total file transfer time over *http* when IPSec is in use and when native TCP/IP is used. The observed transfer time includes the overheads due to the ESP processing during the VPN connection as well as actual file transfer time over the network using the `wget` utility. *The results indicate that the IPSec overheads increase more rapidly than that of native TCP/IP, because they are computationally bound.*

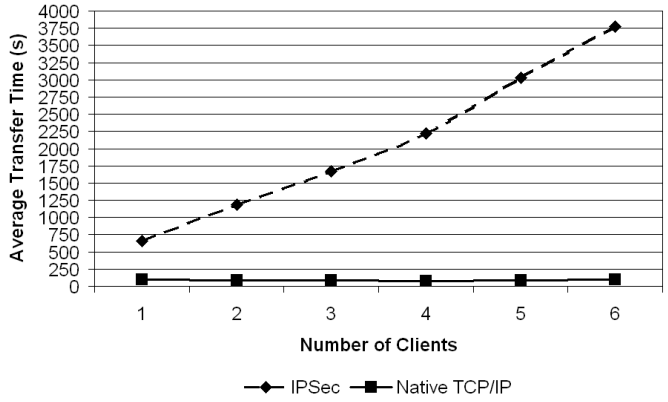

Fig. 1. File transfer time for concurrent transfers (56.47MB file transferred 25 times to each client using the `wget` utility).

The average client throughput in Figure 1 is close to $\frac{\text{single client throughput}}{\text{number of clients}}$. Using this approximation, we project the average throughput under a larger number of clients. This projection indicates that for as few as 25 simultaneously connected clients, the throughput falls off to about 88KB/s while at 6 clients, it was close to 375KB/s.

IV. OPTIMIZING IKE THROUGH CACHING

As we previously noted, the IKE overheads are significant in comparison to ESP overheads for very short, low traffic connections. While ESP overheads may dwarf the IKE overheads on high traffic connections, VPN servers may have different traffic patterns. This section investigates caching-based enhancements to IKE. We begin by analyzing empirical data in Section IV-A. In Section IV-B, we evaluate the effectiveness of various caching strategies to reduce the number of IKE exchanges in which the VPN server participates.

While the notion of caching credentials may seem to run counter to the re-keying present in IPSec, a further examination of the security goals shows they are not necessarily in opposition. Re-keying allows IPSec to ensure Perfect Forward Security, which may be desirable in some instances, but not required in others; it also guards against weaknesses in ESP protocols that might become weak when large amounts of encrypted data are exposed to an adversary. We note that for ESP protocols based on modern ciphers,such as AES, there is little concern that weaknesses in ESP will result in security issues for relatively small amounts of traffic. Further, caching and re-keying can be used in conjunction to perform re-keying only when it is actually necessary, rather than being dictated by possibly intermittent network connectivity.

A. VPN Workload Characterization

We analyzed several 30-day snapshots of logs from the VPN servers deployed at Indiana University's Bloomington campus. Due to space constraints, only the results of the latest snapshot are presented here. The logs contained all the disconnection and connection times for a 30-day window, from August 19 to September 18, 2006 for wired and wireless VPN clients respectively[2]. Additionally, the logs contained: connection date and time, disconnection date and time, an anonymized *username*, a session identifier, and the total amount of data downloaded and uploaded by each client.

	Wireless	Remote
Total Connections	149564	241291
Total Bytes In (in GB)	452.1	2008.2
Average (in MB)	3.09	8.52
Total Bytes Out (in GB)	1010.45	4907.79
Average (in MB)	6.91	20.82
Total Connection Length (in days)	6577.83	31628.64
Average (in hours)	1.05	3.14
Total Unique Users	11250	15579

TABLE II

LOG CHARACTERISTICS

Table II contains the details about the total number of connections, data transmitted, connection durations, and total number of unique users. In addition to this information, Figures 2, 3, and 4 show the daily number of connections, the number of connections lasting various time periods, and the amount of data transferred to and from the VPN servers respectively for both wired and wireless data sets.

Figures 2, 3, 4, and 5 contain several interesting pieces of information: 1) the VPN servers see a wide variation in the

[2]Our campus's wireless LAN deployment requires the users to use VPN. The wired VPN clients are connected remotely.

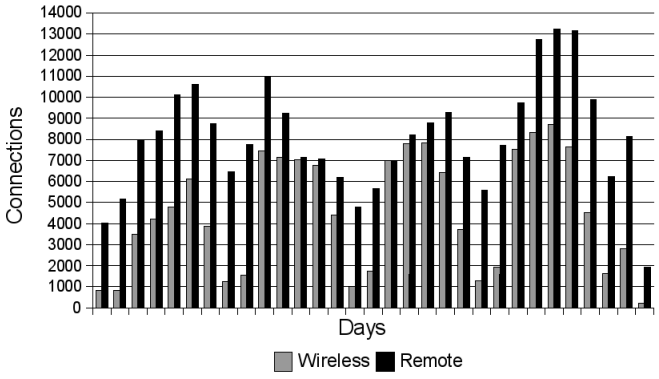

Fig. 2. Total number of connections per day.

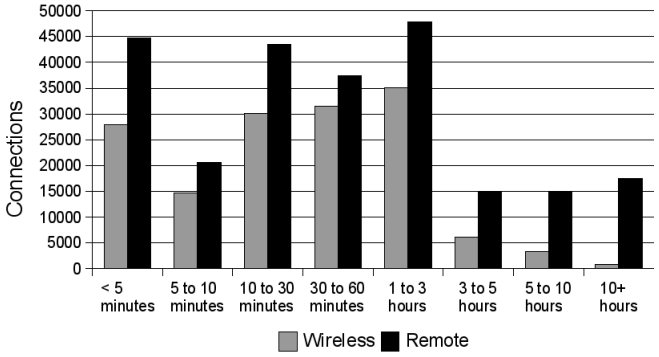

Fig. 3. Duration of connections.

number of daily requests and this variation is slightly more pronounced for wireless VPN clients, 2) most connections transferred 1Mbyte of data or less and wireless clients tend to transfer less data than the wired clients, and 3) while a significant number of connections lasted < 5 minutes, most connections lived for 10 minutes to 3 hours and the wireless clients tended to establish shorter VPN connections in general.

B. Caching Strategies

As Table II shows, the number of unique users is at least an order of magnitude less than the total number of connections.

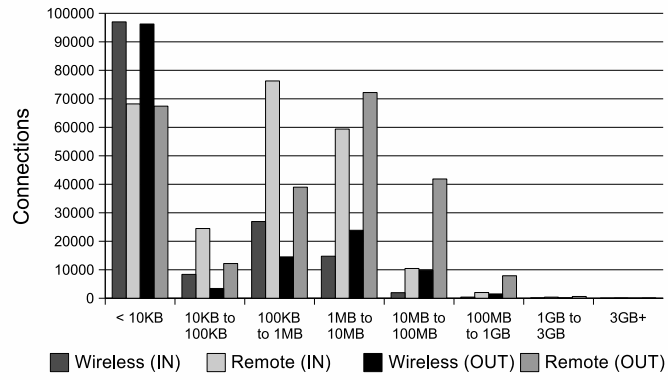

Fig. 4. Volume of data transferred to and from the VPN servers.

We further examined this by binning the number of times each user returns. Figure 5 shows the number of visits per user. We conclude that the vast majority of users connected more than once during our data collection period. This indicates that techniques to cache Security Association information may be worthwhile.

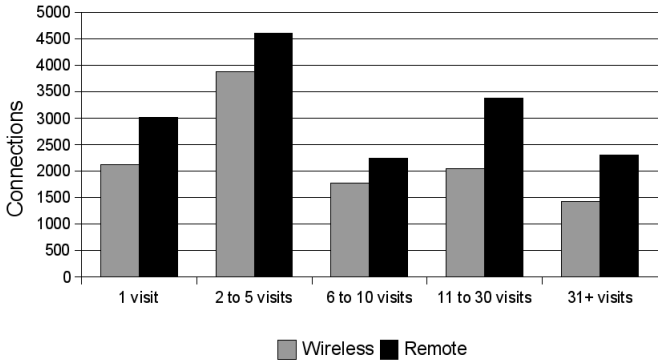

Fig. 5. Number of visits per user.

The two-phased IKE protocol helps establish two sets of secret keys: the IKE keys, which are used only during the second phase of the IKE protocol, and the IPSec keys, which are used by the ESP protocol. Caching both sets of helps avoid both phases of the IKE protocol while caching the IKE keys only helps avoid running the first phase of the IKE protocol. We assume in the subsequent discussion that both the IKE and IPSec key sets are cached.

Next, we utilized the log files to determine how many IKE phase 1 and phase 2 exchanges would occur under various caching strategies. In particular, we tested three caching strategies, the description of which is presented below:

1) Clocked Lifetime: This strategy allows a client to resume using cached credentials only if the actual time elapsed since the last IKE exchange is less than the configured value (the resumption "lifetime"). For example, for a clocked lifetime of 8 hours, a connection could initially last two hours, disconnect, and resume three hours later without having to perform IKE. However, given the same lifetime, a client would be unable to resume if it initially connected for two hours and returned seven hours later. Figure 6 shows the number of IKE operations that would be required for different IKE lifetimes in the VPN logs under this method. In comparison with the $241,291$ ($149,564$) connections in the actual wired (wireless) logs, this strategy cuts down the number of IKE exchanges to less than 50% for a configured clocked lifetime of

2) Established Lifetime: This strategy only counts the actual duration a connection is established towards the lifetime. As an example of this technique, for an 8 hour IKE lifetime, a connection could initially last 2 hours and resume 20 hours later without having to perform an IKE operation. This strategy cuts down the number of IKE exchanges to less than 20% of the current value for a configured established lifetime of 2 hours. The detailed results have been omitted due to space constraints.

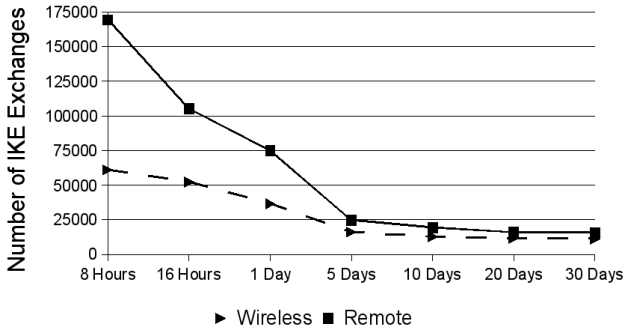

Fig. 6. Number of IKE exchanges required under clocked lifetime caching strategy.

3) Data Lifetime: Under this strategy, the lifetime of a connection is based on the amount of data transferred. Connections would be allowed to resume without requiring an IKE exchange only if the amount of data transferred is less than the configured lifetime. This strategy cuts down the number of IKE exchanges to about 25% of the current value for a configured data lifetime of 20MBytes in the wireless case and for a data lifetime of 50MBytes in the remote case. The detailed results have again been omitted due to space constraints.

Though a direct comparison across the caching strategies tested is difficult due to the differences in potential security implications, *we conclude that caching can significantly reduce the number of IKE exchanges required.*

V. Protocol to Resume From Cached State

To facilitate the resumption of previously cached connections, IPSec clients and servers must be modified. Both need to store the negotiated IPSec symmetric keying information upon connection termination in order to use it for connection resumption. Additionally, a protocol must be created to negotiate the resumption. There is precedent for attempts to optimize IPSec connections. In particular, the IETF has developed a protocol for using Kerberos authentication to optimize IPSec connection establishment [12]. Here, we propose a slight modification of the SKID3 protocol [13] that resumes cached IPSec connections in a secure way.

The first issue to address in such a protocol is that of storing credentials. The issue is trivially addressed for connections that terminate in an orderly fashion since both parties can deactivate and store the credentials upon transmission/receipt of the connection termination messages. For connections that end abruptly, a strategy similar to that used by Openswan, *dead peer detection*, in which keep-alive messages are used to detect connection terminations, can be used. Upon such detection, the credentials can be stored.

Our protocol for resuming a session is depicted in Figure 7. This protocol would work with both IKE version 1 and 2. The process begins when one of the parties involved in a previous IPSec connection issues a request to re-establish the connection. In this request, the initiator must provide its identity so the responder knows what connection is being resumed as well as a nonce value to prove liveness.

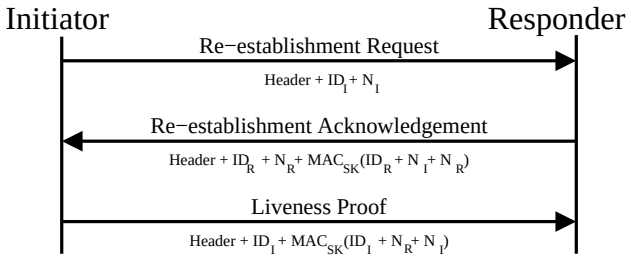

Fig. 7. The protocol for resuming a cached connection.

The responder must determine if it has a valid record (connection information, symmetric authentication key, etc.) of the connection that the initiator is requesting to resume. If not, the responder replies with a negative acknowledgment, refusing to grant the resumption. This can happen because either party can expunge records of connections that are over a certain age or when it is faced with storage constraints. The initiator is then required to under-go the IKE process in order to communicate via IPSec.

If the responder has a valid record of the previous connection, it can use the credentials from the previous IPSec session. It then replies with an acknowledgment including its own ID, the initiator's nonce, its own nonce, and a keyed message authentication code (MAC) of the ID and nonces. This MAC is keyed with the IPSec authentication key from the previous IPSec connection[3] (i.e., the IPSec authentication key is included as input to the MAC).

Upon receiving the recipient's message, the initiator verifies the MAC was constructed correctly. If the MAC and its own nonce value are correct, the initiator is assured the responder is authentic. It then replies with its ID and a keyed MAC of the nonce values and its own ID. The responder can verify the MAC. If it is correct, the responder is assured the initiator is authentic. Upon completion of the re-establishment process, both sides of the connection can use IPSec with the keying information from the previous connection.

The cryptographic overhead of the proposed re-establishment protocol is that of four MAC constructions at both the initiator and responder (two to create the MACs and two to verify the other party's MACs). These operations are three to four orders of magnitude faster than the public key or Diffie-Hellman operations required in IKE, providing a significant performance savings.

There are security implications of caching credentials. If either end point is compromised, its credentials would be lost, allowing an attacker to use them to establish connections. However, these risks are similar to storing private keys and pre-shared secrets. Therefore, we do not regard key caching to be an increased security risk.

[3]The keying information for actual payload packets is negotiated in the second phase of IKE. This information is used to generate an authentication key for IPSec packets. This authentication key is cached and used in this protocol to authenticate resumption.

We note that the storage requirements for the credential cache are reasonably small. Each cached entry would need to store the identity of the remote party and keying information. The ID field used in IKE has an 8 byte header, followed by the identifier payload. If IPv4 addresses are used, the identifier would be 12 bytes. The record would also require the storage of symmetric authentication, encryption keys, and security association information. If we assume two 256 bit keys, this information could reasonably be encoded in under 100 bytes. Therefore, a server caching 50,000 unique records would thus require less than 5 megabytes for storage, which is easily accommodated with modern DRAM capacity.

VI. Related Work

IPSec was the focus of work in [14], where the authors examined the ESP and AH encapsulation overheads. However, a comprehensive picture of IPSec overheads was not presented because the performance penalty associated with the IKE process was not examined. Further, the difference in overheads due to different encryption algorithms and key sizes was not investigated. In [6], we presented work complementary to the current paper, focusing on the timing measurements of IKE and ESP using MD5 for hashing. In this paper, we extend that work by performing concurrent transfer analysis and examining approaches to cache IPSec keys.

The work in [15] introduces a new protocol for key exchanges. In doing so, the authors examined various aspects of IKE, including the complexity of the implementation, the role of re-keying, and susceptibility to denial-of-service attacks. In [12], the authors propose a method for using Kerberos to authenticate a key exchange for IKE while reusing the second mode, quick mode. This work demonstrates that alterations to IKE can boost performance without sacrificing security.

The work in [16] analyzed IPSec using a macroscopic approach. The authors examined the performance of IPSec with and without hardware acceleration by examining the throughput and transfer times of sample files. In [17], the authors discuss the affect of hardware acceleration on AES implementations when used for IPSec, specifically regarding rapid symmetric key switches. Work in [18] provides general discussion on hardware acceleration in IPSec and mentions existing commercial accelerators. These works are complementary to our work and can be utilized for additional performance enhancement.

VII. Conclusion

In this paper, we evaluated the performance of IPSec-based VPN servers in a multiple client setting and found that IPSec does not scale as well as the native TCP/IP implementations. This analysis makes a strong case for performance optimization. Since IKE overheads can be a significant percentage of the overheads, especially for the VPN connections that last for a short duration, we focus on optimizing IKE in this paper. In order to reduce the connection establishment overheads, we investigated various caching strategies for connection-related information using $30-$day VPN traces. Our analysis concluded that caching can be an effective strategy to reduce the connection overheads. The choice of caching strategy and duration depends on the perceived vulnerabilities in any environment. We also proposed a cryptographically secure cache resumption protocol.

Our work in this paper focused primarily on IKE optimization. Though ESP takes very little computational overhead per packet, when a large number of packets are processed in a connection, ESP overheads can dominate the IKE overheads. As our previous work revealed, hashing and encryption play a significant role in the time required for ESP packet processing. The choice of an optimal hashing algorithm, encryption algorithm, and cipher size can significantly improve ESP performance. Additionally, as shown in [16], hardware accelerators can be used to further boost performance.

Acknowledgments

We would like to thank Tom Zeller for insights on deployed VPNs and for providing VPN logs from Indiana University's VPN deployment. We would also like to thank Rob Henderson for his technical assistance in conducting the experiments.

References

[1] S. Kent and R. Atkinson, "Security architecture for the internet protocol," RFC 2401 (Proposed Standard), Internet Engineering Task Force, Nov. 1998, updated by RFC 3168.

[2] N. Doraswamy and D. Harkins, *IPSec: the new security standard for the Internet, intranets, and virtual private networks*, 1st ed. Prentice Hall, 1993.

[3] S. Kent and R. Atkinson, "IP encapsulating security payload," RFC 2406 (Proposed Standard), Internet Engineering Task Force, Nov. 1998.

[4] ——, "IP authentication header," RFC 2402 (Proposed Standard), Internet Engineering Task Force, Nov. 1998.

[5] D. Harkins and D. Carrel, "The Internet Key Exchange (IKE)," RFC 2409 (Proposed Standard), Internet Engineering Task Force, Nov. 1998.

[6] C. Shue, Y. Shin, M. Gupta, and J. Y. Choi, "Analysis of IPSec overheads for VPN servers," IEEE ICNP's NPSec Workshop, 2005.

[7] X. Corporation, "Openswan Web-site," 2004, http://www.openswan.org/.

[8] C. Kaufman, "Internet Key Exchange (IKEv2) protocol," Draft IKEv2, Internet Engineering Task Force, September 2004.

[9] J. Gilmore, H. Daniel, R. Briggs, H. Redelmeier, C. Schmeing, and S. Sgro, "FreeS/WAN Project Web-site," http://www.freeswan.org/.

[10] "Debian Linux Project Web-site," http://www.debian.org/.

[11] "Gentoo Linux Project Web-site," http://www.gentoo.org/.

[12] S. Sakane, K. Kamada, M. Thomas, and J. Vilhuber, "Kerberized internet negotiation of keys (KINK)," RFC 4430 (Proposed Standard), Internet Engineering Task Force, mar 2006.

[13] B. Schneier, *Applied Cryptography, Protocols, Algorithms, and Source Code in C*. John Wiley and Sons, Inc., 1996.

[14] G. C. Hadjichristophi, N. J. Davis IV, and S. F. Midkiff, "IPSec overhead in wireline and wireless networks for web and email applications," in *22nd IEEE International Performance, Computing, and Communications Conference*, Phoenix, Arizona, April 2003.

[15] W. Aiello, S. Bellovin, M. Blaze, R. Canetti, J. Ioannidis, A. Keromytis, and O. Reingold, "Efficient, DoS-resistant, secure key exchange for internet protocols," in *ACM Computer and Communications Security (CCS) Conference*, November 2002.

[16] S. Miltchev, S. Ioannidis, and A. Keromytis, "A study of the relative costs of network security protocols," USENIX, 2002.

[17] D. Whiting, B. Schneier, and S. Bellovin, "AES key agility issues in high-speed IPSec implementations," Counterplane Internet Security, May 2000.

[18] I. Andoni, P. Chodowiec, and J. Radzikowski, "Hardware implementation of IPSec cryptographic transformations," 2001, http://ece.gmu.edu/courses/ECE636/project/reports/IAn_PCh_JRa.pdf.